University of Alberta

GENETIC ALGORITHMS FOR SCHEDULING IN MULTIUSER MIMO WIRELESS COMMUNICATION SYSTEMS

by

Robert Charles Elliott

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Communications

Department of Electrical and Computer Engineering

Spring 2011
Edmonton, Alberta

Examining Committee

Witold A. Krzymień, Electrical and Computer Engineering

Jeffrey Andrews, Electrical and Computer Engineering, University of Texas at Austin

Mike MacGregor, Computing Science

Hai Jiang, Electrical and Computer Engineering

Chintha Tellambura, Electrical and Computer Engineering

To my parents,
who may not understand everything in this thesis,
and yet are always understanding.

Abstract

Multiple-input, multiple-output (MIMO) techniques have been proposed to meet the needs for higher data rates and lower delays in future wireless communication systems. The downlink capacity of multiuser MIMO systems is achieved when the system transmits to several users simultaneously. Frequently, many more users request service than the transmitter can simultaneously support. Thus, the transmitter requires a scheduling algorithm for the users, which must balance the goals of increasing throughput, reducing multiuser interference, lowering delays, ensuring fairness and quality of service (QoS), etc.

In this thesis, we investigate the application of genetic algorithms (GAs) to perform scheduling in multiuser MIMO systems. GAs are a fast, suboptimal, low-complexity method of solving optimization problems, such as the maximization of a scheduling metric, and can handle arbitrary functions and QoS constraints. We first examine a system that transmits using capacity-achieving dirty paper coding (DPC). Our proposed GA structure both selects users and determines their encoding order for DPC, which affects the rates they receive. Our GA can also schedule users independently on different carriers of a multi-carrier system. We demonstrate that the GA performance is close to that of an optimal exhaustive search, but at a greatly reduced complexity. We further show that the GA convergence time can be significantly reduced by tuning the values of its parameters.

While DPC is capacity-achieving, it is also very complex. Thus, we also investigate GA scheduling with two linear precoding schemes, block diagonalization and successive zero-forcing. We compare the complexity and performance of the GA with “greedy” scheduling algorithms, and find the GA is more complex, but performs better at higher

signal-to-noise ratios (SNRs) and smaller user pool sizes. Both algorithms are near-optimal, yet much less complex than an exhaustive search. We also propose hybrid greedy-genetic algorithms to gain benefits from both types of algorithms.

Lastly, we propose an improved method of optimizing the transmit covariance matrices for successive zero-forcing. Our algorithm significantly improves upon the performance of the existing method at medium to high SNRs, and, unlike the existing method, can maximize a weighted sum rate, which is important for fairness and QoS considerations.

Acknowledgements

I must begin by expressing my sincere gratitude to my graduate supervisor, Dr. Witold A. Krzymień, for his guidance and support throughout my doctoral studies. His insight, advice, and assistance with my research program and his financial support have been invaluable. I would also like to thank the members of my candidacy examination committee, Dr. Ivan Fair, Dr. Hai Jiang, Dr. Joe Culberson, and committee chair Dr. Vincent Gaudet, as well as my final Ph.D. defense committee, Dr. Jeffrey Andrews, Dr. Mike MacGregor, Dr. Hai Jiang, Dr. Chintha Tellambura, and committee chair Dr. Tongwen Chen. All of their input and feedback have been very beneficial.

I would like to gratefully acknowledge the funding and research environment provided by TRLabs. Additional funding provided by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Alberta Informatics Circle of Research Excellence (iCORE), the Alberta Ingenuity Fund, Huawei Technologies, and the Rohit Sharma Professorship is also gratefully acknowledged. This work made use of the infrastructure and computational resources of AICT (Academic Information and Communication Technologies) at the University of Alberta, and of WestGrid.

I have appreciated the academic and casual discussions over the years with my fellow graduate students and colleagues at the University of Alberta and TRLabs. I must particularly thank Dr. Shreeram Sigdel, with whom I collaborated on my work of scheduling algorithms for BD and SZF. I would also like to thank Dr. David Mazzarese, Dr. Mohsen Eslami, Dr. Kevin Jacobson, Arash Talebi, and Saeed Kaviani for their help and input. I also thank Chris Van Kleeck for his assistance with proofreading my thesis.

I would also like to thank the fine folks at the German Aerospace Centre (DLR) in Oberpfaffenhofen, Germany, for their accommodation and assistance during my stay there as a visiting researcher in 2005. I would particularly like to acknowledge Ronald Raulefs, Jutta Uelner, Alexander Arkhipov, Stefan Kaiser, Simon Plass, Stephan Sand, and Armin Dammann. My apologies if I have accidentally forgotten to list someone.

Last, but certainly not least, I must thank my parents Mervin and Susie. I have had their constant patience, encouragement, and loving support through all my endeavours both academic and otherwise. They have stood by me through both the happy and the challenging moments of my graduate studies. This thesis would not have been possible without them.

Table of Contents

1 Introduction 1

1.1 Motivation 1
1.2 MIMO Wireless Communications 2
1.3 Orthogonal Frequency Division Multiplexing 2
1.4 Multiuser MIMO Systems 3
1.5 MIMO Multiuser Scheduling 4
1.6 Research Goals and Summary of Contributions 5
1.7 Organization of the Thesis 8

2 Background on MIMO, Precoding, and Scheduling 9

2.1 Single-User MIMO Systems 9
2.2 Multiuser MIMO Systems 10
2.2.1 MAC, BC, DPC, and Duality 12
2.3 MIMO Precoding and Beamforming 15
2.4 MIMO Multiuser Scheduling 17

3 Genetic Scheduling Algorithms for Downlink Transmission in MIMO Single- and Multi-Carrier Systems with Dirty Paper Coding 22

3.1 Introduction 22
3.2 General Design and Optimal Solution for MIMO Multi-Carrier Scheduling 24
3.2.1 Optimal Solution: Mixed Integer Programming and Power Waterfilling 24
3.2.2 Example Scheduling Criteria and Their Utility Functions 26
3.3 Genetic Algorithms 28
3.3.1 General Description 28
3.3.2 Genetic Algorithm for Order-Dependent Precoding and Scheduling 30
3.4 Multiuser MIMO System Model with Single and Multiple Carriers 37
3.4.1 Wireless Channel Model 37
3.4.2 Physical Layer Model 39
3.4.3 Medium Access Control Layer Model 40
3.5 Simulation Results 41
3.5.1 Single-Carrier Results 41
3.5.2 Multi-Carrier Results 46
3.5.3 Convergence 49
3.5.4 Runtime / Complexity Comparison 52
3.6 Conclusion 55

4 Impact on and Improvement of the Convergence of GA Scheduling from Parameter Tuning and Change in Crossover Method 57

4.1 Introduction 57
4.2 Problem Description 57
4.3 Simulation Setup and Results 59
4.3.1 Simulation Setup 59
4.3.2 Simulation Results 59
4.4 Further Discussion 67

4.4.1 Interpretation of Equation for Parameter Values 67
4.4.2 Other SNRs and Utility Functions 69
4.5 Uniform Crossover 71
4.6 Conclusion 75

5 Genetic, Greedy, and Hybrid Scheduling Algorithms for Block Diagonalization and Successive Zero-Forcing 76

5.1 Introduction 76
5.2 System Model with Linear Precoding 76
5.2.1 Block Diagonalization 77
5.2.2 Successive Zero-Forcing 79
5.3 Scheduling Algorithms 81
5.3.1 Genetic Algorithms 81
5.3.2 Greedy Algorithms 83
5.3.3 Hybrid Algorithms 85
5.4 Complexity Analysis 86
5.4.1 Complexity of Various Matrix Operations 86
5.4.2 Complexity of Genetic Algorithm for Block Diagonalization 87
5.4.3 Complexity of Genetic Algorithm for Successive Zero-Forcing 89
5.4.4 Complexity of Greedy Algorithm for Block Diagonalization 93
5.4.5 Complexity of Greedy Algorithm for Successive Zero-Forcing 94
5.4.6 Complexity of Hybrid Algorithms 95
5.5 Simulation Results 96
5.5.1 Block Diagonalization 97
5.5.2 Successive Zero-Forcing 101
5.5.3 Hybrid Algorithm 1 105
5.5.4 Hybrid Algorithm 2 108
5.5.5 Further Discussion 111
5.6 Conclusion 112

6 Improved Covariance Optimization for Successive Zero-Forcing Weighted and Unweighted Sum-Rate Maximization 114

6.1 Introduction 114
6.2 SZF Covariance Optimization 115
6.2.1 Problem Discussion 115
6.2.2 Proposed Conjugate Gradient Projection Method 117
6.3 Simulation Results 121
6.3.1 Unweighted Sum-Rate Performance of Proposed CGP Algorithm 122
6.3.2 Weighted Sum-Rate Performance of Proposed CGP Algorithm 125
6.3.3 Updated Scheduling Performance of GA and GrA 128
6.3.4 Original Covariance Method for User Selection with CGP for Sum-Rate Maximization 129
6.4 Conclusion 132

7 Conclusions and Future Work 134

7.1 Conclusions 134
7.2 Future Work 136

Bibliography 138

A Validation of the Simulation Model 150

A.1 Complex Gaussian Verification 150
A.2 Error in Monte Carlo Simulations 152
A.3 Comparison with Published Results 155

B Optimality Conditions for DPC BC Scheduling 157

C Least Squares Polynomial Fit for GA Tuning 162

D Existing Method to Find Covariance Matrices for Successive Zero-Forcing 163

D.1 MAC Waterfilling 163
D.2 MAC to BC Transformations 164
D.3 SZF Null Space Projections 165

E Supplemental Simulation Results for Scheduling Algorithms under Block Diagonalization and Successive Zero-Forcing 167

F Derivation of Gradient of SZF Weighted Sum Rate 174

G Supplemental Simulation Results for Scheduling Algorithms under SZF with Conjugate Gradient Projection Covariance Optimization Method 176

List of Tables

Table 3.1: Runtime comparison of genetic and exhaustive search scheduling algorithms in terms of number of utility function evaluations required 53

Table 5.1: Adaptive mutation rate parameter values used for varying numbers of active users in pool (K) and varying numbers of simultaneously supportable users (K_0) 82

Table 5.2: Simplified greedy user scheduling algorithm for BD 84

Table 5.3: Simplified greedy user scheduling algorithm for SZF 85

Table 5.4: Summary of the complexity orders of different user scheduling algorithms ... 96

Table 6.1: CGP algorithm for SZF covariance optimization 119

List of Figures

Figure 2.1: Block diagram of a single-user MIMO system. 9

Figure 2.2: (a) Block diagram of MIMO multiple access channel. (b) Block diagram of MIMO broadcast channel. 11

Figure 2.3: Typical dirty paper coding achievable rate region for a 2×2 MIMO broadcast channel. 14

Figure 3.1: Flow diagram of a general case genetic algorithm. 29

Figure 3.2: Two typical chromosomes for single-carrier DPC scheduling with $N_S = 4$ and $K = 10$. (a) Users 2, 4, 7, and 9 are scheduled and have order numbers '10', '11', '00', and '01', respectively. The users are therefore encoded in the order {7,9,2,4}. (b) Users 4 and 8 are scheduled and have order numbers '11' and '01', respectively. They are therefore encoded in the order {8,4}. The remaining two order numbers in the tail of the chromosome are ignored. 32

Figure 3.3: Example of GA operation for multi-carrier DPC scheduling with four subcarriers, four transmit antennas, and 10 active users. (a) Two typical chromosomes and a random crossover location. (b) Crossover operation. (c) Mutation operation. (d) Correction of invalid child chromosomes. 34

Figure 3.4: Flow diagram of the genetic scheduling algorithm. 37

Figure 3.5: Conceptual power spectrum $P(f)$ vs. frequency f of single-carrier and multi-carrier transmissions with same normalized total bandwidth W_T. Total transmit power is divided equally among carriers. (a) Single carrier. (b) OFDM, 4 subcarriers. (c) FDM, 4 subcarriers. 39

Figure 3.6: Performance of maximum throughput scheduling versus SNR for a (N_R,M_T,K) single-carrier MIMO system implemented via GA and ES. (a) $N_R = 1$, $M_T = 2$, and $K = 10$, 20. (b) $N_R = 2$, $M_T = 2$, and $K = 10$, 20. (c) $N_R = 1$, $M_T = 4$, and $K = 10$, 20. 42

Figure 3.7: Performance of maximum throughput scheduling at SNR = 10 dB versus the number of active users for an (N_R,M_T) single-carrier MIMO system implemented via GA and ES. 43

Figure 3.8: Performance of proportionally fair scheduling versus SNR for an (N_R,M_T,K) single-carrier MIMO system implemented via GA and ES. 44

Figure 3.9: Distributions of average rate per user and instantaneous sum-throughput for PF scheduling in an (N_R,M_T,K) single-carrier MIMO system at SNR = 10 dB 45

Figure 3.10: Performance of maximum throughput scheduling vs. SNR for an (N_R,M_T,K) multi-carrier MIMO system implemented via GA and ES. (a) $N_R = 1$, $M_T = 2$, and $K = 10$, 20. (b) $N_R = 2$, $M_T = 2$, and $K = 10$, 20. (c) $N_R = 1$, $M_T = 4$, and $K = 10$, 20. 46

Figure 3.11: Performance of proportionally fair scheduling versus SNR for an (N_R,M_T,K) multi-carrier MIMO system implemented via GA and ES. (a) K = 10, N_R = 1, 2, and M_T = 2, 4. (b) K = 20, N_R = 1, 2, and M_T = 2, 4. 47

Figure 3.12: Distributions of number of simultaneously scheduled users under the proportional fairness scheduling criterion at SNR = 10 dB for an (N_R,M_T,K) multi-carrier MIMO system. 48

Figure 3.13: Distributions of head-of-line delays per user under the PF criterion at SNR = 10 dB for single- and multi-carrier transmission and various values of (N_R,M_T,K). 49

Figure 3.14: Average convergence of GA versus generations for the maximum throughput scheduling criterion at an SNR of 10 dB and various values of (N_R,M_T,K).... 50

Figure 3.15: (a) Performance of single-carrier MT GA scheduling vs. SNR for (N_R,M_T,K) = (1,2,20) and (1,4,10), each with {N_p,N_g} = {10,10} and {20,5}. (b) Difference in sum-throughput for {N_p,N_g} = {20,5} compared to {10,10}. 51

Figure 3.16: Distributions of number of generations required for GA convergence to the optimum utility function value for the maximum throughput scheduling criterion at an SNR of 10 dB and various values of (N_R,M_T,K). 51

Figure 4.1: Distributions of the adaptive mutation rate for the random initial first generation of the genetic scheduling algorithm for DPC with β_1 = 1.2 and β_2 = 10, an SNR of 10 dB, and various values of (N_R,M_T,K). 58

Figure 4.2: Distributions of number of generations required to converge to optimal utility function value for (N_R,M_T,K) = (1,2,10) at an SNR of 10 dB; β_1 = 1.2 (constant), β_2 variable..... 60

Figure 4.3: Distributions of number of generations required to converge to optimal utility function value for (N_R,M_T,K) = (1,2,10) at an SNR of 10 dB; β_1 variable, β_2 = 10 (constant). 60

Figure 4.4: Number of generations required to converge on average to within 1% of optimal utility function value as a function of β_1 and β_2 with (N_R,M_T,K) = (1,2,20) and an SNR of 10 dB..... 61

Figure 4.5: Number of generations required to converge on average to within 0.1% of optimal utility function value as a function of β_1 and β_2 with (N_R,M_T,K) = (1,2,20) and an SNR of 10 dB..... 61

Figure 4.6: Number of generations required to converge on average to within 0.1% of optimal utility function value as a function of β_1 and β_2 with 10 dB SNR and various (N_R,M_T,K). (a) (N_R,M_T,K) = (2,2,20). (b) (N_R,M_T,K) = (1,4,10). (c) (N_R,M_T,K) = (1,4,20). 63

Figure 4.7: Average convergence of GA vs. number of generations at an SNR of 10 dB, with (N_R,M_T,K) = (1,4,20) and two sets of values for β_1 and β_2..... 65

Figure 4.8: Ranges of (β_1,β_2) for which the number of generations for the GA to converge to within 0.1% of the optimum is within 5% of the minimum convergence time. 65

Figure 4.9: Comparison of convergence of GA for (N_R,M_T,K) = (2,2,20) and (1,4,10) when changing $\{\beta_1,\beta_2\}$ at different SNRs. (a) SNR = 10 dB. (b) SNR = 0 dB. 70

Figure 4.10: Average convergence of GA scheduling algorithm with the proportional fairness criterion, an SNR of 10 dB, and various values for β_1 and β_2. 71

Figure 4.11: Comparison of GA convergence with 1-point crossover (1X) and uniform crossover (UX) operators, each with two sets of β parameter values. (a) (N_R,M_T,K) = (1,2,20). (b) (N_R,M_T,K) = (2,2,20). 73

Figure 4.12: Comparison of GA convergence with 1-point crossover (1X) and uniform crossover (UX) operators, each with two sets of β parameter values. (a) (N_R,M_T,K) = (1,4,10). (b) (N_R,M_T,K) = (1,4,20). 73

Figure 5.1: Performance vs. K of exhaustive search, greedy, genetic, and SCAHE scheduling algorithms for BD; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. (a) SNR = 5 dB, and (b) 10 dB. 98

Figure 5.2: Performance vs. K of exhaustive search, greedy, genetic, and SCAHE scheduling algorithms for BD; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) SNR = 5 dB, (b) 10 dB, and (c) 15 dB. 99

Figure 5.3: Performance vs. K of exhaustive search, greedy, and genetic scheduling algorithms for SZF; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. (a) SNR = 5 dB, and (b) 10 dB. 101

Figure 5.4: Performance vs. K of exhaustive search, greedy, and genetic scheduling algorithms for SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) SNR = 5 dB, (b) 10 dB, and (c) 15 dB. 102

Figure 5.5: Average proportion of time that a given number of users out of a maximum of K_0 is scheduled using various algorithms for $M_T = 8$, $N_k = N = 2$, and $K_0 = 4$, with $K = 10$, 40, and 100. (a) BD, SNR = 5 dB. (b) BD, SNR = 10 dB. (c) SZF, SNR = 5 dB. (d) SZF, SNR = 10 dB. 104

Figure 5.6: Performance of exhaustive search, greedy, and genetic scheduling algorithms for SZF vs. log(log K); $M_T = 8$, $N_k = N = 2$, $K_0 = 4$; SNR = 5 dB and 10 dB. 104

Figure 5.7: Improvement in average sum rate vs. K of hybrid algorithm 1 over the unseeded GA while seeding the top n users into c chromosomes of the GA population. $M_T = 8$, $N_k = N = 2$, $K_0 = 4$; SNR = 10 dB. 106

Figure 5.8: Performance of hybrid algorithm 1 vs. K for BD and SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) BD, SNR = 5 dB; (b) SZF, SNR = 5 dB; (c) BD, SNR = 10 dB; and (d) SZF, SNR = 10 dB. 107

Figure 5.9: Performance of hybrid algorithm 2 vs. K for BD while letting the GA within HA2 run for N_g generations; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$, SNR = 10 dB. 108

Figure 5.10: Performance of hybrid algorithm 2 vs. K for BD and SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) BD, SNR = 10 dB; (b) SZF, SNR = 10 dB; (c) BD, SNR = 20 dB; and (d) SZF, SNR = 20 dB. 109

Figure 5.11: Maximum average sum rate vs. SNR for BD and SZF using an exhaustive search; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$, $K = 16$. 111

Figure 6.1: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD; $M_T = 4$, $K = K_0 = 2$, $N = 2$. 122

Figure 6.2: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD; $M_T = 6$, $K = K_0 = 3$, $N = 2$. 122

Figure 6.3: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD, using exhaustive search scheduling; $M_T = 8$, $K_0 = 4$, $N = 2$, $K = 16$..... 123

Figure 6.4: Distribution of improvement in average sum rate at 20 dB when initializing CGP algorithm with $\mathbf{Q}_{k,o}$; $M_T = 6$, $K = K_0 = 3$, $N = 2$. 125

Figure 6.5: Average weighted sum rate vs. SNR with proposed SZF covariance optimization method and with BD; $M_T = 8$, $K = K_0 = 4$, $N = 2$, $w_k = k/10$. 126

Figure 6.6: Proportion of time that user index is encoded in given position to maximize SZF weighted sum rate, where user weights equal 1/10 of user indices ($w_k = k/10$); $M_T = 8$, $K = K_0 = 4$, $N = 2$. Index "0" indicates no user encoded in that position. (a) User index is encoded first. (b) User index encoded second. (c) User index encoded third. (d) User index encoded fourth..... 127

Figure 6.7: Average sum rate vs. K comparing original and proposed CGP covariance optimization methods; $M_T = 8$, $N = 2$, $K_0 = 4$. (a) SNR = 10 dB, and (b) 20 dB..... 128

Figure 6.8: Distributions of convergence of CGP algorithm such that SZF sum rate changes by less than 1×10^{-3} / 5×10^{-8} bit/s/Hz; $M_T = 8$, $N = 2$, $K_0 = 4$. (a) SNR = 10 dB, and (b) 20 dB. 130

Figure 6.9: Average GA sum rate vs. K comparing CGP and DL fitness for scheduling, followed by CGP sum rate maximization with RR and DL initialization; $M_T = 8$, $N = 2$, $K_0 = 4$. Performance without CGP algorithm also shown. (a) SNR = 10 dB, and (b) 20 dB. 131

Figure A.1: Comparison of the distribution of values of a random Gaussian variable generated by Matlab (solid lines) with the theoretical Gaussian distribution (dashed lines). (a) PDF, (b) CDF. 151

Figure A.2: Comparison of the distribution of values of a random Rayleigh variable generated by Matlab (solid lines) with the theoretical Rayleigh distribution (dashed lines). (a) PDF, (b) CDF. 151

Figure A.3: Comparison of average sum rate and 95% confidence intervals vs. K for genetic scheduling algorithm for BD with varying number of Monte Carlo simulation runs N_{samp}; $M_T = 8$, $N = 2$, $K_0 = 4$, SNR = 10 dB. 154

Figure E.1: Performance vs. K of exhaustive search, greedy (GrA), genetic (GA), and SCAHE [69] scheduling algorithms for BD and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. 168

Figure E.2: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for BD and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. 169

Figure E.3: Performance vs. K of exhaustive search, greedy (GrA), genetic (GA), and SCAHE [69] scheduling algorithms for BD and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. 170

Figure E.4: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for BD and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. 171

Figure E.5: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for SZF and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. 172

Figure E.6: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for SZF and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. 173

Figure G.1: Performance vs. K of greedy (GrA) and genetic (GA) scheduling algorithms for SZF, using proposed CGP (with "round-robin" initialization) and original methods to obtain covariance matrices. Performance of GrA and GA for BD also shown. $M_T = 8$, $N = 2$, $K_0 = 4$, various SNRs. 177

List of Abbreviations

1X	One-point crossover
3G	Third generation
3GPP	Third Generation Partnership Project
4G	Fourth generation
AWGN	Additive white Gaussian noise
BC	Broadcast channel
BD	Block diagonalization
bit/s/Hz	Bits per second per Hertz
bits/c.u.	Bits per channel use
CDF	Cumulative distribution function
CGP	Conjugate gradient projection
CoMP	Coordinated multipoint
CSI	Channel state information
dB	Decibel(s)
DFT	Discrete Fourier transform
DL	Dabbagh/Love
DPC	Dirty paper coding
ES	Exhaustive search
FDM	Frequency division multiplexing
FFT	Fast Fourier transform
F-norm	Frobenius norm
GA	Genetic algorithm
Gbit/s	Gigabits per second
GrA	Greedy algorithm
GSO	Gram-Schmidt orthogonalization
HA1	Hybrid algorithm 1
HA2	Hybrid algorithm 2
HSPA	High Speed Packet Access
IEEE	Institute of Electrical and Electronics Engineers
i.i.d.	Independent and identically distributed
ISI	Inter-symbol interference
LTE	Long Term Evolution
MAC	Multiple access channel
Mbit/s	Megabits per second
MC	Multi-carrier
MHz	Megahertz
MIMO	Multiple-input, multiple-output

MISO	Multiple-input, single-output
M-LWDF	Modified largest weighted delay first
MT	Maximum throughput
MUI	Multiuser interference
Nu-SVD	Null-space-directed singular value decomposition
OFDM	Orthogonal frequency division multiplexing
OSDM	Orthogonal space division multiplexing
PDF	Probability distribution function
PF	Proportional fairness / proportionally fair
QoS	Quality of service
RR	Round-robin
SC	Single-carrier
SCAHE	Shen/Chen/Andrews/Heath/Evans
SESAM	Successive encoding and successive allocation method
SINR	Signal-to-interference-plus-noise ratio
SISO	Single-input, single-output
SNR	Signal-to-noise ratio
SVD	Singular value decomposition
SZF	Successive zero-forcing
TDMA	Time division multiple access
THP	Tomlinson-Harashima precoding
Tx	Transmit(ter)
UX	Uniform crossover
VoIP	Voice over Internet Protocol
WiMAX	Worldwide Interoperability for Microwave Access
WSR	Weighted sum rate
ZF	Zero-forcing
ZFB	Zero-forcing beamforming

Mathematical Notation

x, X	Scalar (italics)
$\mathbf{x}$	Vector (lowercase boldface)
$\mathbf{X}$	Matrix (uppercase boldface)
$x \approx y$	x is approximately equal to y
$x \triangleq y$	x is defined as y
$\max\{a,b\}$	Maximum of a and b
$\min\{a,b\}$	Minimum of a and b
$(x)^+$	Maximum of x and 0
$\lceil x \rceil$	Ceiling function; nearest integer greater than or equal to x
$E_{\mathbf{X}}\{\cdot\}$	Expected value / expectation over $\mathbf{X}$ (subscript may be omitted)
$\mathbf{I}_N$	($N\times N$) identity matrix (subscript may be omitted in general case)
$\mathbf{0}_{M\times N}$	An all-zero ($M\times N$) matrix (subscript may be omitted in general case)
X^*	Best value for scalar X
$\mathbf{X}^*$	Conjugate of matrix $\mathbf{X}$ (note also smaller * than above)
$\mathbf{X}^T$	Transpose of matrix $\mathbf{X}$
$\mathbf{X}^H$	Hermitian (conjugate) transpose of matrix $\mathbf{X}$
$\mathbf{X}^{-1}$	Inverse of square matrix $\mathbf{X}$
$\mathbf{X}^\dagger$	Moore-Penrose pseudoinverse of matrix $\mathbf{X}$
$\mathbf{X}^{1/2}$, $\mathbf{X}^{-1/2}$	Square root and inverse square root of $\mathbf{X}$
$\lvert\mathbf{X}\rvert$	Determinant of matrix $\mathbf{X}$
$Tr(\mathbf{X})$	Trace of matrix $\mathbf{X}$
$\text{rank}(\mathbf{X})$	Rank of matrix $\mathbf{X}$
$\lVert\mathbf{X}\rVert_F$	Frobenius norm of $\mathbf{X}$
$\mathbf{X} \succeq 0$	$\mathbf{X}$ is positive semidefinite
$blkdiag(\mathbf{X}_1,\cdots)$	A block-diagonal matrix formed from the matrices $\mathbf{X}_1$, …
$\mathcal{U},\mathbb{U}$	A set of numbers
$\mathbb{C}$	Set of complex numbers
$\lvert\mathbb{U}\rvert$	Cardinality of set $\mathbb{U}$
$\varnothing$	Empty set
$\mathcal{O}(\bullet)$	"Big-O" ("order of") notation

List of Notable Symbols

$\mathbf{A}_j$	Interference experienced by user j on broadcast channel, when used in context of MAC to BC transformations
$\mathbf{B}_j$	Interference experienced by user j on multiple access channel, when used in context of MAC to BC transformations, or a precoder input covariance matrix, when used in context of linear beamforming
C	MIMO channel capacity
C^*	Best chromosome in a population
C_E	Ergodic MIMO channel capacity
G	Utility function value
G_i	Utility function value (fitness) of chromosome i
G_{MT}	Utility function for maximum throughput criterion
G_{PF}	(Causal) utility function for proportional fairness criterion
$\mathbf{G}_i$	An effective channel matrix for user i, when used in context of MAC waterfilling, or a (proportional) gradient for user i, when used in context of conjugate gradient projection
$\bar{\mathbf{G}}_k, \hat{\mathbf{G}}_k$	Normalized and projected gradients for user k, respectively
$\mathbf{H}$	Aggregate MIMO channel matrix of users in multiuser system (or matrix for the one user in a single-user system)
$\mathbf{H}_{eff}$, $\mathbf{H}_e$, $\mathbf{H}_{k,e}$	Effective channel matrix (for user k)
$\mathbf{H}_k$, $\mathbf{H}_{jk}$	Channel matrix for user k (on subcarrier j, for a multi-carrier system)
$\mathbf{H}_k^B, \mathbf{H}_k^M$	Channel matrix for user k, specifically denoting for the BC and MAC, respectively
$\tilde{\mathbf{H}}_k, \bar{\mathbf{H}}_k$	Aggregate channel matrices for BD and SZF, respectively, used to find null space basis vectors / precoding or beamforming matrix for user k
K	Number of active users
K_0	Maximum number of simultaneously scheduled users (hard limit)
M_T	Number of transmit antennas
N	Number of receive antennas (per user); see also N_R
N_C	Number of (sub)carriers in a multi-carrier system
$N_{combinations}$	Number of (unordered) combinations of scheduled users
N_g	Number of generations (iterations) that genetic algorithm runs
$N_{ordered\_selections}$	Number of ordered selections of scheduled users
N_p	Population size (number of chromosomes) for genetic algorithm
N_R	Number of receive antennas (per user); see also N
N_S	Maximum number of simultaneously scheduled users (soft limit)
N_{samp}	Number of samples / Monte Carlo simulation iterations
$\mathbf{n}$, $\mathbf{n}_k$, $\mathbf{n}_{jk}$	Noise vector (for user k) (on subcarrier j)
p_c	Probability of crossover operation in genetic algorithm

p_i	Waterfilling power allocation, for eigenmode i
p_m	Probability of mutation in genetic algorithm
p_{sel_i}	Probability of chromosome i being selected for breeding in genetic algorithm
P	Transmit power constraint
$\mathbf{P}_k$	Transmit covariance matrix for user k, usually in the context of a MAC
$\mathcal{P}_i^{\perp}$, $\mathcal{P}_k^{\perp}$	Projector matrix for BD greedy algorithm, for iteration i / user k
Q_k	Queue length for user k
$\mathbf{Q}_k$	Transmit covariance matrix, usually in the context of linear beamforming (exception: when referring to a QR decomposition, $\mathbf{Q}$ is a unitary matrix)
$\mathbf{Q}_{k,o}$	Transmit covariance matrix for SZF, found by method in [50]
$\tilde{r}_k$	Rank of aggregate channel matrix $\tilde{\mathbf{H}}_k$
R_{BD}	Sum rate for BD
R_k, R_{jk}	Rate for user k (on subcarrier j, for a multi-carrier system)
R_k^B, R_k^M	Rate for user k, specifically denoting for the BC and MAC, respectively
$\overline{R}_k$	Average rate for user k
R_{MAC}	Sum rate for the MAC
R_{SZF}, R_{WSZF}	Sum rate and weighted sum rate for SZF, respectively
s_{jk}	Binary variable; equals 1 if user k scheduled on subcarrier j, 0 otherwise
S_E	Standard error of the mean for Monte Carlo simulations
$\mathbf{s}$, $\mathbf{s}_k$	Data symbol vector (intended for user k)
$\mathbf{S}_k$	A covariance matrix for user k during iterative MAC waterfilling, or a search direction for user k during conjugate gradient projection
$\mathbb{S}$	A set of scheduled users
t_c	A time constant for exponential filter when calculating average rate $\overline{R}_k$
$\mathbf{T}_k$	Transmit filter matrix for user k, when used in the context of conjugate gradient projection
$\hat{\mathbf{T}}_k$	A transmit filter (before normalization) for user k
u_i	User selected during iteration i of greedy scheduling algorithms
U, U_k	Utility value (for user k)
U_{PF}	(Non-causal) utility function for proportional fairness criterion
$\mathcal{U}$	Set of active users for greedy scheduling algorithms
$\mathbb{U}_i$	Subset of active users during iteration i of greedy scheduling algorithms
$\mathcal{U}_s$	Set of scheduled users for greedy scheduling algorithms
$\overline{v}_k$	Rank of the null space of $\overline{\mathbf{H}}_k$
$\mathbf{V}_i$, $\hat{\mathbf{V}}_k$	Row basis for BD greedy scheduling algorithm, at iteration i or for user k, respectively
$\overline{\mathbf{V}}_i^1$	Column basis for SZF greedy scheduling algorithm at iteration i
$\tilde{\mathbf{V}}_k^0$, $\overline{\mathbf{V}}_k^0$	Null space basis for $\tilde{\mathbf{H}}_k$ and $\overline{\mathbf{H}}_k$, for BD and SZF, respectively

w	The weight (number of '1's) in the head of a chromosome
w_k	A weight for the rate of user k, in a weighted sum rate
W_C	Bandwidth of (sub)carriers in a multi-carrier system
W_k	Head-of-line packet delay for user k
W_T	Total useful bandwidth in a single- or multi-carrier system
$\mathbf{W}$	Aggregate precoding / beamforming matrix of users
$\mathbf{W}_k$	Precoding / beamforming matrix for user k
$\mathbf{x}$, $\mathbf{x}_k$, $\mathbf{x}_{jk}$	Transmitted data signal (intended for user k) (on subcarrier j)
$\mathbf{y}$, $\mathbf{y}_k$, $\mathbf{y}_{jk}$	Received data signal (by user k) (on subcarrier j)
α	Step size, when used in context of conjugate gradient projection
β_1, β_2	Tunable parameters in adaptive mutation rate for genetic algorithm
η	OFDM bandwidth efficiency
λ_j	Eigenvalues or Lagrange multipliers, depending on context
μ, μ_{jk}	Lagrange multipliers
μ_G	Mean of the fitness of a population of GA chromosomes
ξ	Correlation threshold for greedy scheduling algorithms
π, $\pi(k)$	A permutation of user indices, and the index at position k of the permutation
ρ	The signal-to-noise ratio in the context of ZF beamforming, or a Frobenius norm in the context of conjugate gradient projection
$\hat{\sigma}$	Standard deviation of results obtained from Monte Carlo simulations
σ_G	Standard deviation of the fitness of a population of GA chromosomes
σ_n^2	Variance of additive white Gaussian noise
$\mathbf{\Sigma}_k$, $\mathbf{\Sigma}_{jk}$	Transmit covariance matrix for user k (on subcarrier j), usually in the context of DPC on a broadcast channel
$\mathbf{\Omega}_{jk}$	A Lagrange multiplier
∇_k	Gradient with respect to user k

Chapter 1

Introduction

1.1 Motivation

Over the past two decades, there has been an ever-increasing demand for mobile communication services. Initially, mobile services were limited to voice traffic. However, with the growth of the Internet, mobile users are now very much interested in data capabilities. Current third generation (3G) cellular systems have begun to satisfy some of this demand with techniques such as adaptive modulation and turbo coding. However, as the market penetration of smartphones such as Apple's iPhone and Research in Motion's Blackberry increases, the demand for high speed wireless data grows rapidly. Thus, the key focus of research in wireless networks is towards spectral efficiency and very high data rates.

Fourth generation (4G) systems are expected to provide a large improvement in data rates. The International Telecommunication Union has recommended in [1] and [2] that 4G systems should be able to provide 1 Gbit/s for low mobility users and 100 Mbit/s for high mobility users. The minimum requirements for peak spectral efficiency are 15 bit/s/Hz on the downlink and 6.75 bit/s/Hz on the uplink. Achieving those goals with current techniques is very difficult, and the radio channel itself is more problematical than wire or fibre links.

To fulfill this end, research over the past decade has focused on the spatial techniques that are possible with multiple antennas at the transmitters and receivers. Designs employing both multiple transmit antennas and multiple receive antennas are known as multiple-input, multiple-output (MIMO) systems. Due to the additional spatial resources available in MIMO systems, higher capacity and throughput are possible without the need for additional bandwidth or power. MIMO techniques have been proposed for and incorporated into current and future wireless communication standards, including IEEE 802.11n [3] (for wireless local area networks), 3GPP HSPA+ [4], 3GPP LTE [5] and LTE-Advanced [6], and IEEE 802.16e-2005 [7] and 802.16m (Advanced WiMAX) [8].

[9] contains a comprehensive overview of MIMO techniques in advanced cellular systems.

1.2 MIMO Wireless Communications

The key to the additional gains in performance in MIMO systems is the additional degrees of freedom that come with the multiple antennas. These are additional resources that can be exploited beyond the frequency and time dimensions used in earlier single-antenna systems. Seminal work [10],[11] has demonstrated that by adding antennas to transmitters and receivers, under rich scattering conditions the capacity of a MIMO system can scale linearly with the minimum of M_T and N_R, where M_T is the number of antennas at the transmitter, and N_R is the number of antennas at the receiver.

One possible way of exploiting the degrees of freedom in a MIMO system is through the use of diversity. If the antennas are spaced far enough apart, the fading signal paths provided by each of those antennas can be considered to be independent. In such a case, the likelihood of all the paths being in a fade is much lower than the chance of a single path itself fading. Copies of the signal can be sent or received along each of these paths, creating a much more reliable data link overall.

Alternatively, the degrees of freedom can be used for spatial multiplexing. The multiple antennas can be used to form several transmission streams. Additional data can be sent along each of these streams, resulting in a much higher capacity for the link. Both techniques require an uncorrelated rich scattering environment to work. As one might expect, there is a tradeoff between the gains achievable through spatial diversity and spatial multiplexing [12].

1.3 Orthogonal Frequency Division Multiplexing

Although radio bandwidth is scarce, it is nonetheless still an important factor in improving the throughput of 4G wireless systems. Compared to bandwidths of a few MHz in current 3G systems, future designs are expected to use much larger frequency bands. LTE supports a scalable bandwidth of up to 20 MHz, and LTE-Advanced supports scaling the bandwidth up to 100 MHz [6].

However, with these larger bandwidths comes the problem of frequency-selective fading. The channel gain is not constant along the entire bandwidth; it instead varies across the frequency range. This leads to problems with inter-symbol interference (ISI). One way to combat ISI is to split the frequency band up into smaller sub-bands, so that

the fading across each of the sub-bands is approximately flat. Such frequency division multiplexing (FDM) techniques have been well known for several decades. Original designs separated the frequency bands entirely so they would not interfere. However, a more efficient technique was later found to allow the frequency bands to overlap. By spacing the subcarrier frequencies just sufficiently far apart that the signals maintain orthogonality, the frequency spectrum is used more efficiently while still in principle[1] avoiding interference between subcarriers. This multi-carrier system design is known as orthogonal frequency division multiplexing (OFDM). Such multi-carrier systems allow for additional degrees of freedom in dividing the system resources and thus enhancing its performance.

In practice, OFDM can be implemented using the discrete Fourier transform (DFT) [14]. The DFT can be realized inexpensively in hardware with high computational efficiency using the fast Fourier transform (FFT). OFDM has been incorporated into many wireless communication standards, including those referenced earlier at the end of Section 1.1. [13] contains an overview of issues and techniques for OFDM as applied to wireless communications, including MIMO-OFDM.

1.4 Multiuser MIMO Systems

A typical cellular system consists of numerous users in communication with their corresponding base stations. In a MIMO multiuser system, each base station would have multiple transmit / receive antennas. In general, each user may also have multiple transmit / receive antennas, although currently, practically all users are equipped with only a single antenna[2]. On the uplink, or multiple access channel (MAC), several users communicate simultaneously with their base station, while on the downlink, or broadcast channel (BC), the base station transmits data to several users simultaneously. These multiple overlapping signals result in multiuser interference (MUI). A significant challenge of multiuser MIMO systems is to somehow reduce or eliminate MUI. Since in general the users cannot cooperate, it is the role of the base station to deal with the MUI.

On the broadcast channel, it is known that the sum-capacity is achieved through a precoding process known as dirty paper coding (DPC) [15],[16],[17]. DPC achieves capacity by transmitting to multiple users and successively removing the effect of

[1]Implementation issues like an offset in the carrier frequencies or a time-dispersive multipath channel may result in interference between the subcarriers or the symbols [13]. However, such issues are outside the scope of this work.

[2] This case is often called a multiple-input, single-output (MISO) system.

interference on each user it encodes. At any given point in the encoding process, a user will not experience interference from the users encoded prior. Unfortunately, DPC is extremely complex and difficult to implement in practice. The process requires the base station having information about the downlink channel state of each user being encoded, having non-causal knowledge of each of the users' signals that will be sent, and determining optimal power covariance matrices for each user to reduce the remaining MUI. While certain methods approximate DPC, often lower complexity methods are desired.

Linear beamforming methods are thus of interest in the literature. Such methods are significantly less complex, though suboptimal, when compared to DPC. Beamforming processes the signal for each user independently by multiplying it by a certain beamforming weight vector(s) across the multiple transmit antennas. Proper design of the weight vector(s) can reduce or eliminate the MUI, but can also reduce the available degrees of freedom while doing so.

1.5 MIMO Multiuser Scheduling

With multiple users in the system, there exists another resource that can be harnessed, namely so-called multiuser diversity. Just as with multiple signal paths in a MIMO system meaning it is unlikely for all paths to be in a fade simultaneously, with multiple users, it is unlikely that all users will experience a fade simultaneously, provided that their channels are independent of each other. If the data that the users are receiving can tolerate some delay with no ill effect, the base station can exploit multiuser diversity to further increase capacity. A well-designed scheduler can transmit to users who presently have good channel conditions, and delay transmitting to other users until their channel conditions improve. This concept is already used in 3G cellular systems. It is well known that in a system with a single transmit antenna, it is optimal in terms of sum-capacity to devote all transmit power to the single user with the best channel [18]. However, in MIMO systems, this is no longer the case; as stated above, capacity is achieved with DPC while transmitting to multiple users simultaneously.

Even though multiple users can be supported simultaneously, there is still usually a relatively low upper limit to the number of users that the base station can transmit to at once. This limit is usually related to the number of transmit antennas M_T at the base station. The limit is very likely to be reached in a cellular system, where in general the number of users in communication with the base station is much greater than M_T. Using

the multiple antennas to help eliminate MUI in general also reduces the number of users that can be simultaneously supported. Hence, this further necessitates the need for a scheduling algorithm at the base station.

Since scheduling multiple users is required to achieve the sum-capacity in a MIMO system, MUI will arise, as already mentioned. Fortunately, the multiple antennas also provide assistance in dealing with the problem. The users will in general be separated in space from each other. The scheduling algorithm can make use of this by selecting users with a significant spatial separation of their channels. Signals directed towards these users would already experience little interference before complex coding is considered. Thus, the scheduling algorithm can also help reduce the burden of MUI removal between the users. On the whole, a MIMO scheduler thus should consider both channel gains and spatial separation in its scheduling decisions.

One further very important factor that the scheduler must incorporate is fairness. In general, the users' channels are heterogeneous; the users have different average channel gains, experience different levels of shadowing and multipath fading, etc. If the scheduler were to focus solely on system throughput, users in comparatively poor channel conditions would receive very little service. Thus, the scheduler must balance system throughput with fairness.

Lastly, as the demand for high data rates over wireless links increases, so too does the demand for various multimedia services. The traffic for certain types of these services may be particularly sensitive to the delay it experiences. Examples include streaming of media, videoconferencing, Voice over Internet Protocol (VoIP) communications, and online gaming. These services may have certain quality of service (QoS) parameters that must be met, such as a maximum delay or minimum throughput. Failing to meet these parameters could lead to a degradation of service quality and user dissatisfaction.

Thus, overall, the various factors involved in the design of MIMO systems and scheduling algorithms are quite complex. We discuss the technical aspects of multiuser MIMO systems and scheduling algorithms in more detail in Chapter 2.

1.6 Research Goals and Summary of Contributions

The primary goal of this research is to further investigate scheduling methods in MIMO systems. Scheduling of users in MIMO systems has been investigated to some extent, but generally not including issues related to fairness or QoS. This is particularly the case for DPC and certain other precoding schemes. In these schemes, the order in

which the users are encoded will affect the interference they experience and thus the data rates they can support. Thus, the scheduling algorithm should be aware of and capable of adjusting the encoding order of users.

The scheduling algorithms should also be fast and of low complexity. There is often a limited time in which to make a scheduling decision; transmission intervals are often kept short so as to avoid the channel changing significantly within that interval. In general, the scheduler is aiming to optimize some sort of utility function for the system that incorporates whatever parameters or constraints that are on the data traffic being carried. For example, the scheduler may want to maximize the system throughput, yet make sure that users are fairly served, and that delays are not too high. The optimal method of doing so is to exhaustively check all possible selections of users, but this would take far too long in practice. Thus, the scheduler must be of low complexity in making a decision towards the goal of maximizing the utility.

Thus, in summary, the first goal of this research is to investigate scheduling algorithms for DPC that are of low complexity, and are cognizant of both encoding order and QoS demands. We first wish to find said algorithms for DPC, since DPC achieves the capacity of a multiuser MIMO system. However, since DPC is quite complex, we also wish to investigate other precoding schemes. Thus, the second goal of this research is to extend the scheduling algorithms developed for DPC to more practical methods like linear beamforming.

The main contributions provided by the work in this dissertation are as follows.

- We have investigated the use of genetic algorithms for scheduling with DPC, an area which has received little prior research focus. Genetic algorithms (GAs) are a stochastic method of solving optimization problems, such as the maximization of scheduling algorithm utility functions. They are particularly known for their speed in finding very good solutions, and for being able to handle a wide variety of utility functions. We propose a GA scheduling approach under DPC precoding that accounts for the encoding order, and further extend that approach to allow scheduling of users independently on the carriers of an OFDM system. We investigate the performance of the GA for two cases: maximizing the system throughput, and maximizing the sum of the users' instantaneous rates relative to their average rates (also known as the proportional fairness criterion [19]). These two cases provide a good indication of how the GA would work with any more general utility function that can be expressed in terms of a weighted sum rate. We demonstrate that the GA

performance is near-optimal compared with an exhaustive search at a greatly reduced complexity. Furthermore, in the case of an OFDM system, an increase in spectral efficiency is shown relative to the single-carrier case.

- While the GA performs close to the exhaustive search, further improvements can still be obtained. We investigate the tuning of parameters of an adaptive mutation rate within the GA on the time it takes the GA to converge. We demonstrate that there is in fact a range of values for the parameters that lead to a near-minimum convergence time, and that it is important for the parameter values to be tuned to within that range. With tuning, the convergence time can decrease considerably, dropping the time to less than 30% of that required for untuned values in one case. A simple equation that is linear in the parameters is proposed to find their proper values for changing numbers of supported users and user pool sizes. We also investigate the effect of changing the crossover method in the GA on its convergence, but find that there is little gain to be found in doing so, especially compared to the parameter tuning.
- We adapt the GA scheduling algorithm to two cases of linear precoding, using block diagonalization (BD) and successive zero-forcing (SZF). We compare the performance of the GA and other "greedy" scheduling algorithms to an exhaustive search. A comprehensive analysis of the complexity of both the GA and the greedy algorithms is also conducted. We find that the greedy algorithms are less complex than the GA. The GA outperforms the greedy algorithms at lower user pool sizes and higher signal-to-noise ratios (SNRs), while the greedy algorithm is better with more users and lower SNRs. We further propose two hybrid scheduling algorithms incorporating and combining traits of the genetic and greedy algorithms. These hybrid algorithms perform better than the original algorithms they are based on, with no increase in the order of complexity.
- During our work with SZF, we identified a deficiency at higher SNRs in the previously published method used to allocate power to users. The resulting throughput is considerably lower than what it theoretically should be. We propose a new method to optimize the power allocated to users based on conjugate gradient projection that significantly increases the resulting throughput of the system. Furthermore, the existing method is only designed to attempt to maximize the system sum rate, and thus is not meant to handle issues of fairness or quality of service. Our proposed method also allows for the maximization of a weighted sum rate with SZF, where the weights may incorporate the relevant QoS constraints. We demonstrate

that our proposed method significantly outperforms the existing method at medium to high SNRs, regardless of the specific scheduling algorithm used, whether it is an optimal exhaustive search, or the genetic and greedy algorithms mentioned above that we proposed and investigated.

1.7 Organization of the Thesis

Chapter 2 begins by examining the background and details of MIMO aspects more closely. We discuss single-user and multiuser MIMO systems, including details about linear and non-linear methods to handle multiuser interference. We also give an overview of existing methods of incorporating fairness and QoS into scheduling utility functions.

Chapter 3 covers the use of genetic algorithms for scheduling in single- and multi-carrier dirty-paper-coded systems. We describe genetic algorithms in general, how they operate, and how they can be adapted for scheduling. The system model used for the investigation is described, and the results of Monte Carlo simulations are presented. We discuss the performance of the GA, including results on throughput, delay and convergence. A brief comparison of the runtime versus an exhaustive search is also provided.

Chapter 4 describes the effects of tuning parameters within the GA. We provide simulation results evaluating how the tuning affects the convergence of the GA, and propose a simple linear equation to find suitable parameter values. We also compare the performance of the GA when using two different crossover operators.

In Chapter 5, we examine the performance of genetic, greedy, and hybrid scheduling algorithms with BD and SZF. BD and SZF are described in detail. A complexity analysis of the investigated algorithms is provided, followed by simulation results covering their performance for a variety of user pool sizes and SNRs.

Chapter 6 describes our proposed improved method for SZF transmit covariance optimization. We describe the proposed algorithm itself and provide simulation results for a few simple cases. We investigate how the algorithm performs when maximizing a weighted sum rate. We then re-examine the performance of the genetic and greedy scheduling algorithms for SZF using the new covariance algorithm.

Finally, in Chapter 7, we provide overall conclusions for this work, and provide suggestions for possible areas of future work.

Chapter 2

Background on MIMO, Precoding, and Scheduling

2.1 Single-User MIMO Systems

A single-user point-to-point MIMO system is depicted in Figure 2.1. The transmitter has M_T transmit antennas, while the receiver has N_R receive antennas. The complex received signal vector $\mathbf{y} = \left[y_1, y_2, \ldots, y_{N_R} \right]^T \in \mathbb{C}^{N_R \times 1}$ can be written as:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{2.1}$$

where $\mathbf{x} = \left[x_1, x_2, \ldots, x_{M_T} \right]^T \in \mathbb{C}^{M_T \times 1}$ is the transmitted signal vector, $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$ is the noise vector, and $\mathbf{H} \in \mathbb{C}^{N_R \times M_T}$ is the channel matrix. Each element h_{ij}, $1 \le i \le N_R$, $1 \le j \le M_T$, represents the complex channel gain between the ith receive antenna and the jth transmit antenna. The entries of the noise vector $\mathbf{n}$ are independent and identically distributed (i.i.d.), circularly symmetric, complex Gaussian random variables with zero mean and variance σ_n^2 (or equivalently $\sigma_n^2/2$ per complex dimension), such that $E\left\{\mathbf{n}\mathbf{n}^H\right\} = \sigma_n^2 \mathbf{I}_{N_R}$. Usually, the transmitter has a constraint P on the maximum power available, requiring $Tr(\mathbf{\Sigma}) \le P$, where $\mathbf{\Sigma} = E\left\{\mathbf{x}\mathbf{x}^H\right\}$.

The capacity of the channel is achieved when the transmitted signal $\mathbf{x}$ is Gaussian

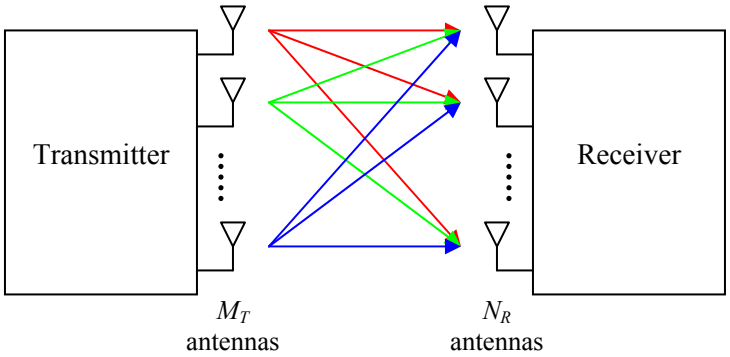

Figure 2.1: Block diagram of a single-user MIMO system.

distributed. If complete channel state information (CSI) is available at both the transmitter and the receiver, then the MIMO channel capacity is given by [10]:

$$C = \sum_{i=1}^{r} \left(\log_2 \left(\frac{\mu \lambda_i}{\sigma_n^2} \right) \right)^+ = \log_2 \prod_{i=1}^{r} \left(1 + \frac{p_i \lambda_i}{\sigma_n^2} \right). \tag{2.2}$$

In (2.2), the λ_i terms are the eigenvalues of the matrix product $\mathbf{H}\mathbf{H}^H$ (or equivalently $\mathbf{H}^H\mathbf{H}$), while r is the rank of $\mathbf{H}$; the λ_i terms are also equal to the squared singular values of $\mathbf{H}$. The values of p_i and μ are chosen according to the waterfilling power allocation over the eigenmodes of $\mathbf{H}$ with the power constraint P, such that $p_i = \left(\mu - \sigma_n^2 / \lambda_i \right)^+$ and $\sum_{i=1}^{r} p_i = P$. The operator $(x)^+$ denotes $\max\{x,0\}$.

In the event that channel knowledge is available only at the receiver, but not at the transmitter, the best capacity the transmitter can achieve is by allocating equal power to each of the transmit antennas. In this case, the capacity becomes [11]:

$$C = \log_2 \prod_{i=1}^{r} \left(1 + \frac{P \lambda_i}{M_T \sigma_n^2} \right). \tag{2.3}$$

Using the identities $|\mathbf{I} + \mathbf{AB}| = |\mathbf{I} + \mathbf{BA}|$ and $\left|\mathbf{I} + \mathbf{M}\right| = \prod_m \left(1 + \lambda_m\right)$, this capacity can be rewritten as:

$$C = \log_2 \left| \mathbf{I}_{N_R} + \frac{P}{M_T \sigma_n^2} \mathbf{H}\mathbf{H}^H \right| = \log_2 \left| \mathbf{I}_{M_T} + \frac{P}{M_T \sigma_n^2} \mathbf{H}^H\mathbf{H} \right|. \tag{2.4}$$

In a rich scattering environment, $\mathbf{H}$ will be full rank, so $r = \min\{M_T,N_R\}$. In such a case, it can be shown that as the signal-to-noise ratio (SNR) grows large, the capacity will scale linearly with $\min\{M_T,N_R\}$ [10],[11]. One such environment is the spatially uncorrelated Rayleigh fading channel, where the entries of $\mathbf{H}$ are modeled as i.i.d. circularly symmetric complex Gaussian variables with zero mean and unit variance (or variance 0.5 per dimension). Under this model, the absolute values of the entries of $\mathbf{H}$ follow a Rayleigh distribution [20]. Since $\mathbf{H}$ is random, one is often interested in the *ergodic* capacity of the channel, which is the expected value of the capacity with respect to the channel matrix, i.e., $C_E = E_{\mathbf{H}}\{C\}$.

2.2 Multiuser MIMO Systems

In a multiuser MIMO system, we are generally concerned with one of two possible models wherein K active users are attempting to simultaneously communicate with a base station. It is furthermore assumed that these users are unable to cooperate in their

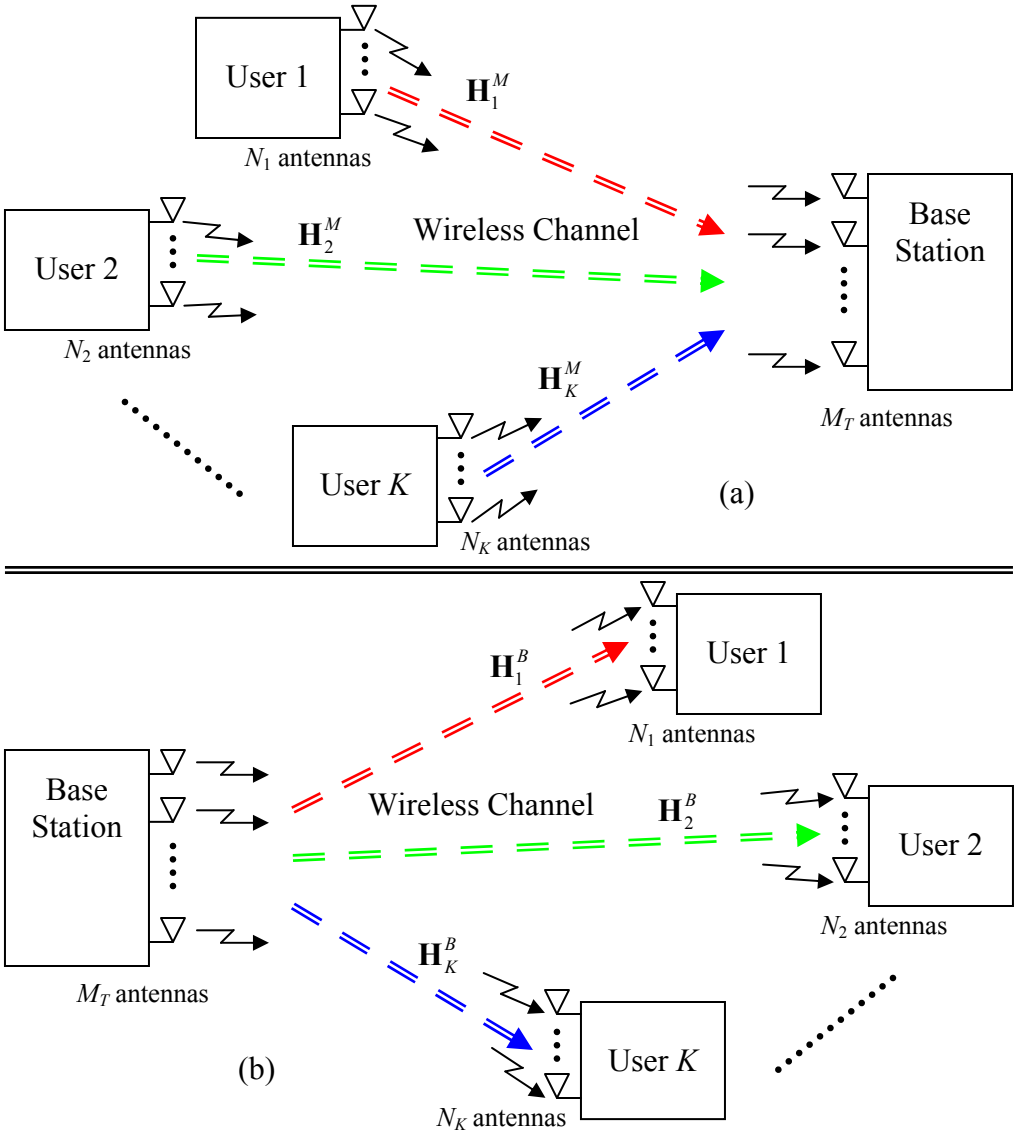

Figure 2.2: (a) Block diagram of MIMO multiple access channel. (b) Block diagram of MIMO broadcast channel.

communications. (If they were, the situation would reduce to an equivalent single-user MIMO system as described above.) The two models in essence describe a typical cellular scenario, and differ in the direction of communication. On the multiple access channel (MAC), also called the uplink, several users transmit data simultaneously to a base station. On the broadcast channel (BC), also known as the downlink, the base station transmits data to several users. Not all of the data is necessarily intended for all of the users on the downlink. In both cases, the multiple transmissions can impede each other, leading to multiuser interference (MUI). Often, the number of users K may be greater

than the base station can support simultaneously. In such a case, scheduling of users is required, as shall be discussed further in Section 2.4.

2.2.1 MAC, BC, DPC, and Duality

On the MAC, let $\mathbf{H}_k^M \in \mathbb{C}^{M_T \times N_k}$ be the channel matrix between the *k*th user and the base station, for all $k = 1,2,\ldots,K$. The received signal vector $\mathbf{y} \in \mathbb{C}^{M_T \times 1}$ at the base station will be:

$$\mathbf{y} = \sum_{k=1}^{K} \mathbf{H}_k^M \mathbf{x}_k + \mathbf{n} . \tag{2.5}$$

$\mathbf{x}_k \in \mathbb{C}^{N_k \times 1}$ is the signal vector transmitted by user *k*, and $\mathbf{n} \in \mathbb{C}^{M_T \times 1}$ is a complex additive white Gaussian noise (AWGN) vector[1]. The base station must separate the signals of the various users. It is known that the capacity of the Gaussian MAC is achieved when the base station employs successive interference cancellation to decode the users' signals [10],[21],[22]. That is, after decoding the signal for a given user, the transmitted signal for that user is then recreated and subtracted from the combined received signal to remove the interference of that signal on the signals from the other users. The order in which the users are decoded will thus affect the rates they receive. Let $\{\pi(1),\pi(2),\ldots,\pi(K)\}$ denote the order of the decoding, where $\pi(1)$ is the user decoded *last*, and $\mathbf{P}_k = E\left\{\mathbf{x}_k \mathbf{x}_k^H\right\} \in \mathbb{C}^{N_k \times N_k}$ denote the covariance of the transmitted signal of user *k*. Then, the achievable rate for each user is:

$$R_{\pi(k)}^M = \log_2 \frac{\left|\mathbf{I} + \sum_{i=1}^{k} \mathbf{H}_{\pi(i)}^M \mathbf{P}_{\pi(i)} \left(\mathbf{H}_{\pi(i)}^M\right)^H\right|}{\left|\mathbf{I} + \sum_{i=1}^{k-1} \mathbf{H}_{\pi(i)}^M \mathbf{P}_{\pi(i)} \left(\mathbf{H}_{\pi(i)}^M\right)^H\right|}, \tag{2.6}$$

and the achievable MAC sum rate is:

$$R_{MAC} = \sum_{k=1}^{K} R_{\pi(k)}^M = \log_2 \left|\mathbf{I} + \sum_{i=1}^{K} \mathbf{H}_{\pi(i)}^M \mathbf{P}_{\pi(i)} \left(\mathbf{H}_{\pi(i)}^M\right)^H\right| . \tag{2.7}$$

On the BC, let $\mathbf{H}_k^B \in \mathbb{C}^{N_k \times M_T}$ be the channel matrix between the base station and the *k*th user, for all $k = 1,2,\ldots,K$. The received signal vector $\mathbf{y}_k \in \mathbb{C}^{N_k \times 1}$ at each user *k* will be:

$$\mathbf{y}_k = \mathbf{H}_k^B \sum_{j=1}^{K} \mathbf{x}_j + \mathbf{n}_k . \tag{2.8}$$

[1] In the literature, σ_n^2 is usually assumed without loss of generality to be equal to 1. We also make this assumption throughout this thesis unless a specific value is stated for σ_n^2.

$\mathbf{x}_k \in \mathbb{C}^{M_T \times 1}$ is the signal vector transmitted by the base station intended for user *k*, and $\mathbf{n}_k \in \mathbb{C}^{N_k \times 1}$ is the complex AWGN vector experienced by user *k*. Since the users cannot cooperate in decoding their signals, it is the responsibility of the base station to help eliminate interference between the users, by precoding or preprocessing the data before it is transmitted.

In a system with perfect channel knowledge and a source of interference known non-causally at the transmitter (but not necessarily at the receiver), the transmitter can employ a technique known as writing on dirty paper. It is shown in [15] that the transmitted signal can be encoded in such a way that the known interference can be presubtracted, in essence removing its effect from the transmitted signal, making the system capacity the same as if the interference was not there. [15] demonstrated this for Gaussian-distributed interference and noise sources, but this was later extended in [23] to show that only one of the interference or noise needs to be Gaussian; the other can be arbitrarily distributed.

On the BC, the signal intended for any given user is interference for any other user. However, the transmitter knows those signals in advance, and thus can encode them successively with dirty paper coding (DPC) [16],[17] to remove the interference from any user *j* on the signal for *k*, where $j < k$. Similar to the MAC, the encoding order will thus affect the rates each user receives. With the encoding order $\{\pi(1),\pi(2),\ldots,\pi(K)\}$, where $\pi(1)$ is encoded *first*, then an achievable set of rates is given by [24]:

$$R_{\pi(k)}^{B} = \log_2 \frac{\left| \mathbf{I} + \mathbf{H}_{\pi(k)}^{B} \left(\sum_{j \geq k} \mathbf{\Sigma}_{\pi(j)} \right) \left(\mathbf{H}_{\pi(k)}^{B} \right)^H \right|}{\left| \mathbf{I} + \mathbf{H}_{\pi(k)}^{B} \left(\sum_{j > k} \mathbf{\Sigma}_{\pi(j)} \right) \left(\mathbf{H}_{\pi(k)}^{B} \right)^H \right|}. \tag{2.9}$$

$\mathbf{\Sigma}_k = E\left\{ \mathbf{x}_k \mathbf{x}_k^H \right\} \in \mathbb{C}^{M_T \times M_T}$ is the covariance matrix for the signal intended for user *k*. The achievable dirty paper region is then defined as the set of all possible rates from (2.9) for all covariance matrices $\mathbf{\Sigma}_k$ subject to the power constraint $\sum_{\forall k} Tr\left(\mathbf{\Sigma}_k\right) \leq P$ and over all possible encoding orders.

Much effort has gone into characterizing the DPC rate region. It has been shown in [16],[17],[24],[25],[26] that DPC is optimal and achieves the capacity of the MIMO BC, and that the capacity region of the BC is the same as for the MAC. In other words, the MIMO MAC is the dual to the MIMO BC. If the BC has a set of channel matrices $\mathbf{H}_k$, a power constraint $\sum_{\forall k} Tr\left(\mathbf{\Sigma}_k\right) \leq P$, and an encoding order π, where user $\pi(1)$ is encoded *last*, the same rates will be achieved on the dual MAC with channel matrices $\mathbf{H}_k^H$, a sum-

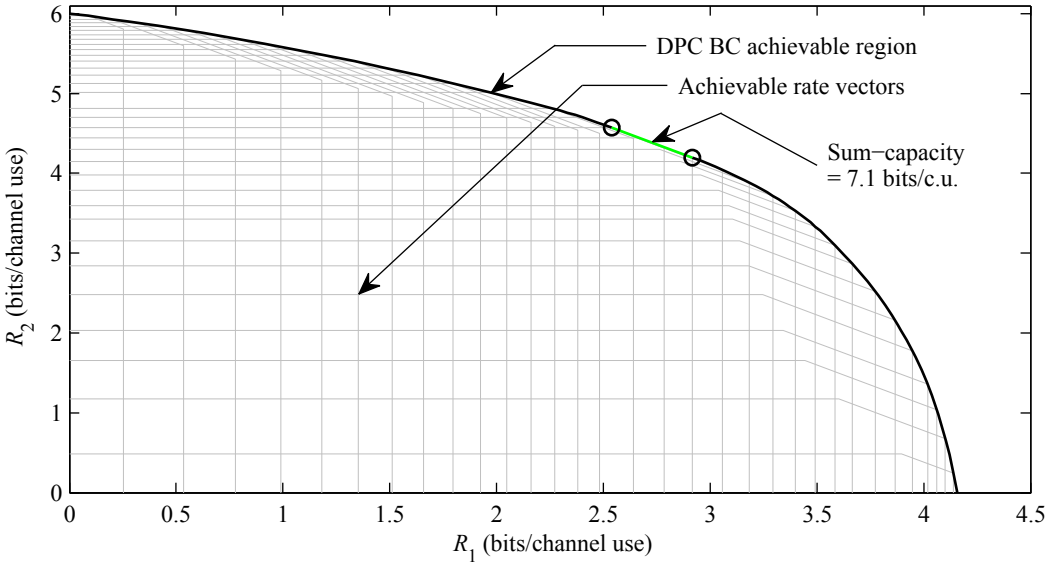

Figure 2.3: Typical dirty paper coding achievable rate region for a 2×2 MIMO broadcast channel.

power constraint on the users of $\sum_{\forall k} Tr\left(\mathbf{P}_k\right) \le P$, and the reverse decoding order π where user $\pi(1)$ is decoded *first*.

A typical DPC capacity region is shown in Figure 2.3. Two users with two antennas each receive data from a two-antenna base station. The SNR for each user is 10 dB, and the channel matrices for each user are:

$$\mathbf{H}_1 = \begin{bmatrix} -0.3 - 0.2i & 0.6 - 0.2i \\ -0.1 + 0.3i & 1.0 + 0.2i \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 0.4 + 0.1i & 0.3 - 0.6i \\ -0.6 + 1.7i & 0.6 - 0.7i \end{bmatrix}. \tag{2.10}$$

The sum-capacity is achieved along a line segment on the boundary of the achievable region. The two end points of the segment correspond to the two possible encoding orders of the users. Any point along the line segment can be achieved in the long term by time-sharing between the two encoding orders.

While dirty paper coding is optimal for the BC, it is also non-linear and highly complex. Much research effort has been dedicated to practical implementations of DPC. A number of these are based on extensions of Tomlinson-Harashima precoding (THP) [27],[28], which was originally designed to counteract inter-symbol interference, wherein a signal interferes with itself over time. Some applications and extensions of THP to the MIMO BC are described in [29],[30],[31]. THP has also been combined with trellis-shaping codes to help remove some shaping loss caused by a modulo operation in THP [32]. Other methods of implementing DPC include superposition coding [33],[34], vector

perturbation [35],[36], and lattice-based precoding [37],[38],[39]. All of these methods, however, remain non-linear and quite complex.

2.3 MIMO Precoding and Beamforming

Because of the complexity of DPC, other suboptimal methods are of interest to help reduce the interference between users. Linear beamforming methods are particularly desirable due to their comparatively low complexity. The signal for each user can be precoded or preprocessed separately by multiplying the data $\mathbf{s}_k \in \mathbb{C}^{N_k \times 1}$ for the user k by a beamforming matrix $\mathbf{W}_k \in \mathbb{C}^{M_T \times N_k}$. Then, the transmitted signal $\mathbf{x} = \sum_k \mathbf{W}_k \mathbf{s}_k$. The beamforming matrices can be designed to remove at the transmitter some or all of the interference between the users. However, doing so can also restrict the degrees of freedom that the system has to transmit streams of data to users. Nonetheless, it is known that as K becomes large, the sum rate of beamforming asymptotically approaches that of DPC [40],[41]. This is intuitively explained; as the number of active users to choose from becomes large, the likelihood of finding users with near-orthogonal channels increases. Thus, beamforming functions just as well as DPC if there is little interference between users to start with.

One linear method is channel inversion, sometimes called zero-forcing beamforming (ZFB) [17], which is designed for users with a single receive antenna[1]. Channel inversion creates a set of orthogonal, non-interfering channels. Let $\mathbf{H} = \left[\mathbf{H}_1^T, \mathbf{H}_2^T, \cdots, \mathbf{H}_K^T\right]^T$ be the vertical concatenation of K users' channels (where $K \le M_T$). Then, the signal transmitted by the base station is $\mathbf{x} = \mathbf{H}^\dagger \mathbf{s}$, where $\mathbf{s}$ is the vector of data symbols for the K users; the kth entry of $\mathbf{s}$ is the data symbol s_k for user k. $\mathbf{H}^\dagger$ is the Moore-Penrose pseudoinverse[2] of $\mathbf{H}$ [43], given by $\mathbf{H}^\dagger = \mathbf{H}^H \left(\mathbf{H}\mathbf{H}^H\right)^{-1}$. The beamforming matrix $\mathbf{W}_k$ for user k is the kth column of $\mathbf{H}^\dagger$. An alternative method is to use regularized channel inversion, also called minimum mean squared error precoding [35],[44]. In this case, $\mathbf{s}$ is processed by the matrix $\mathbf{H}^H \left(\mathbf{H}\mathbf{H}^H + \frac{K}{\rho}\mathbf{I}\right)^{-1}$, where ρ is the SNR. Both methods are somewhat sensitive to errors in the channel estimate, and can require a great deal of power to perform the channel inversion with badly-conditioned channels, which can lead to noise enhancement

[1] With single-antenna users, the data vector $\mathbf{s}_k$ for each user becomes a scalar s_k.
[2] If there are power constraints per antenna at the base station instead of a total power constraint, some other generalized matrix inverse may provide better performance instead [42].

at the receivers [45]. There is also an increasing gap in capacity compared to DPC as the SNR grows [45].

ZFB is a special case of a larger class of precoding schemes called orthogonal space-division multiplexing (OSDM). OSDM is meant to completely remove MUI at the transmitter. ZFB can be extended to users with multiple receive antennas by considering each antenna as a separate user and zero-forcing accordingly. However, such a scheme is suboptimal, as the multiple receive antennas at a user can coordinate with each other to process the received signal. MUI should instead be removed between users, but not between antennas of a user. Schemes such as those described in [46],[47],[48] accomplish this by forcing the transmitted signal for a user to lie in the null space of the channel of all other users. In other words, $\mathbf{H}_k\mathbf{W}_j = \mathbf{0}$ for all $k \neq j$. The most commonly used of these schemes is known as block diagonalization (BD), after the nomenclature in [46]. The iterative method of [48], called Nu-SVD, is more general, but also more complex at larger M_T and K.

Completely nulling the MUI is not necessarily the best solution; for example, capacity-achieving DPC only removes interference successively. By relaxing the zero-MUI condition, the performance of linear beamforming can be improved. This is part of the concept behind zero-forcing-DPC (ZF-DPC) [17],[49]. ZF-DPC, which is designed for single-antenna users, performs an LQ decomposition[1] of $\mathbf{H}$ (i.e., $\mathbf{H} = \mathbf{LQ}$), where $\mathbf{L}$ is a lower triangular matrix, and $\mathbf{Q}$ is unitary. Then, by choosing $\mathbf{W}_k$ as the kth column of $\mathbf{Q}^H$, any user i will not experience interference from any user j, for all $i < j$. This was extended to users with more than one antenna in [50], where the extension is called successive zero-forcing (SZF). This block successive zero-forcing was also proposed in [51], in relation to a scheme called successive encoding and successive allocation method (SESAM). Both ZF-DPC and SZF approach the capacity of DPC in the low-SNR regime, in the limit as the available transmit power P becomes vanishingly small [17],[50], provided that an optimal user ordering is used for the interference removal. We also note that some other methods of beamforming try to handle the MUI without outright removing some or all of it [52],[53].

In ZF-DPC and SZF, some MUI is removed through linear beamforming. It is furthermore possible to remove the remaining user interference by encoding the users' signals with DPC. Overall, interference on user k from users $j > k$ is removed by

[1] Equivalently, one could perform a QR decomposition on $\mathbf{H}^H$, where $\mathbf{R}$ is an upper triangular matrix. Then, $\mathbf{H} = (\tilde{\mathbf{Q}}\mathbf{R})^H = \mathbf{R}^H\tilde{\mathbf{Q}}^H = \mathbf{LQ}$.

beamforming, and interference from users $j < k$ is removed by DPC. In the case of SZF, this extension is called SZF-DPC[1] [50]. The effective overall channel therefore becomes diagonal for ZF-DPC and block-diagonal for SZF-DPC. Since the use of DPC is assumed in these two schemes, they are no longer strictly linear, and they are technically both more complex than DPC alone. The benefit of the two schemes is that the (block-) diagonal effective channel makes power allocation to the users much simpler than for DPC or SZF [50]. Both ZF-DPC and SZF-DPC approach the capacity of DPC in the high-SNR regime, in the limit as the transmit power P becomes infinite [50], for any arbitrary ordering of users.

We lastly note that the above described methods of beamforming are mostly designed to maximize the system sum rate under a power constraint. There are also alternative goals in linear beamforming, such as minimizing power allocation subject to minimum signal-to-interference-plus-noise ratio (SINR) constraints [54],[55], maximizing the minimum user SINR or rate subject to a power constraint [54],[56],[57], and minimizing the mean squared error [58],[59],[60].

2.4 MIMO Multiuser Scheduling

It is well known that in a single-input, single-output (SISO) system, where the transmitter and receivers each have only one antenna, it is optimal in terms of throughput to transmit to a single user at a time using all available transmit power [18],[61]. The random fluctuations inherent in the channel combined with a large user pool result in there likely being at least one user with a very good channel at any given time. Different users are hence selected at different scheduling intervals in a time division multiple access (TDMA) scheme. This concept is known as multiuser diversity.

It is thus natural that initial scheduling efforts in MIMO were directed towards exploiting the same idea. However, having multiple antennas at the transmitter and receivers has the effect of reducing the magnitude of those fluctuations; this phenomenon is sometimes called “channel hardening” [62]. This reduces the amount of multiuser diversity that can be exploited through only TDMA. Some early schemes proposed to counteract channel hardening by artificially increasing the variations in the channel. In [63], the multiple antennas are used to create random beam directions to exploit what the

[1] Strictly speaking, ZF-DPC in [17], etc., only considered this latter case with total interference removal and did not consider only partial interference removal like in SZF. However, SZF would be equivalent to such a scheme when the users have only one receive antenna.

authors call “opportunistic beamforming”; users who happen to be in that direction would have good channels and be scheduled. Similar concepts are used in other work (e.g. [64]), except that the beamforming changes deterministically instead of randomly, in order to aid channel estimation.

Nevertheless, such TDMA methods are inherently inferior since they only schedule a single user at a time. As we know from earlier, multiple users should be scheduled simultaneously through DPC or beamforming. The gain of DPC over TDMA was investigated in [65]. It was found that at high SNR, the gain of DPC over TDMA on the broadcast channel is $\min\{M_T/N_R,K\}$,when each user has N_R receive antennas and $M_T \geq N_R$ (as is normally the case). This was further shown in [41], where the authors demonstrated that while the sum rate of both DPC and beamforming scale as $M_T \log(\log KN_R)$, the sum rate of TDMA only scales as $\min\{M_T,N_R\}\cdot\log(\log K)$. However, there still may be an upper limit on the number of users that can communicate simultaneously, depending on the system setup. For example, as noted in the last section, when using ZFB, at most M_T users can be supported at any given time.

Properly exploiting multiuser diversity to maximize the sum rate consists of two factors; while selecting users with good channel gains is still important, users should also be chosen with a large spatial separation, to help reduce interference [66]. Thus, the selection of users can also play a role in reducing MUI just as precoding or beamforming does. [40],[67],[68],[69] contain examples of scheduling algorithms designed to, in part, select users based on reducing MUI. [67] attempts to select groups of users where a “good” antenna (i.e., with a high channel gain) for one user is “bad” for all other users. [68] selects groups of users to maximize the separation in the angles of arrival of their signals. [40] and [69] select users based in part on the orthogonality between the channels of the users.

Considering scheduling algorithms overall, the vast majority of the algorithms are designed to maximize the system sum-capacity or sum-throughput. Just a few examples of these are [40],[70],[71],[72] for ZFB, [69],[73],[74] for BD, [75] for DPC, [49],[76], [77] for ZF-DPC, [78],[79] for THP, and [80] for SZF-DPC. However, such algorithms can also tend to be unfair. Maximizing the throughput means that users who are the “best” (e.g. with good channel gains and spatially separated) would be favored, while users in poorer channel conditions would receive considerably less service. Thus, scheduling algorithms should also incorporate fairness into their scheduling criteria.

When fairness is considered, most research considers some variation of the proportional fairness (PF) criterion [63],[81]. This criterion schedules users with the highest ratio of current rate to average rate, selecting users when they tend to be near a peak in their own relative signal strength. (We discuss the PF criterion further in Chapter 3.) Examples include [82],[83],[84],[85],[86]. While the PF criterion has been extensively examined, it does not guarantee any specific requirements for throughput, delay, etc. A related metric is the (weighted) alpha rule [87],[88], which can be tuned between the performance of maximum throughput, proportional fairness, and max-min fairness, the latter of which maximizes the minimum average throughput of all the users. In past work [89],[90], we have also proposed a metric to linearly tune the system performance between that of maximum throughput and proportional fairness.

As a general rule, a scheduler is usually aiming to optimize some sort of utility function. That utility function can incorporate whatever quality of service (QoS) parameters are important to the scheduling. Several utility functions have been investigated for various resource constraints in the context of SISO systems. [91] and [92] define metrics to account for the queue length or delay of users. In [91], the Modified Largest Weighted Delay First (M-LWDF) algorithm selects users with the largest utility of $\gamma_k W_k R_k$, where W_k is the delay experienced by user k, R_k is the current rate supported by user k, and γ_k is a weight that can account for different QoS classes[1]. W_k can also be replaced by Q_k, the length of the queue (in packets or bits) for user k. [92] defines the exponential rule, which replaces W_k or Q_k with an exponential function of the delay or queue length. Both of these algorithms are known to be throughput-stable, which means that the average queue length for each user shall remain finite. The algorithms are meant to keep the chance of the delays or queue length exceeding some maximum threshold below a specified probability. They are thus useful in conjunction with delay-sensitive applications.

[93] defines a series of utility functions that are optimal in terms of resource allocation under either long-term resource-sharing constraints, or long-term minimum-performance constraints. In the first of these cases, consider a system that wishes to maximize some utility U (for example, throughput), subject to users being served a certain proportion of the time. The utility for each user is then of the form $U_k + v_k$, where U_k is the current utility of user k (e.g. the current rate it can support), and $v_k \geq 0$ ensures

[1] [91] recommends making γ_k inversely proportional to the average rate of user k, thus giving it some similarity to the PF criterion.

that user k has received its allocated proportion of time. v_k will be zero if the user has been served enough to satisfy the constraint, and greater than zero otherwise. This form is meant if the resource shared (e.g. time) is not the same as the utility being maximized (e.g. throughput). If they are the same (e.g. each user k should receive at least a certain proportion $0 \le a_k \le 1$ of the overall throughput), the utility function should instead be of the form $(\kappa + v_k)U_k$, where $\kappa = 1 - \sum_{\forall k} a_k v_k$ and $\sum_{\forall k} a_k \le 1$. For the case of minimum-performance guarantees (e.g. each user shall receive a certain minimum throughput), the utility function takes the form $\alpha_k U_k$, where $\alpha_k \ge 1$ ensures that user k has received its minimum performance. α_k will be one if it has received that minimum, and greater than one otherwise. [93] proves that these forms of utility functions will maximize the desired utility in the long term under the specified constraints, and explains how to adjust v_k and α_k in order to properly track whether the constraints are being met. In general, the parameters can be updated with an exponential filter at each scheduling interval.

One drawback to the above utility functions is that technically they are only defined for the case when a single user is served at a time. However, it is still possible to use them in MIMO applications. For example, it is shown in [94] that the multiuser equivalent of the M-LWDF algorithm is simply to maximize $\sum_{\forall k} \gamma_k W_k R_k$. The metrics of [93] have been extended to an OFDM system in [95] by simply summing the metric over all subcarriers, but this extension also assumes that no more than one subcarrier is assigned per user, and no more than one user is assigned per subcarrier. Summing those metrics over all users would likely still work to some degree in a MIMO system with multiple simultaneous users. However, the metrics may no longer be optimal for maximizing the utility. To the best of our knowledge, this has not yet been formally investigated in the literature.

Investigations of scheduling for QoS in MIMO systems have only recently become of interest. The problem of power minimization under minimum rate / SINR constraints is somewhat popular; see for example [83],[96],[97]. However, in this work, we are more interested in the problem of maximizing utility (usually involving throughput), subject to QoS constraints and a maximum power constraint. In general, one may consider two classes of service or traffic: real-time (i.e., delay sensitive, such as video traffic), and non-real-time, or best effort (which can tolerate some delay without harm). When these two types of traffic coexist in the same system, the scheduler can deal with this in one of two ways. First, the scheduler can deal with all types of traffic at once, allowing the QoS

of the two classes to be handled by the utility function. The scheduler should assign service to the real-time data more often. Investigations in [98],[99],[100] use this option; their results demonstrate how well certain types of utility functions handle the various QoS constraints. Unsurprisingly, utility functions incorporating delays handle delay-sensitive traffic the best. However, there is also a tradeoff in throughput and delay; increasing the average throughput also increases the average and peak delay users experience. The second option is to differentiate between the two classes and assign real-time traffic with higher priority. These schemes assign a certain proportion of the available resources to the delay-sensitive traffic first. Then, if there are resources remaining, they are allocated to best-effort traffic. Examples of this include [101] and [102]. This second option tends to meet QoS constraints somewhat better, but at the expense of overall system throughput. [103] describes a hybrid between the two choices; users are assigned resources first by the utility function for all classes. Then, if certain users have constraints that are being violated, the scheduler goes back and removes resources from users that do not require them, and reassigns them to users who do.

Overall, MIMO scheduling with QoS has not been explored as fully as MIMO scheduling without QoS. This is an open field to which research effort can be dedicated, especially as QoS concerns become more prevalent in wireless cellular networks.

Chapter 3

Genetic Scheduling Algorithms for Downlink Transmission in MIMO Single- and Multi-Carrier Systems with Dirty Paper Coding

3.1 Introduction

Utility-based scheduling algorithms ultimately are a specialized form of optimization. There exists some type of metric or cost function that is to be maximized or minimized, which incorporates the relevant parameters, usually under a certain set of constraints. Examples of these parameters may include throughput, packet delay or queue length [91], etc. For the downlink, the metric typically includes the maximization of a function of the system capacity or throughput, under the constraint that the total power allocated to all the scheduled users should not exceed the total available power. On the uplink, the metric may be to minimize the total transmit power subject to each user sending data at a given rate or having a minimum SINR [55]. For the case of a MIMO system, there would likely be an upper limit on the number of users N_S that can be scheduled simultaneously. In certain cases like in a system operating under optimal dirty paper coding (DPC) [15],[16], [17], this constraint may be soft, i.e., the system does not need to (but can) transmit to more than N_S users at a time [75]. Under other schemes like ZFB [40] or BD [46], that constraint is instead a hard limit; the system cannot transmit to more users than there are transmit antennas. There may furthermore be quality of service (QoS) parameters to satisfy. In some cases, certain parameters (e.g. packet delay) may change with time; most often, in these cases, those parameters are treated as constant during the scheduling interval, and are updated after a scheduling decision is made.

The scheduling decision does not necessarily consist of only which users to schedule, especially with a cross-layer design. As a general rule, the factors that the scheduler can control during the decision process may also include the order in which those users are

encoded, the amount of data sent to each of the scheduled users (which is usually expressed in terms of throughput and may depend on factors such as modulation and coding, which may or may not be under the control of the scheduler), and the power to be allocated to each of the scheduled users (generally set as required to achieve the scheduled rates). For MIMO systems, the scheduler may also decide how many data streams should be sent to each user, what beamforming vectors / transmit filters should be used (in the case of linear precoding), and which antennas to transmit with (e.g. if the system supports antenna selection).

Strictly speaking, to determine the optimal set of users, the algorithm would have to search over all possible combinations of at most N_S users from the pool of K users:

$$N_{combinations} = \sum_{k=1}^{N_S} \binom{K}{k}. \tag{3.1}$$

Note that in certain scenarios, e.g. at low SNR, it may be in fact optimal to schedule less than the maximum number of users in order to maximize / minimize the utility function. Thus, the sum over k is required in the above equation, instead of simply having $k = N_S$.

In the case of order-dependent processing (e.g. DPC, THP, SZF), the order in which the selected users are encoded will also affect their resulting rates, and hence may change the utility function value. Thus, the size of the search space increases to:

$$N_{ordered\_selections} = \sum_{k=1}^{N_S} k! \binom{K}{k}. \tag{3.2}$$

This exhaustive search and ordering quickly become infeasible as either the number of supportable users N_S or the number of active users K grows. Consequently, lower complexity methods are required.

In this chapter, we investigate the use of genetic algorithms for lower complexity downlink scheduling in a MIMO system. To begin, we first examine the general scheduling optimization problem that genetic algorithms are meant to simplify. We then discuss genetic algorithms and how they can be adapted to the scheduling problem. The remainder of the chapter discusses the simulation setup and results for a genetic scheduling algorithm in single-carrier and multi-carrier MIMO systems employing DPC. The performance is compared to the optimal solution. Issues of convergence and complexity are also discussed. Our contributions in this chapter have appeared in [104], [105],[106].

3.2 General Design and Optimal Solution for MIMO Multi-Carrier Scheduling

Before discussing the genetic algorithm, first the MIMO multi-carrier problem that the genetic algorithm is intended to solve should be examined in more detail. Most commonly, the scheduler is designed to somehow maximize the throughput of the system and to the individual users, or at least factor in those rates in its decision. As such, the utility function $G(R_1,\ldots,R_K)$ in reference to the user rates R_k is usually constrained to be:

$$\frac{\partial G}{\partial R_k} > 0, \quad \forall R_k \geq 0, \forall k \,. \tag{3.3}$$

In other words, if the rate for any user is increased while the rates for the other users remain constant, the utility function should also increase. It is possible for the utility function to be discontinuous, particularly if hard QoS constraints must not be violated. For example, if a certain minimum throughput is guaranteed, the function may have a low value if a user is below that threshold; otherwise, when the user is above the threshold, the function will have a high value. Nonetheless, the constraint in (3.3) can still be considered valid by defining the partial derivative to be some positive value at the discontinuity, whether by using the derivative immediately before or after the discontinuity or, perhaps, using the Dirac delta function at the discontinuity (i.e., considering the derivative to be "infinitely positive") in the case of a step discontinuity in the utility function.

Given the channel matrices $\{\mathbf{H}_{j1},\ldots,\mathbf{H}_{jK}\}$ of the pool of users on each subcarrier j, the goal of the scheduling algorithm is to find the set of scheduled users $\mathbb{S}$, the encoding order π of the scheduled users, and the transmit covariance matrices $\mathbf{\Sigma}_{jk}$ for those users such that $\sum_{\forall j,k} Tr\left(\mathbf{\Sigma}_{jk}\right) \leq P$ and the resulting rates R_k, $\forall k \in \mathbb{S}$ (where $R_k = \sum_{\forall j} R_{jk}$) maximize the utility function $G(R_1,\ldots,R_K)$. It is assumed that for any user $k \notin \mathbb{S}$, $\mathbf{\Sigma}_{jk} = \mathbf{0}$, $\forall j$, where $\mathbf{0}$ is an all-zero $M_T \times M_T$ matrix, and hence, $R_k = 0$. For $k \in \mathbb{S}$, $\mathbf{\Sigma}_{jk}$ is positive semidefinite and has $Tr\left(\mathbf{\Sigma}_{jk}\right) > 0$ for at least one subcarrier j, and thus, $R_k > 0$.

3.2.1 Optimal Solution: Mixed Integer Programming and Power Waterfilling

The optimal solution can be obtained by decoupling the variables of the scheduled set of users $\mathbb{S}$ and their encoding order π from that of the transmit covariance matrices $\mathbf{\Sigma}_{jk}$.

That is, for every possible $\mathbb{S}$ and π, $\mathbf{\Sigma}_{jk}$ can be calculated independently from any other set / order. The optimization then proceeds as follows.

1) For a given set $\mathbb{S}$ and encoding order π, given the channel matrices $\mathbf{H}_{jk}$ on all subcarriers, calculate the optimal power transmit covariance matrices $\mathbf{\Sigma}_{jk}$ through waterfilling for the selected users and encoding order. The user rates $\{R_1,\ldots,R_K\}$ can thereby be determined from the covariance matrices. This waterfilling may possibly account for certain aspects of the utility function G. For instance, the matrices $\mathbf{\Sigma}_{jk}$ and rates R_k that would maximize a weighted sum rate $\sum_{\forall k} w_k R_k$ with specific weights w_k may be different than for another set of weights, and different again than, for example, an unweighted sum rate (i.e., if the weights are set to the same positive constant for each user).
2) Based on the computed rates, determine the value of the utility function $G(R_1,\ldots,R_K)$ for this selection of users and encoding order.
3) Repeat the first two steps for all other possible selections of users and encoding orders. The total number of possible solutions in the general case to exhaustively search through is given by (3.2).
4) Select the set $\mathbb{S}$ and encoding order π that gives the maximum value for the utility function as the scheduling decision (possibly using secondary criteria in the event two solutions give the same maximum).

As mentioned above, the user rates can be expressed in terms of $\mathbb{S}$, π, and $\mathbf{\Sigma}_{jk}$. Thus, the utility function can also be expressed in terms of those variables, i.e.,

$$G\left(R_1,\ldots,R_K\right) = G\left(\mathbb{S};\pi;\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right). \tag{3.4}$$

The optimal utility function value is therefore:

$$G^* = \max_{\mathbb{S},\pi,\left(\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right)} G\left(\mathbb{S};\pi;\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right), \tag{3.5}$$

and for a given $\mathbb{S}$ and π,

$$G^*_{\mathbb{S},\pi} = \max_{\left(\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right):\mathbf{\Sigma}_{jk}\succeq 0,\forall j,k;\sum_{\forall j,\forall k\in\mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right)\le P} G\left(\mathbb{S};\pi;\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right). \tag{3.6}$$

To make the optimization problem slightly less complex, we modify the transmit power constraint to:

$$\sum_{\forall k} Tr\left(\mathbf{\Sigma}_{jk}\right) \le P/N_C \ . \tag{3.7}$$

That is, the available transmit power is equally divided across all N_C subcarriers. This largely enables the subcarriers to be scheduled independently of each other, depending on the utility function.

We show in Appendix A that the optimal transmit covariance matrices in (3.6) for a given set of users and encoding order when using DPC satisfy the following set of equations:

$$\mathbf{\Sigma}_{jk} = \mathbf{0}_{M_T \times M_T}, \quad \text{if } s_{jk} = 0; \tag{3.8}$$

$$\sum_{k \in \mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) = P/N_C, \quad j = 1,...,N_C; \tag{3.9}$$

$$\mu_{jk} Tr\left(\mathbf{\Sigma}_{jk}\right) = 0; \tag{3.10}$$

$$\sum_{i \le k}\left(\frac{\partial G}{\partial R_i}\mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u \ge i}\mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^*\right) - \sum_{i<k}\left(\frac{\partial G}{\partial R_i}\mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u>i}\mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^*\right); \tag{3.11}$$

$$-\lambda_j s_{jk}\mathbf{I}_{M_T} - \mu_{jk}\mathbf{I}_{M_T} + \mathbf{\Omega}_{jk}^T = \mathbf{0}_{M_T \times M_T}, \quad \forall k \in \mathbb{S}$$

$$\mathbf{\Sigma}_{jk}, \mathbf{\Omega}_{jk} \succeq 0. \tag{3.12}$$

In the above, s_{jk} is a binary variable equal to 1 if user k is scheduled on subcarrier j, and 0 otherwise. With the optimal transmit covariance matrices, the optimal user rates can then be determined, and hence, the maximum value of the utility function can be found for the selected set of users and encoding order.

The covariance optimization problem is not convex, thus finding the optimal $\mathbf{\Sigma}_{jk}$ directly on the BC can be quite difficult. It is, in general, much easier to solve for the optimal transmit covariance matrices on the dual MAC and then convert those matrices into their equivalents on the BC (as in [107] and [82]) than to directly solve for the BC matrices. It is in fact unnecessary to transform to the BC until a scheduling decision is made on the dual MAC. We therefore use this duality and the methods in [107] and [82][1] to find the optimal transmit covariance matrices in this work.

3.2.2 Example Scheduling Criteria and Their Utility Functions

In this work, we consider two criteria for scheduling in multi-carrier systems employing DPC: the maximum throughput criterion and the proportional fairness criterion.

[1] At the time this work was carried out, we were only aware of the method of [82] to maximize a weighted sum rate for the MAC when the users have an arbitrary (i.e., more than one) number of receive antennas. Since then, far more efficient methods for maximizing a weighted sum rate have been published, such as that in [108].

The maximum throughput criterion maximizes the instantaneous system throughput at each scheduling instance, i.e.,

$$G_{MT} = \sum_{k=1}^{K} R_k \,. \tag{3.13}$$

Maximizing G_{MT} results in the highest possible system capacity; the metric is simply that of the system sum rate. Under DPC, however, the encoding order has no effect on the system sum-capacity. Although the individual user rates and the optimal transmit covariance matrices will change, the sum of those rates will not. Hence, multiple solutions can give the same value of the metric. For this work, we break the tie in these situations in a max-min sense: whichever encoding order maximizes the smallest of the user rates is the preferred solution.

The maximum throughput criterion suffers from a lack of fairness. Users that experience consistently poor channel conditions (e.g. far from the base station or in a shadowed location) will receive very little service and, in extreme cases, no service at all. Hence, some fairness constraint is usually imposed on the scheduling algorithm.

The proportional fairness criterion is one of the best known compromises between fairness and throughput. A scheduler that is proportionally fair will maximize the following utility function [19]:

$$U_{PF} = \sum_{k=1}^{K} \log_2\left(\overline{R}_k\right). \tag{3.14}$$

Here, $\overline{R}_k$ is the average long-term throughput of user k. A scheduler is said to be in a state of proportional fairness if, by changing from the current rate vector $\mathbf{r}_x$ to another rate vector $\mathbf{r}_y$, the sum of the proportional changes in rates is less than or equal to zero [19].

A proportionally fair scheduler is often used in third generation cellular systems. The form in (3.14) is not normally used, because it is non-causal (the long-term average is the expected rate over all time, including the future), nor is it in the form required in (3.3). Rather, the following form is usually used:

$$G_{PF} = \sum_{k=1}^{K} \frac{R_k}{\overline{R}_k} \,. \tag{3.15}$$

A scheduler that maximizes G_{PF} also maximizes U_{PF}, as proven in [19],[109],[110]. This form of the utility function is a weighted sum rate, where the users' weights are $w_k = 1/\overline{R}_k$. The non-causal average throughput $\overline{R}_k$ is normally approximated by a moving average over a window of past slots and is calculated by:

$$\bar{R}_k(t+1) = \left(1 - \frac{1}{t_c}\right)\bar{R}_k(t) + \frac{1}{t_c}R_k(t). \tag{3.16}$$

$\bar{R}_k$ is updated for the next slot ($t + 1$) after the scheduling decision for slot t has been made, and thus the rates $R_k(t)$ are known. If user k is not scheduled at slot t, $R_k(t) = 0$. The time constant t_c is the window size for the averaging; in this work, we use $t_c = 100$ slots.

3.3 Genetic Algorithms

3.3.1 General Description

One potential suboptimal solution to the scheduling optimization problem lies in genetic algorithms [111]. Genetic algorithms (GAs) to some degree mimic biological systems in their operation. The algorithm starts with a set (or *population*) of data structures that are called *chromosomes* (or sometimes *genes*), which represent possible solutions to the optimization problem. These chromosomes are similar in concept to their biological counterparts. During each iteration (or *generation*) of the algorithm, several pairs of the chromosomes (called *parents*) from the population may be crossbred with a *crossover* operation by swapping information between the data sequences to form *offspring* or *children*. The likelihood of a chromosome being selected for breeding is related to its *fitness*, i.e., how good of a solution to the utility function that chromosome represents. Chromosomes that are more fit are more likely to be selected for breeding. The child chromosomes may also undergo a *mutation* operation, wherein each and any of the constituent elements of the children has a probability of being randomly altered. The offspring then replace their parents for the next iteration of the algorithm. For each generation, the child chromosomes should be checked to see if they meet the constraints on the problem, and for fitness in conjunction with the original utility function. *Elitism* may also be employed, wherein the "best" chromosome(s) may be kept from the previous generation when a new generation is created, so as not to lose the best solution found thus far. Through this "survival of the fittest" process, the chromosomes eventually evolve towards the optimal solution, while the random breeding and mutation process helps to ensure that the algorithm does not get stuck at a local maximum, but rather converges to the global maximum. Figure 3.1 shows the overall operation of a genetic algorithm. The specific implementation of the algorithm will depend on the problem being solved, but most GAs follow this general structure. [112] contains a good overview of the various options available in the design of each step of the GA.

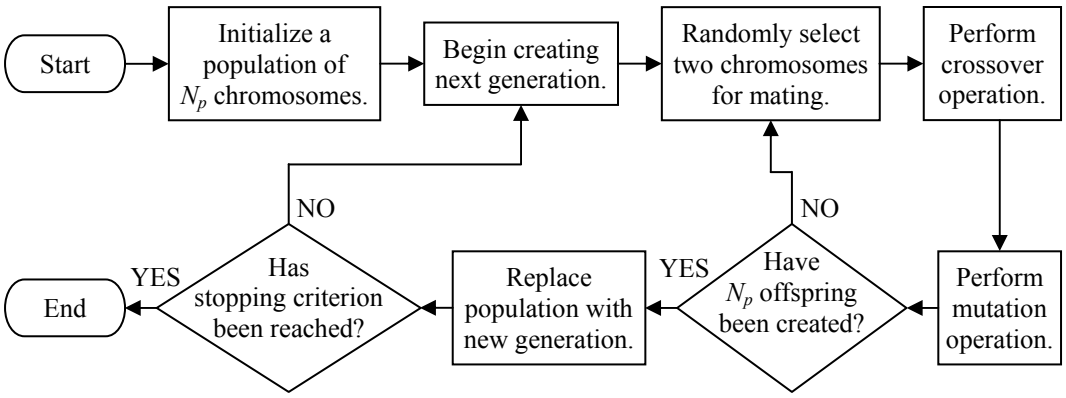

Figure 3.1: Flow diagram of a general case genetic algorithm.

Genetic algorithms belong to a class of stochastic and heuristic optimization techniques that operate in a somewhat random fashion, yet are guided in some manner towards the optimal solution. Other such techniques include simulated annealing [113] and particle swarm optimization [114]. GAs are a fairly popular method of optimization; within just the field of communications they have been used to help design and optimize antennas and electromagnetic devices [112], analog circuits [115], digital filters [116], wireless sensor networks [117], radio spectrum allocation [118], antenna selection algorithms [119], and so on. One reason for their popularity is their robustness to the utility function being optimized. The stochastic nature helps ensure that the algorithm finds a solution to the problem close to the global optimum. GAs have the benefit that they can still operate even if the utility function is not convex. (In the context of scheduling, when QoS constraints such as resource sharing or minimum performance requirements are included, this may end up being the case.) Many other optimization methods, although being able to find a stationary point faster than a GA in some cases, also require a convex function to operate on to ensure that the stationary point is a global optimum point instead of a local optimum. Genetic algorithms are also particularly suited for scheduling operations, as they are known for their speed of convergence. Due to its random nature, the algorithm may take some time to locate the overall optimal solution; in fact, for a limited operating time, finding the global optimum is not necessarily guaranteed. However, the GA will usually find a very good solution that is close to the optimal one quickly.

Dealing with any constraints can be one of the more complicated issues in dealing with genetic algorithms. In each of the stages of the algorithm, the constraints can be considered in some fashion. During the mating selection process, the probability of a chromosome being selected for mating can be related to the fitness of that gene, which in

turn can be related to the constraints. That is, if a chromosome violates the constraints of the optimization, it should be of lower fitness, and consequently can be given a lower probability of selection. Care should also be taken in how rigidly the constraints are dealt with in the fitness function. If a constraint is violated, but only slightly, that solution may still be acceptable depending on the situation, but with some penalty in terms of its fitness. However, if multiple constraints are involved, it could become difficult to tell from a low value of the fitness function alone if a potential solution provides a high value for one parameter to be optimized while violating several other constraints, or if the solution is poor but does not violate any constraints. When performing the crossover and mutation processes, the constraints can also be considered. If a child of the process does not meet the constraints, that child could simply be removed. Alternatively, the child chromosome could be repaired so that it does meet at least some of the constraints. It may also be possible to define a birthing or mutation function that causes a child to fall within the constraints so long as the parents already meet the constraints. The actual specifics of the process depend on the implementation of the algorithm and the optimization problem to be solved. Some possible methods of handling constraints in GAs are discussed in [120].

3.3.2 Genetic Algorithm for Order-Dependent Precoding and Scheduling

A genetic algorithm used for scheduling works somewhat differently from a typical scheduling algorithm. Usually, a scheduling algorithm calculates a metric for each user, then selects a user or users based on those metrics. After the selection, the algorithm must usually then go back and calculate new metrics for the remaining users based on the selected user(s). The algorithm iterates in this fashion until a maximum of N_S users have been selected. In comparison, the genetic algorithm does somewhat the reverse; instead, it first selects an entire potential group of users via the chromosomes, and then calculates a fitness function for that set of users after the selection. Only after the users are selected does the algorithm check the fitness of the selection and see if the constraints are met. In essence, the GA partially decouples the process of selecting a group of users and encoding order from the calculation of the fitness for that group.

As already mentioned, the scheduling optimization problem the GA is attempting to solve has two parts: the optimal selection of a group of users and encoding order, and the optimization of the utility function for that group and order. Within this work, we use the GA to solve the selection portion of the problem only; this is the more difficult part of the

two. The chromosomes represent potential solutions to that problem *only*, i.e., which users will be scheduled and in what order will their data be encoded. The GA mating process does not in and of itself calculate the fitness of each chromosome. This fitness can be calculated separately from the mating process itself, and within this work, (globally) optimally. The utility function (fitness) that is being maximized is a scheduling metric for each selection and ordering, such as those given by (3.13) and (3.15).

A genetic algorithm was used in [109] to implement the maximum throughput and proportional fairness scheduling criteria in the context of orthogonal transmit spatial multiplexing (i.e., ZFB) for mobiles with a single receive antenna. In that case, the only scheduling constraint was that the number of users to be selected must be less than or equal to the number of transmit antennas. The algorithm was able to achieve a performance within about 0.5 dB of an optimal scheduler (implemented via integer programming and exhaustive search) in a much smaller number of calculations. For example, with 20 active users and 4 transmit antennas, the algorithm converged to a good solution in about 1/36 of the time of the optimal scheduler.

The overall problem investigated herein is fairly similar to the problem in [109]. As such, the GA in [109] provides a good basis from which to expand upon for scheduling with order-dependent encoding (e.g. DPC), and also for multi-carrier transmission. The key differences between the investigation in [109] and in this work are listed as follows.

1) The general design in [109] was for a system that uses ZFB, whereas this work considers DPC. Consequently, the design in this work also considers the effect of the encoding order on the solution.
2) The problem that was considered in [109] encompassed only single-antenna receivers, whereas this work considers receivers with multiple antennas. Hence, the optimal solution involves power allocation over complex-valued covariance matrices instead of real-valued scalar powers.
3) The system in [109] used single-carrier transmission. In comparison, this work for DPC also considers a multi-carrier system, and thus, the solution involves resource allocation and optimization across multiple carriers, adding an additional dimension to the problem.

In [109], Lau uses a bit vector as a chromosome to indicate which users are scheduled. In this work, we extend this scheme to account for an encoding order and to allow for scheduling on multiple carriers. To begin, considering just a single-carrier

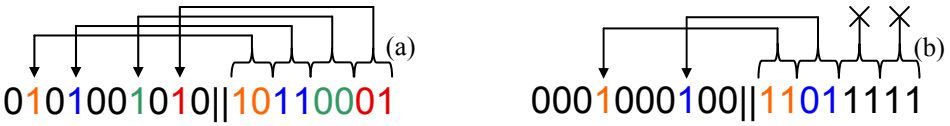

Figure 3.2: Two typical chromosomes for single-carrier DPC scheduling with $N_S = 4$ and $K = 10$. (a) Users 2, 4, 7, and 9 are scheduled and have order numbers '10', '11', '00', and '01', respectively. The users are therefore encoded in the order {7,9,2,4}. (b) Users 4 and 8 are scheduled and have order numbers '11' and '01', respectively. They are therefore encoded in the order {8,4}. The remaining two order numbers in the tail of the chromosome are ignored.

system, denoting which users are scheduled can still be accomplished by a vector of K bits. A '1' in position k of the vector denotes a scheduled user, while a '0' denotes an unscheduled user. As in [109], the weight w (i.e., the number of '1's) of this vector should be between 1 and N_S inclusive. The binary chromosome representation is a particularly good choice for scheduling (as opposed to, say, real-valued chromosomes), since the scheduling decision is itself binary; either a user is scheduled, or it is not.

This leaves the encoding order to be represented in the chromosome. In our modification, we refer to the K-bit vector that denotes which users are scheduled as the "head" of the chromosome. The additional bits in the chromosome that denote the encoding order of the scheduled users we refer to as the "tail". The tail of the chromosome consists of $N_S \times \lceil \log_2(N_S) \rceil$ bits (where $\lceil \cdot \rceil$ denotes the ceiling function). Each group of $\lceil \log_2(N_S) \rceil$ bits denotes the relative position of a scheduled user in the encoding order and is referred to as an "order number". The first order number in the tail of the chromosome refers to the first '1' in the head of the chromosome, the second order number refers to the second '1', and so on. The relative order of the users is determined by the binary value of the order number, i.e., a larger value means a later position in the ordering. The only constraint on the representation is that the first w order numbers should be unique, i.e., they need not be sequential. Furthermore, in the event that fewer than N_S users are scheduled, any order numbers beyond the first w are simply ignored. Two example chromosomes are shown in Figure 3.2. Note that the representation is independent of the number of antennas at the receivers, allowing the GA to be scaled to any number of receive antennas or to systems with different numbers of receive antennas per user. (However, in other encoding schemes, the number of supportable users N_S may depend on the number of receive antennas.)

The proposed chromosome representation is not the only possible one, nor is it necessarily the most efficient. For example, an alternative chromosome could consist of N_S groups of $\lceil \log_2(K) \rceil$ bits. Each group would stand for the user ID of the users to be

scheduled, while the encoding order is simply represented by which group the ID is in. That is, the first group of bits represents the first encoded user, and so forth. An additional $\lceil \log_2(N_S) \rceil$ could be added to indicate how many users are scheduled, to allow scheduling less than the maximum supportable number of users. Since K is usually much larger than N_S, this representation would use less memory than the first proposed scheme. However, our chosen representation has other benefits in its favor:

- The encoding order is more decoupled from the user selection. Therefore, it is somewhat easier to consider the same selection of users, but a different encoding order, by simply changing a few bits in the tail of the chromosome. In comparison, to exchange the encoding order of two users in the second representation, all the bits from two of the groups would have to be swapped.
- Crossover operations are more likely to lead to invalid children, and corrections are more difficult, with the second representation. During the crossover operation, it could be possible to create a user ID that does not exist. This scenario is the most likely to occur if K is slightly over a power of 2. For example, if $K = 36$, if a crossover occurs after the first bit of an ID of 33 (‘100001’) and an ID of 26 (‘011010’), the ID of ‘111010’ (58) would be created, which is larger than the number of users. This could be avoided by restricting crossovers in the second representation to between the groups of bits, but it is also still possible to create a chromosome that has the same user scheduled twice. In both cases, correcting the chromosome is more difficult. Multiple bits in general would have to be changed to correct the chromosome. In comparison, the errors that can occur in the first representation are much easier to fix most often with some simple bit toggles.
- Mutation is easier and / or works better with the first representation. Mutation in the first case is a simple bit-toggle operation, which is extremely efficient to implement. This same type of mutation could be applied to the second representation, but the mutation would not work as well. The probability of mutation tends to be fairly low, so within a given user ID, only one bit is likely to be changed, if any. This restricts the mutation of the ID to a sort of Gray mapping [20] around the original ID. That is, a single-bit mutation would only change the ID to one of $\lceil \log_2(K) \rceil$ other IDs near the original binary value. This is not necessarily a good thing in scheduling. Rather, it would in general be more beneficial for a mutation to possibly result in any other user being considered for scheduling, for the largest diversity in the choice of selection. This is accomplished in the first representation. Doing so in the second representation

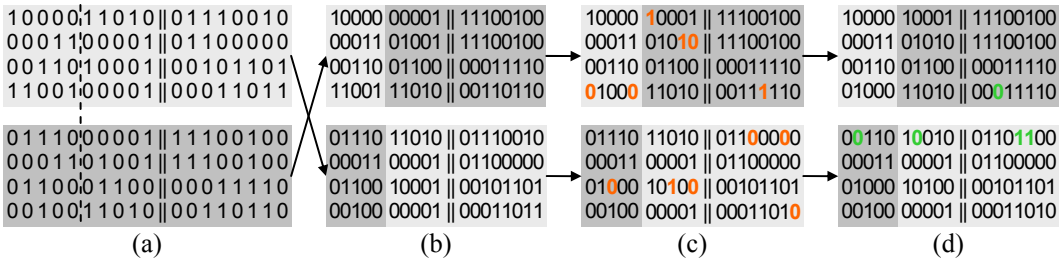

Figure 3.3: Example of GA operation for multi-carrier DPC scheduling with four subcarriers, four transmit antennas, and 10 active users. (a) Two typical chromosomes and a random crossover location. (b) Crossover operation. (c) Mutation operation. (d) Correction of invalid child chromosomes.

is still possible, but it would require a multi-bit mutation operation, operating on a bit-group scale instead of on individual bits, which is not as operationally efficient.

Extension to a multi-carrier scenario is quite simple. Rather than a bit vector, we instead use a bit matrix. The order-aware single-carrier chromosome structure is repeated for N_C rows, with the jth row of the matrix representing the scheduling and encoding order on the jth subcarrier. It is also possible to schedule subcarriers in groups rather than individually, in which case the number of rows would be equal to the number of groups. Typical chromosomes for the multi-carrier case are shown in Figure 3.3.

As mentioned earlier, the problem and thus our GA methodology are similar to that described in [109]. Further implementation details are as follows.

1) *Initialization*: A population of N_p chromosomes is initialized at random. As previously mentioned, this initialization is constrained such that, in the head of chromosome, the weight w_j of each of the rows is between 1 and N_S inclusive. In addition, the order numbers in the tail are randomly initialized under the constraint that, for each row, the first w_j order numbers are unique. This initial population represents the initial choices for the scheduling of users and their encoding order. In general, it is not known what the optimal ordered grouping is; thus, the initial random population provides a diverse set of possible choices from which the search for the optimal ordered grouping can be started.
2) *Selection*: In [109], an intermediate population is formed from the initial population in each generation based on remainder stochastic sampling. Chromosomes are then selected from this intermediate population for crossover and mutation. In contrast, we select chromosomes directly from the population in a "roulette wheel" method based on their fitness. The fitness of a given chromosome i is defined by the value of the utility function G_i for the solution that is represented by that chromosome with

optimal transmit covariance matrices, as given by (3.13) and (3.15). The probability of a chromosome being selected for breeding is given as:

$$p_{sel\_i} = \frac{G_i}{\sum_{n=1}^{N_p} G_n} \,. \tag{3.17}$$

Thus, the more fit a chromosome is, the more likely that it will be selected and hence pass along its characteristics to the next generation.

3) *Crossover*: Our GA uses a so-called "one-point" crossover operator similar to that in [109], with a probability of $p_c = 1$. Once two "parent" chromosomes are selected by the previous step (it is possible that the same chromosome will be selected twice), a crossover point is chosen somewhere at random along the length of the chromosome. In the multi-carrier scenario, this same location is used across all rows of the chromosome. Any bits after the crossover point are swapped between the parent chromosomes to form two new "children" chromosomes. By combining the information of the parents in such a fashion, the chromosomes combine the partial characteristics of a good solution (e.g. some of the best users to be scheduled) into a new chromosome that represents a new and hopefully better solution.

4) *Adaptive Mutation*: Mutation is applied to the children chromosomes that were created by the above crossover operation. The same adaptive mutation probability as in [109] is adapted for this work. The probability of any given bit in the children chromosomes toggling is given as:

$$p_m = \frac{1}{\beta_1 + \beta_2 \, \sigma_G / \mu_G} \,, \tag{3.18}$$

where σ_G and μ_G are, respectively, the standard deviation and mean of the fitness of the current population before selection, and $\beta_1 = 1.2$ and $\beta_2 = 10$ are constants. The crossover operator has the most notable effect on the convergence of the GA when the genetic diversity in the population is large (e.g. in the initial random population). However, once the algorithm begins to converge on a solution, the mutation operator becomes increasingly important. As the GA starts to converge, the chromosomes in the population tend to end up sharing the same characteristics. (For example, several chromosomes may have some of the same users selected to be scheduled.) Thus, the fitness of those chromosomes tends to be similar. However, it is in general unknown if a particular chromosome represents a globally optimum solution, a locally optimum solution, or otherwise. Mutation allows the algorithm to maintain a wide variety of possible solutions and helps avoid stalling on a local maximum, and it

forces the algorithm to consider solutions other than those near the current best solution. Hence, when the variation in the fitness of the population becomes smaller, the probability of mutation should increase. This is accomplished by the adaptive mutation rate in (3.18). If the algorithm is converging on a solution, the standard deviation of the fitness of the chromosomes will drop relative to the mean fitness of the population, which results in a larger value for p_m. If, instead, there is a fair amount of diversity in the fitness of the chromosomes, the standard deviation will comparatively be larger, and hence, the mutation rate will be smaller.

5) *Constraints*: During crossover and mutation, it is common for children to be created that violate the constraints of scheduling at most N_S users or of allocating each scheduled user a unique order number. These invalid chromosomes must somehow be dealt with. We correct any constraint violations as the last step in the breeding process. For each row in the chromosome, if the weight of the head is larger than N_S, '1's are toggled at random to '0's until the weight is N_S. In the less likely event that the weight of the head is zero, one bit is toggled at random to a '1'. In the tail of the chromosome, if there are duplicated order numbers, one of the duplicates is replaced at random with an order number that is not yet used by one of the scheduled users. This is repeated until no duplicates remain.
6) *Elitism*: For each generation of N_p chromosomes, only $N_p - 2$ new chromosomes are created through the breeding process. To prevent the previous best chromosome C^* from the prior generation from being destroyed in the breeding process, a copy is inserted into the new population as one of the two remaining chromosomes. This process also causes the value of the utility function found by the algorithm to be strictly non-decreasing over time; otherwise, the value could, in fact, diverge from the optimum. The final chromosome of the population is also a copy of C^*, except that the encoding order of two of the scheduled users is swapped at random.
7) *Iteration*: Once a new population of N_p chromosomes has been formed by the above selection and mutation processes, it replaces the old generation. The procedure then iterates for a total of N_g generations. The values of N_p and N_g are dependent on the maximum number of scheduled users N_S and the number of active users K, as described in Sections 3.5.3 and 3.5.4.

Figure 3.3 shows the typical operation of the GA during the breeding of two multi-carrier chromosomes within one generation. Figure 3.4 depicts a flow diagram summarizing the overall operation of our genetic algorithm.

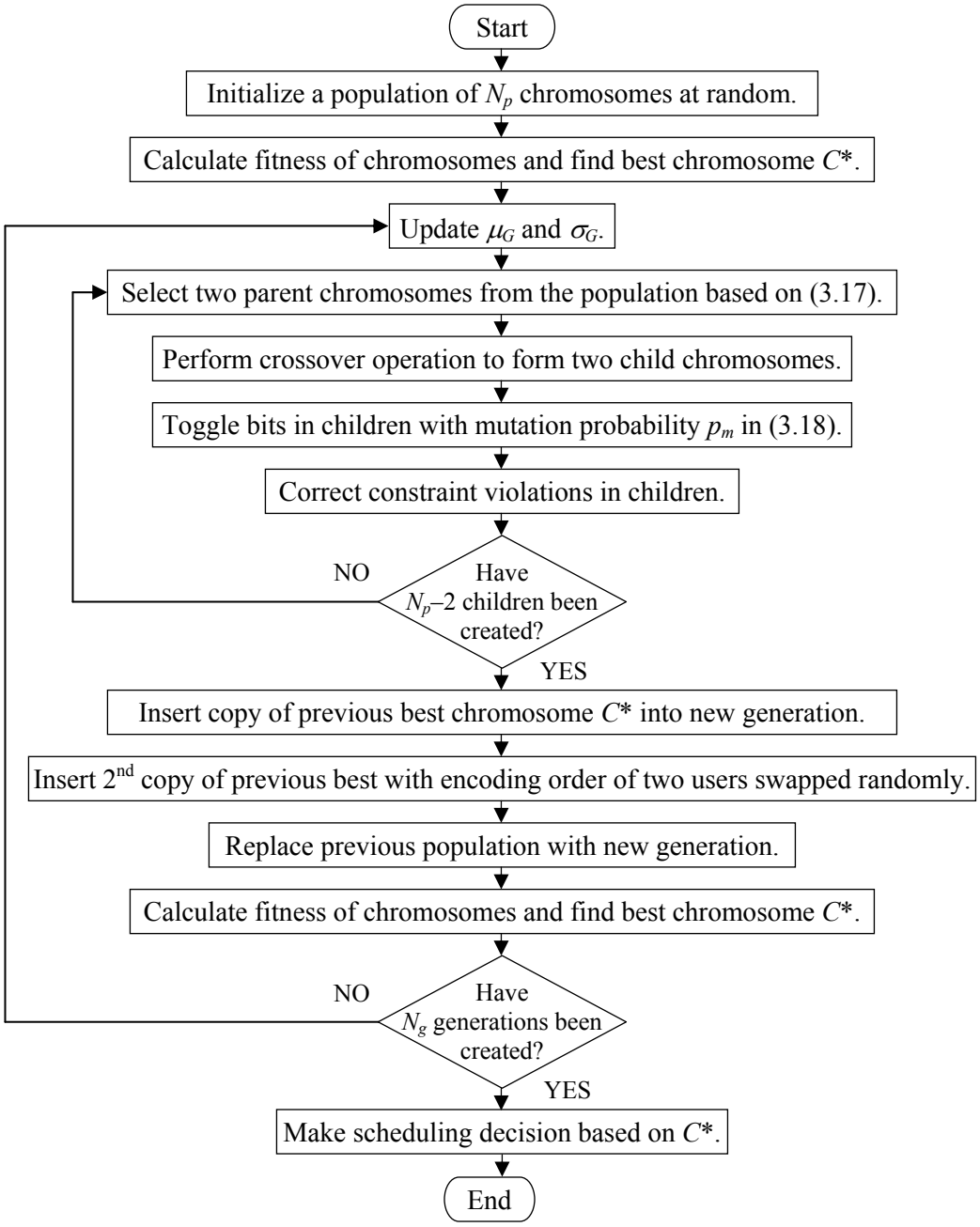

Figure 3.4: Flow diagram of the genetic scheduling algorithm.

3.4 Multiuser MIMO System Model with Single and Multiple Carriers

3.4.1 Wireless Channel Model

We wish to isolate the effects of the scheduling algorithm from any other effects that are specific to the implementation of the system (e.g. modulation and coding) or the type of data being carried. Hence, at the base station, we assume that packets arrive in such a

manner that there is always data available to transmit to all users that request service. We also assume that the data rates that each user can support are determined by channel capacity to provide an upper bound on the performance. Turbo and low-density parity-check codes [121] approach capacity within 0.5 dB; thus, this is a reasonable and commonly-used assumption.

We consider a transmission system that consists of a base station with M_T antennas, which schedules transmissions to a pool of K active users, each equipped with N_R receive antennas. The channel gain between any given transmit-receive antenna pair on any carrier is modeled as an i.i.d. circularly-symmetric complex Gaussian process with unit variance (i.e., Rayleigh fading). A block-fading (quasi-static fading) model is assumed, where the coherence time of the channel is much longer than the scheduling and transmission interval, such that the channel gains are approximately constant for the entire interval. (Such a scenario is typical in the current generation of wireless packet data systems, e.g. [4] and [122], and where the transmission interval, which is on the order of milliseconds, is much less than the coherence time of the fading channel.) All channel gains then independently change between intervals. The path loss, shadowing, and fading conditions for each user are statistically identical. In the case of the proportional fairness scheduling criterion, a log-normal shadowing component [123] with a standard deviation of 8 dB is also added to the signal to provide a variation in the average signal strength (and, hence, the average supportable rate) across the set of active users. 8 dB is used as it is approximately in the middle of the range of standard deviations observed for both macrocellular and microcellular applications [123].

The channel between the base station and the kth user on the jth carrier is defined by the $N_R \times M_T$ channel matrix $\mathbf{H}_{jk}$. The combined received signal for all users on the jth carrier is:

$$\mathbf{y}_j = \begin{bmatrix} \mathbf{y}_{j1} \\ \vdots \\ \mathbf{y}_{jK} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{j1} \\ \vdots \\ \mathbf{H}_{jK} \end{bmatrix} \mathbf{x}_j + \begin{bmatrix} \mathbf{n}_{j1} \\ \vdots \\ \mathbf{n}_{jK} \end{bmatrix} = \mathbf{H}_j \mathbf{x}_j + \mathbf{n}_j , \tag{3.19}$$

where $\mathbf{x}_j$ is the $M_T \times 1$ vector of transmitted symbols from the base station on the jth carrier, $\mathbf{H}_j$ is the $(N_R \cdot K) \times M_T$ aggregate channel matrix for all K users, $\mathbf{y}_{jk}$ is the $N_R \times 1$ vector of received symbols for user k on the jth carrier, and $\mathbf{n}_j$ is the $(N_R \cdot K) \times 1$ Gaussian noise vector on carrier j with variance[1] $\sigma_n^2 \mathbf{I}_{N_R \cdot K}$.

[1] We assume without loss of generality that $\sigma_n^2 = 1$ throughout this work.

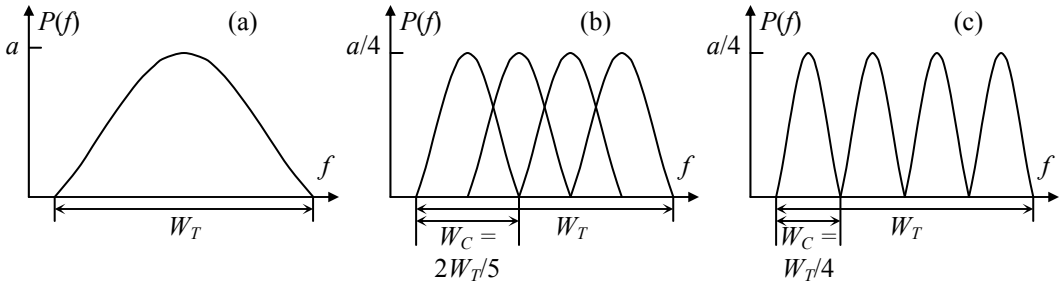

Figure 3.5: Conceptual power spectrum $P(f)$ vs. frequency f of single-carrier and multi-carrier transmissions with same normalized total bandwidth W_T. Total transmit power is divided equally among carriers. (a) Single carrier. (b) OFDM, 4 subcarriers. (c) FDM, 4 subcarriers.

Each carrier is assumed to fade independently of all other carriers. Furthermore, it is assumed that the signal transmitted on any given carrier causes zero interference on any other carrier. In the multi-carrier case in this work, we assume that the total useful bandwidth W_T (over which data signals are transmitted, not including factors like guard intervals, etc.) is identical to that in the single-carrier case to provide proper normalization for the spectral efficiency. Figure 3.5 demonstrates this with $N_C = 4$ subcarriers using orthogonal frequency division multiplexing (OFDM) and regular FDM.

It is assumed that the base station perfectly knows the channel matrices for all users at all frequencies. This knowledge can come through channel estimation at the mobiles, based on a pilot signal that the base station transmits, which is then sent to the base station on a feedback channel. Alternatively, in the case of a time-division-duplex system for the uplink and downlink, the base station can estimate the uplink channels for the mobiles from their signals and then derive the downlink channels based on channel reciprocity.

3.4.2 Physical Layer Model

As we have previously mentioned, we wish to isolate the effects of the scheduling algorithm from the specific details of the implementation of the system. Hence, we take an information-theoretic approach to the performance of the physical layer. Furthermore, we are also interested in accounting for the effect that encoding order has on scheduling and the performance of the system. Thus, we consider a system where the users' signals are encoded via DPC [15],[16],[17]. DPC is known to achieve the sum-capacity of a MIMO BC. Although DPC is currently infeasible, it can closely be approximated by some of the methods described in the previous chapter (e.g. [32],[34],[39]). Nevertheless, in this work, we are concerned with scheduling complexity, not encoding complexity,

and hence simply assume a system employing capacity-achieving DPC. To bypass the non-convexity of the BC, we instead perform utility function calculations on the dual MAC.

3.4.3 Medium Access Control Layer Model

On the medium access control layer, a scheduling algorithm is responsible for the allocation of power and resources to the active users. As mentioned, it is assumed that the base station knows the channel matrices of the users, usually via feedback from the users based on measurements made from a pilot signal. The scheduler takes these matrices as an input and attempts to maximize some sort of metric using the matrices, the total available transmit power P, and possibly other parameters. For simplicity, we assume the transmit power is divided equally among the subcarriers. The outputs of the scheduler are the set of scheduled users, the rates for those users, and the corresponding transmit covariance matrices (i.e., the power allocations). This information can be transmitted to the users on a downlink control channel.

Strictly speaking, there is no limit to the number of users that can simultaneously be scheduled with DPC. (It should be noted, however, that certain approximations have limits based on the number of transmit or receive antennas.) However, if waterfilling the power is performed to maximize the sum-throughput, chances are that certain users will end up being allocated zero power and, hence, effectively not be scheduled. It is shown in [124] that, if there are M_T transmit antennas, at most M_T^2 users will be allocated non-zero power. In general, one cannot tell how many users must be scheduled to achieve the sum-capacity. However, it has been shown in [75] that, most of the time, it is sufficient to schedule, at most, the same number of users as there are transmit antennas. Scheduling less than M_T users will often lead to a significant loss in capacity, unless the SNR is low. However, scheduling more than M_T users does not result in a significant gain in capacity; the system is already very close to capacity once M_T users are scheduled. Hence, the scheduler need not consider additional users beyond $N_S = M_T$.

Very little work in the literature looks at scheduling in conjunction with DPC alone. This is due to several reasons. In part it is due to the current infeasibility of DPC. Hence, scheduling efforts are more focused on other methods, such as linear precoding methods, or other suboptimal adaptations like ZF-DPC. Also, as just discussed, scheduling in DPC is not strictly necessary; waterfilling over all possible users in effect chooses which users to transmit to by allocating them non-zero power. However, we have focused on

scheduling under DPC for this work for a few reasons. First, DPC still is optimal, and we wish to analyze the performance of the GA in a "best case" scenario, to focus solely on the effect of the scheduling. Secondly, there may arise in the future some method that closely approximates DPC that is feasible for implementation. Such a method might have restrictions on the number of supportable users. Our GA would still be valid in this case. Lastly, the GA we have used for DPC scheduling is still trivially extendable to other precoding methods besides DPC, so it is also usable in present practical systems.

3.5 Simulation Results

In this section, we analyze the performance of the GA in implementing the maximum throughput (MT) and proportional fairness (PF) scheduling criteria in a system using DPC in both single-carrier and multi-carrier scenarios. In the multi-carrier scenario, we consider a relatively simple system with only four subcarriers to help reduce the simulation complexity. The performance of the GA is compared to an exhaustive search (ES), which serves as an upper bound. In addition, in the single-carrier case, we also compare a few results by using ZFB instead of DPC for reference to the earlier work in [109]. In all cases, a maximum of $N_S = M_T$ users are scheduled per subcarrier.

3.5.1 Single-Carrier Results

Figure 3.6 demonstrates the performance of the maximum throughput scheduler in terms of the utility function value G_{MT} (i.e., the system sum-throughput) versus SNR for various values of M_T, N_R, and K. For reference, the optimal performance of an ES is also shown. The ES provides the best possible performance for choosing a maximum of M_T out of K users to serve simultaneously using DPC. It is shown that the GA achieves approximately 94–99% of the sum-throughput relative to the ES while being significantly less complex. Interestingly, the relative performance of the GA compared to the ES is essentially independent of the SNR. Over the SNR range of 0 to 10 dB, the change in relative performance is at most about 1.7%, for $(N_R,M_T,K) = (1,4,20)$; for the other curves, the change is even less. It is also observed that the performance of the GA relative to the ES is better for $M_T = 2$ than for $M_T = 4$. This is not that surprising, as the larger the value of M_T (and hence N_S), the larger the search space is for the optimal user selection (as given by (3.2)). Nonetheless, even for this more complicated problem, the GA performs quite well. Overall, in terms of SNR, the GA performance is generally inferior to the optimal performance by about 0.5 dB.

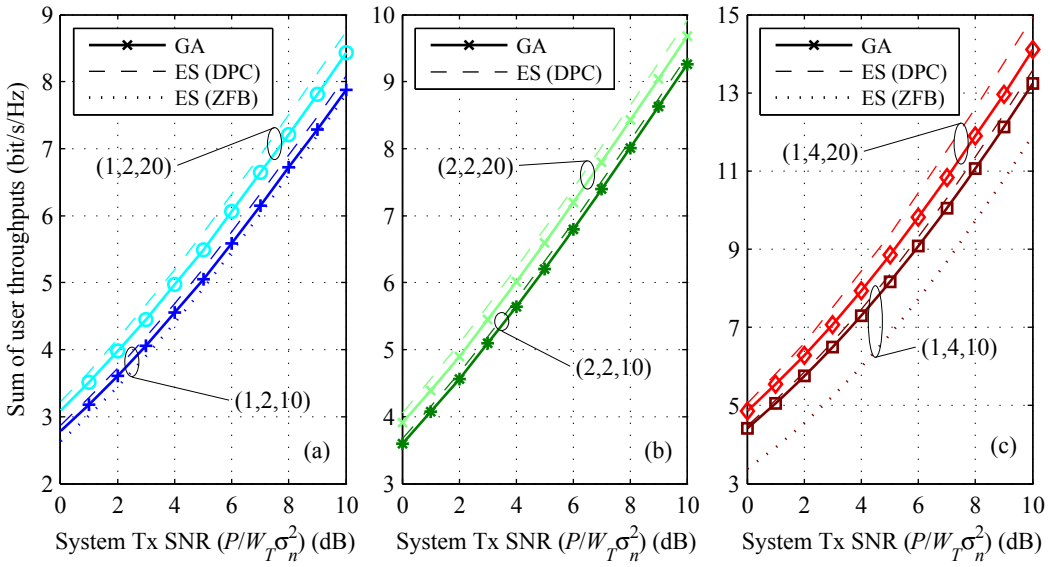

Figure 3.6: Performance of maximum throughput scheduling versus SNR for a (N_R,M_T,K) single-carrier MIMO system implemented via GA and ES. (a) $N_R = 1$, $M_T = 2$, and $K = 10, 20$. (b) $N_R = 2$, $M_T = 2$, and $K = 10, 20$. (c) $N_R = 1$, $M_T = 4$, and $K = 10, 20$.

For reference, we also simulated the performance of the system using DPC, but wherein the system attempted to transmit to as many users as possible, and with optimal power allocated to those users in order to maximize the system capacity. This is to reinforce the conclusions drawn in [75]. As expected, there is very little difference in the sum-throughput when scheduling only M_T users compared to as many users as possible. In fact, for the most part, the performance is largely coincident with the ES results in Figure 3.6. Thus, we do not plot those results there, as the lines would essentially overlap. For $M_T = 4$, we observed at best less than an additional 0.3 bit/s/Hz in throughput (or a 2% increase), which is mostly negligible.

Comparing the results with a system that uses ZFB for (N_R,M_T,K) = (1,2,10) and (1,4,10), it is shown that the suboptimal DPC GA performance is not worse than the optimal ZFB performance. In fact, the GA shows a gain of up to about 2 dB in Figure 3.6(c) compared with the optimal ZFB results. Therefore, the performance of our GA for DPC is also better than that of the GA for ZFB as in [109], since that GA must accordingly be no greater than the exhaustive search for that scenario.

Lastly, it can also be observed that there is an increase in throughput as N_R, M_T, or K increases, as expected from the corresponding increase in spatial or multiuser diversity. The largest gain in throughput comes from increasing M_T, because doing so increases both the number of spatial channels and / or streams that can be transmitted, and the maximum number of users that can simultaneously be scheduled. This gain in throughput

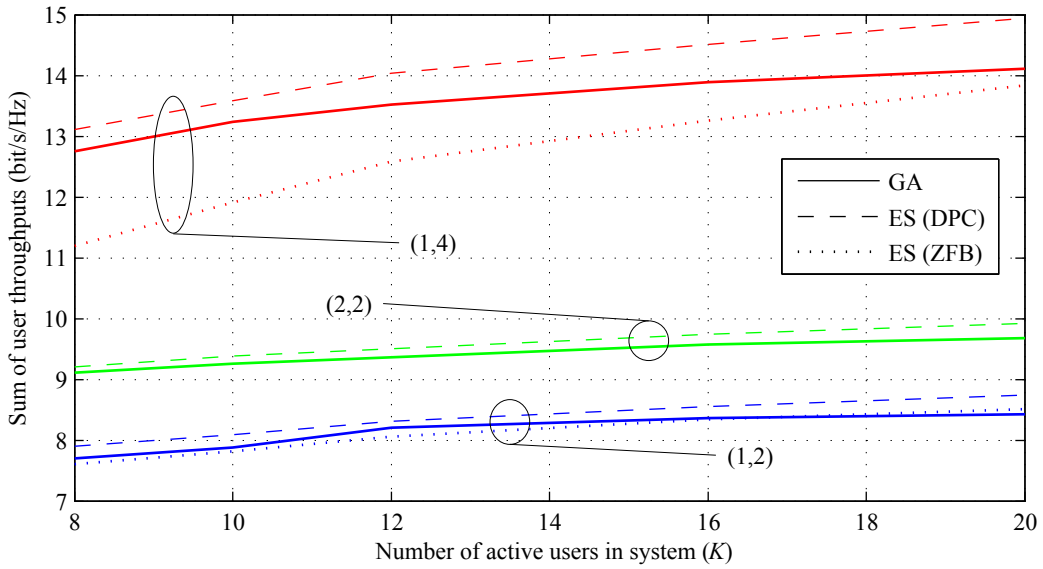

Figure 3.7: Performance of maximum throughput scheduling at SNR = 10 dB versus the number of active users for an (N_R,M_T) single-carrier MIMO system implemented via GA and ES.

comes despite the fact that the available transmit power is split among a larger number of users, thereby decreasing the individual capacities of those users.

Figure 3.7 shows the maximum throughput performance versus K at an SNR of 10 dB and better illustrates the effect of multiuser diversity. As the number of active users increases, so does the sum-throughput. However, this gain comes at the cost of increased complexity at the base station, both in terms of determining the users' channels and the function evaluations of the scheduler. For the former, having more users equates to more feedback being required in order for the base station to have channel state information available for each of the users in order to make its scheduling decisions. The issue of feedback is beyond the scope of this work; we simply assume that the base station somehow has perfect knowledge of the channels. More important for this work is the latter case: as the number of users increases, eventually, either the GA population or the number of generations must also increase to compensate for the larger number of possible permutations of scheduled users (i.e., the larger number of potential optimization solutions). This directly results in additional utility function evaluations and, hence, more complexity within a given scheduling interval. We discuss the issue of complexity in more detail later in Section 3.5.4.

Figure 3.8 shows the performance of the proportional fairness scheduler versus SNR in terms of the utility function value U_{PF} achieved. Note that the values are negative, because the average rate per user is, in general, less than 1 bit/s/Hz, which yields a

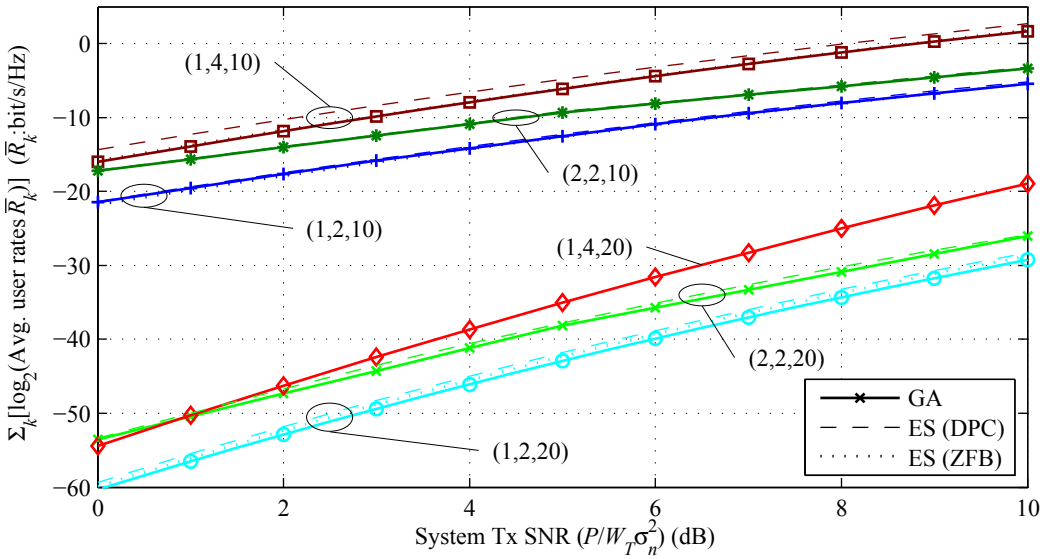

Figure 3.8: Performance of proportionally fair scheduling versus SNR for an (N_R,M_T,K) single-carrier MIMO system implemented via GA and ES.

negative value for the logarithm. Complexity and time considerations prevented us from performing an exhaustive search for the case of (N_R,M_T,K) = (1,4,20). Each Monte Carlo run of the PF simulation first requires some initialization time for the average throughput for each user to reach its approximate steady-state value; each average rate is initialized to a low value, which then increases and converges to around the steady-state as users are scheduled. Additional scheduling instances are then required once the steady-state is reached; it is only these later samples which are useful for the simulation output. (In comparison, the maximum throughput simulations require much less time, as they do not require average statistics to be built up; the MT criterion only uses the instantaneous channel states at each scheduling interval.) This reason, combined with the combinatorially increasing complexity of the exhaustive search, makes the time for an ES simulation in the (1,4,20) case prohibitively large.

It is again shown that the GA and an ES yield very similar results. In several cases, the plots for the exhaustive search and the GA almost overlap. As with the MT criterion, the PF performance of the GA, in terms of the utility function value achieved, is inferior to the optimal solution by at most approximately 0.5 dB. Interestingly, the performance of our GA is also approximately equal to the optimal performance of an ES when using ZFB.

Figure 3.9 illustrates the distributions of average rates $\bar{R}_k$ and instantaneous scheduled sum rates $\sum_{\forall k} R_k$ that the different scheduling methods with the PF criterion

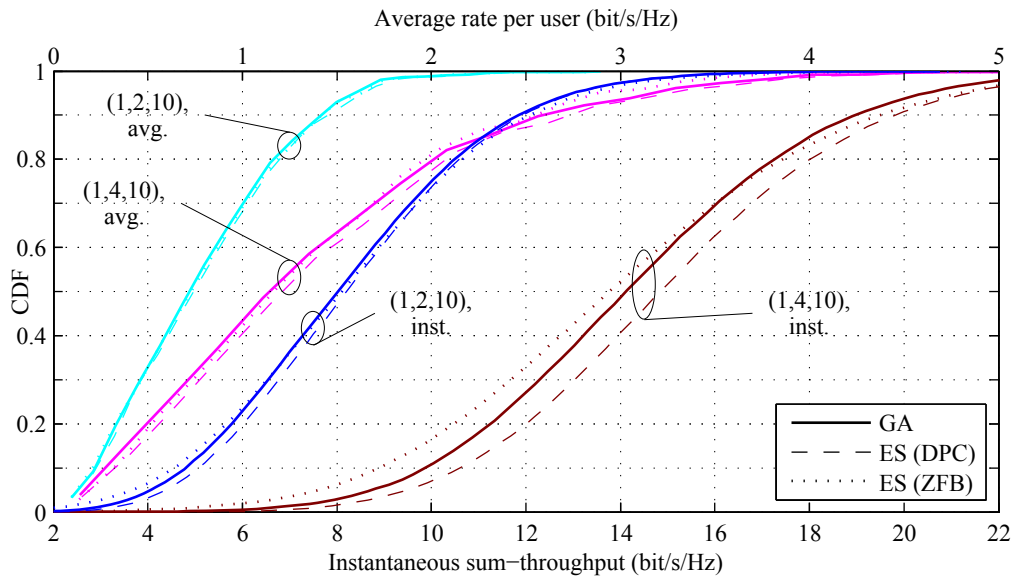

Figure 3.9: Distributions of average rate per user and instantaneous sum-throughput for PF scheduling in an (N_R,M_T,K) single-carrier MIMO system at SNR = 10 dB.

achieve. We consider the cases of (N_R,M_T,K) equal to (1,2,10) and (1,4,10) at an SNR of 10 dB. Comparing the sum-rate curves, the GA curves have basically the same shape as the exhaustive DPC curves but are slightly shifted to the left. This shift is about 0.2 and 0.74 bit/s/Hz for the (1,2,10) and (1,4,10) cases, respectively. Equivalently, this is on average about 0.1 to 0.185 bit/s/Hz per user for the two cases. A similar but smaller shift is shown in the average rate curves (about 0.02 and 0.1 bit/s/Hz, respectively). Comparing the GA results with the ES curves for ZFB, the (1,4,10) case is particularly noteworthy. The median of the average rates per user is approximately the same for the ZFB curve as it is for the GA curve, both being approximately 1.15 bit/s/Hz. However, the ZFB curve is slightly steeper; i.e., with ZFB, there is a smaller range of likely average rates. Looking at the instantaneous sum-rate curves, the ZFB curve generally displays a smaller sum rate than the GA curve, particularly below the 70th percentile. In other words, the instantaneous sum-throughput for an exhaustive search with ZFB will be lower than that for the GA and DPC about 70% of the time. Thus, between the two sets of curves, it can be stated that the optimal performance under ZFB is, perhaps, slightly more fair than the suboptimal DPC GA in terms of the variation of average rates across the set of active users, but this fairness comes at the expense of a significantly lower system capacity. This is in spite of the fact that both cases achieve about the same average utility function value, as seen in Figure 3.8.

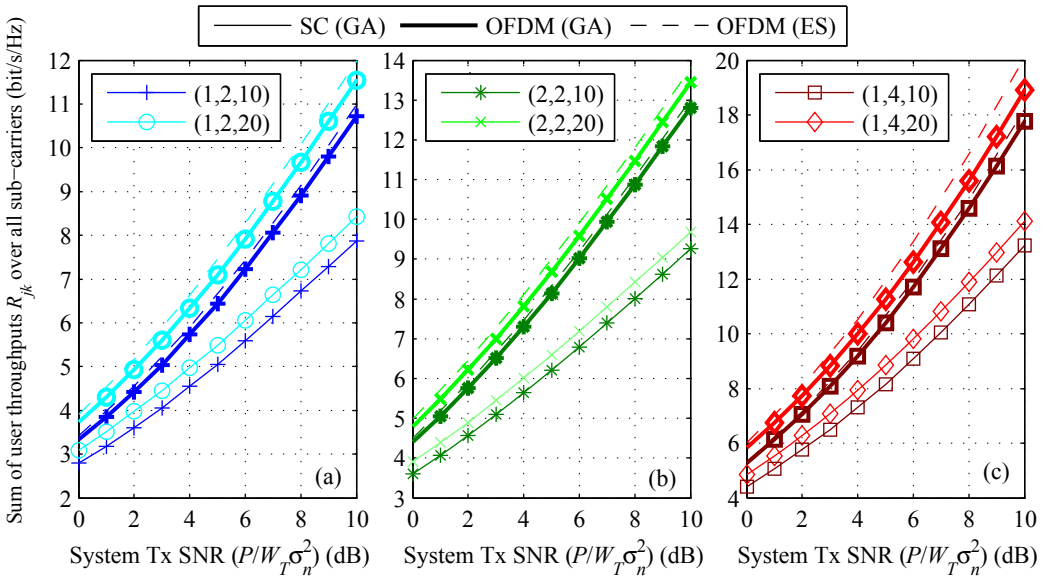

Figure 3.10: Performance of maximum throughput scheduling vs. SNR for an (N_R,M_T,K) multi-carrier MIMO system implemented via GA and ES. (a) $N_R = 1$, $M_T = 2$, and $K = 10, 20$. (b) $N_R = 2$, $M_T = 2$, and $K = 10, 20$. (c) $N_R = 1$, $M_T = 4$, and $K = 10, 20$.

3.5.2 Multi-Carrier Results

Figure 3.10 shows the performance of the GA implementing the maximum throughput criterion in a multi-carrier OFDM system relative to an ES. The single-carrier GA performance is also shown for reference. The multi-carrier system occupies the same total useful bandwidth W_T as the original single-carrier system, so that the SNR and throughput of the single- and multi-carrier systems can be properly normalized and compared. As in the single-carrier case, the throughput that the GA achieves is approximately 94–99% of that of the ES or, in terms of SNR, again about 0.5 dB worse. Thus, the extension of the GA to the multi-carrier scenario works just as well as with a single carrier. Furthermore, there is a notable increase in spectral efficiency that results from the use of OFDM, as a result of the subcarriers overlapping and, hence, more efficient use of bandwidth. The system sum-capacity relative to the single-carrier scenario increases by a factor of about 1.2 at an SNR of 0 dB, up to a factor of about 1.36 at 10 dB. In terms of SNR, the gain is about 1.5 to 3 dB. We have also considered the case when an ordinary FDM system is used (with non-overlapping subcarriers). In this case, the total bandwidth W_T is divided equally into four subcarriers, each with a bandwidth of $W_C = W_T/4$. Interestingly, the GA performance is this case is virtually indistinguishable from that of the single-carrier GA (thus, we do not plot it on the graphs). This implies that the performance of the GA per (sub)carrier is identical when

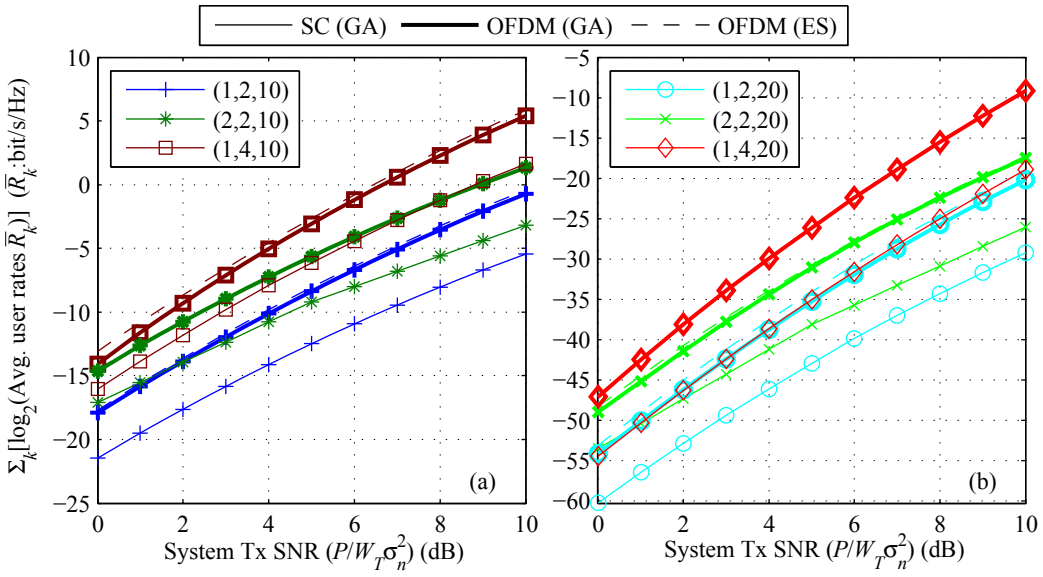

Figure 3.11: Performance of proportionally fair scheduling versus SNR for an (N_R,M_T,K) multi-carrier MIMO system implemented via GA and ES. (a) $K = 10$, $N_R = 1$, 2, and $M_T = 2$, 4. (b) $K = 20$, $N_R = 1$, 2, and $M_T = 2$, 4.

the available transmit power is divided equally among the subcarriers. Thus, the performance of the GA should scale to even larger numbers of subcarriers, though at the expense of added scheduling complexity. (We discuss the issue of complexity in more detail in Section 3.5.4.) With that added complexity, even further gains in the OFDM case likely could be obtained if even more subcarriers were used, due to the increased bandwidth efficiency η of OFDM, given by $\eta = N_C/(N_C + 1)$ [13], if the guard interval is ignored. (The bandwidth efficiency of FDM is the same as for single-carrier transmission.) Lastly, we note increases in throughput with increasing M_T, N_R, and K just as in the single-carrier case.

Figure 3.11 shows the GA performance for the proportional fairness criterion in the multi-carrier case. Once again, the GA performance is, at most, approximately 0.5 dB inferior to that of the ES. A gain in spectral efficiency for the OFDM case relative to the single-carrier one, similar to the gain for the maximum throughput criterion, can again be noted with the proportional fairness criterion. This indicates that users are receiving higher average throughputs (in bits per second per hertz) than in the single-carrier case.

In theory, the multi-carrier system can simultaneously schedule up to $N_C \times M_T$ different users, provided the SNR is sufficiently high, as $N_S = M_T$ users can be scheduled on each subcarrier. The likelihood of this occurrence increases with K due to the corresponding increase in multiuser diversity. A larger user pool increases the likelihood of a different set of users providing the largest scheduling metric on each subcarrier. Our

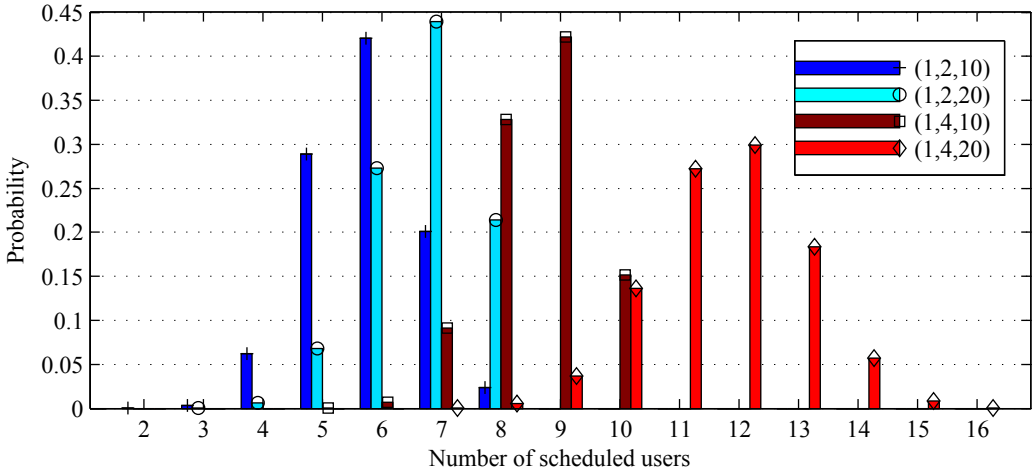

Figure 3.12: Distributions of number of simultaneously scheduled users under the proportional fairness scheduling criterion at SNR = 10 dB for an (N_R,M_T,K) multi-carrier MIMO system.

simulations show that the system indeed tends to simultaneously schedule large numbers of users. Figure 3.12 shows the distributions of the number of scheduled users for the proportional fairness criterion at an SNR of 10 dB. As an example, for the (N_R,M_T,K) = (1,2,10) case, the system usually schedules between five to seven users out of the eight maximum possible users. When K increases to 20, this increases to most often between 6 and 8 users.

This result has two main implications. First, each user is most often assigned to just one or, on occasion, two subcarriers. Hence, their instantaneous throughput (in bits per second) is lower than in the single-carrier case, because their assigned bandwidth is lower. Second, the delays of each user also decrease. More users are simultaneously scheduled in the multi-carrier case; hence, any particular user would be scheduled more often than in the single-carrier case. The CDF of the head-of-line delays per user (that is, how many transmission scheduling intervals elapse between a given user being selected, then being selected again) is shown in Figure 3.13. Unsurprisingly, our simulations indicate that, on average, because the multi-carrier system has four subcarriers, the delays per user are approximately four times smaller than in the single-carrier system. The user delays decrease; thus, the average user throughput increases, which helps in compensating for the reduction in instantaneous throughput. As a side comment, we also note scaling N_R has no effect on the delays, nor does simultaneously scaling K and M_T by the same factor.

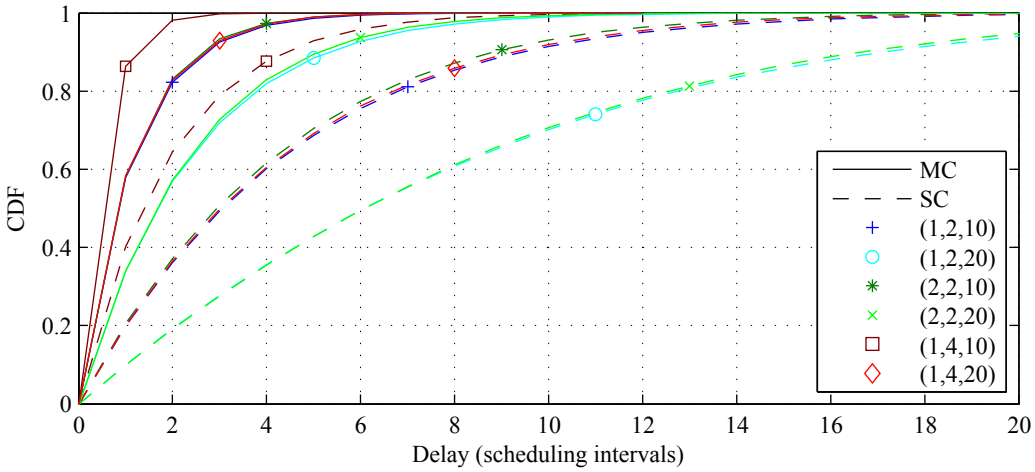

Figure 3.13: Distributions of head-of-line delays per user under the PF criterion at SNR = 10 dB for single- and multi-carrier transmission and various values of (N_R,M_T,K).

3.5.3 Convergence

Because of the random nature of the selection and breeding of the GA, the convergence of the GA is also a stochastic process. Hence, it is possible that a GA will not find a globally optimal solution for a utility function, particularly for a fixed number of iterations. The size of the search space obviously also affects the likelihood of finding the global optimum. The probability of not finding the optimum solution, of course, decreases the longer the algorithm is allowed to run. In addition, if elitism is not employed, the algorithm can, in fact, diverge from the optimum if a previously good solution is lost. With the use of elitism, the utility function value will strictly be non-decreasing with the number of generations. Nonetheless, despite the chance of not converging to the global optimum, GAs are known for quickly finding a good solution to optimization problems. Hence, if that solution is sufficiently good, in many cases, it is not necessary to wait for the algorithm to find a better solution.

Figure 3.14 shows the convergence of the GA with the number of generations for the maximum throughput criterion at an SNR of 10 dB. The figure shows, on average, how far from the optimal utility function value the algorithm is. As already seen in Figure 3.6, the convergence is largely independent of SNR; the GA achieves approximately the same proportion of the ES regardless of the SNR. We have also seen that the convergence under the proportional fairness criterion is quite similar, because the size of the search space is identical. Hence, we can focus solely on the maximum throughput criterion at one specific SNR when examining the convergence in more detail. We set $N_p = 20$ for $M_T = 4$ and $K = 20$, and $N_p = 10$ for all other cases.

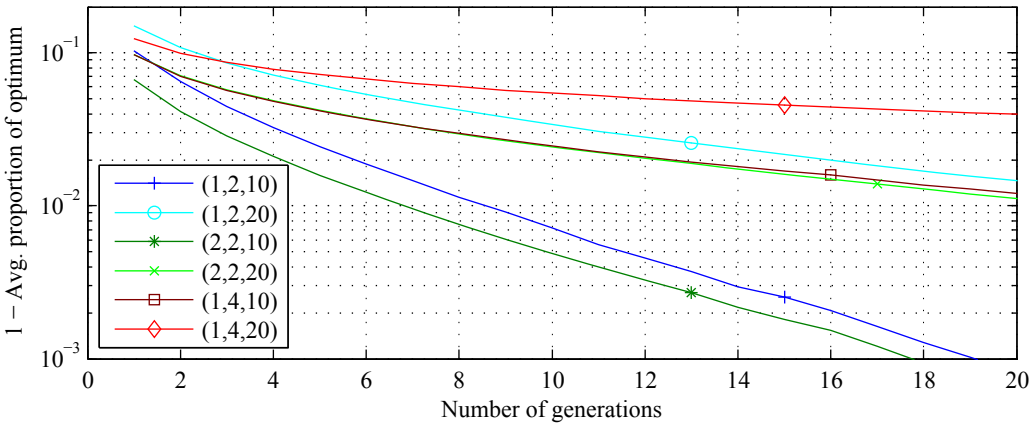

Figure 3.14: Average convergence of GA versus generations for the maximum throughput scheduling criterion at an SNR of 10 dB and various values of (N_R,M_T,K).

Figure 3.14 shows that initially the GA rapidly converges towards the optimum utility function value. After just a couple of generations, the algorithm achieves, on average, about 90% or more of the optimal value. (Recall also that generation 1 is the initial, random population.) As the algorithm progresses, the rate of convergence slows. In part, this is due to the algorithm, in some cases, prematurely halting on a local optimum point. In these cases, the mutation operation helps the algorithm become "unstuck" and consider other possibly better solutions. In other cases the rate of convergence slows, simply because the algorithm has already converged to an optimal or near-optimal solution for the scheduling problem.

It can be seen that the number of receive antennas per user has very little effect on the convergence of the DPC genetic scheduling algorithm. It can also be seen that scaling either K or M_T has similar effects on the convergence, as seen by the closeness in performance in the lines for (N_R,M_T,K) = (1,2,20) and (1,4,10). The figure furthermore demonstrates that for M_T = 2 and K = 10, 5 generations is sufficient for the algorithm to converge to within about 99% of the optimal value on average. For the remainder of the cases, 10 generations is a sufficient time for the algorithm to run. Thus, these values for N_g are used throughout this chapter.

Interestingly, it is not simply the number of generations the algorithm runs that affects the overall convergence, but rather the product $N_p \times N_g$, which is the total number of times the GA calculates the utility function. To demonstrate this, Figure 3.15 compares two cases each of the performance of the GA at (N_R,M_T,K) = (1,2,20) and (1,4,10). In the first case, N_p = 10 and N_g = 10 as used throughout the rest of this chapter, whereas in the second, N_p is doubled to 20 and N_g is halved to 5. The product $N_p \times N_g$ thus remains 100 in

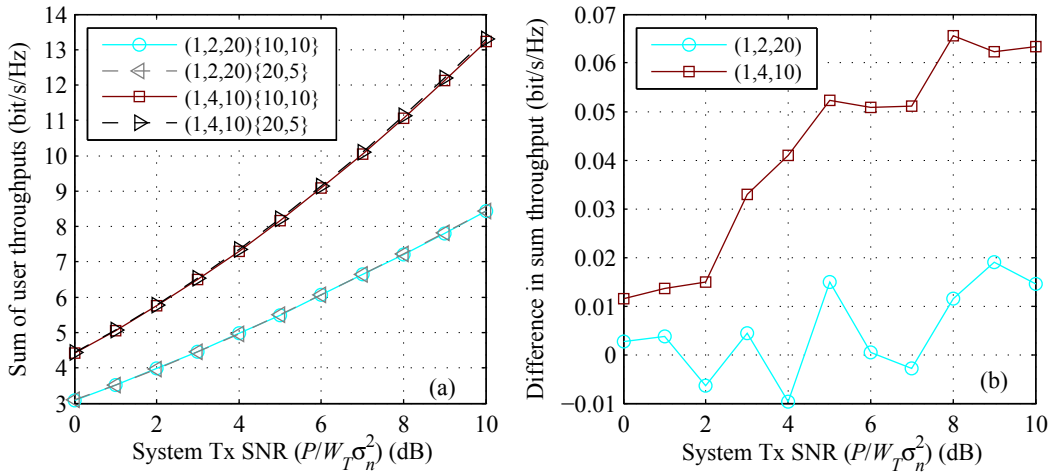

Figure 3.15: (a) Performance of single-carrier MT GA scheduling vs. SNR for (N_R,M_T,K) = (1,2,20) and (1,4,10), each with $\{N_p,N_g\}$ = {10,10} and {20,5}. (b) Difference in sum-throughput for $\{N_p,N_g\}$ = {20,5} compared to {10,10}.

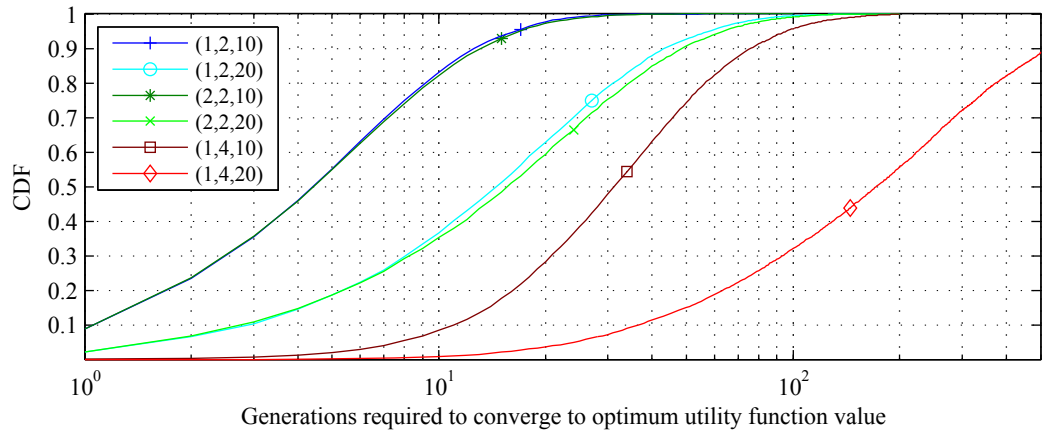

Figure 3.16: Distributions of number of generations required for GA convergence to the optimum utility function value for the maximum throughput scheduling criterion at an SNR of 10 dB and various values of (N_R,M_T,K).

all the cases. It is seen in Figure 3.15(a) that the throughput is virtually identical for both cases of N_p and N_g. Figure 3.15(b) confirms this, showing the maximum difference between the average throughput of the two cases is no greater that 0.07 bit/s/Hz, which is insignificant and likely within the error range of the Monte Carlo simulation.

Figure 3.16 shows distributions of the number of generations required for the GA to find the globally optimum utility function value (i.e., the optimum selection of users and their encoding order). As expected, the larger the search space, the more generations are required. However, as Figure 3.14 implies, even in the cases where the algorithm has not found the optimal value, the value is usually quite close to the optimum. Based on the two figures, it can be determined that, for those cases that have not converged to the

globally optimum value after the maximum number of generations has been reached (i.e., $N_g = 5$ generations for $M_T = 2$ and $K = 10$, and $N_g = 10$ generations otherwise), the GA has found a solution that, on the average, is within about 4–6% of the optimum value. Thus, as we have previously stated, although it may take the GA some time to find the overall optimum solution, it can quite rapidly find a near-optimum solution.

In practice, it would be quite unlikely to know ahead of time what the globally optimal utility function value is, and thus how far the GA is away from it. Thus, it is hard to determine how many generations the GA should run for. We saw positive results in this work with $N_p \times N_g$ scaling proportional to N_S and K. However, other stopping criteria could also be used in practice. For instance, if there is some sort of scheduling deadline that exists in the system, the GA could simply run until that deadline is reached, then output its best decision. Alternatively, the GA could monitor the utility function value; if the value stalls for a number of generations past some threshold, the GA could then exit. If the system keeps statistics involved in past scheduling decisions, these could also be incorporated in deciding when to quit. For example, it could operate for only x generations more once a threshold, based on the past utility values, is reached.

3.5.4 Runtime / Complexity Comparison

Table 3.1 compares the runtime of the genetic algorithm to an exhaustive search in terms of the number of utility function evaluations required for each. The utility function that is evaluated is identical for both an ES and the GA, as given by (3.13) and (3.15), with the optimal transmit covariance matrices determined by the methods in [107] and [82], respectively. The table applies to both the single-carrier and multi-carrier scenarios; however, it should be noted that each individual utility function evaluation is N_C times more complex in the multi-carrier scenario than its single-carrier equivalent. (Essentially, the single-carrier function is evaluated once per subcarrier, because each subcarrier is independently scheduled.) In addition, note that the number of evaluations is independent of the number of receive antennas at the mobiles. However, as the number of receive antennas increases, the larger channel matrices means that each evaluation will require more computations.

The number of function evaluations used by our GA is larger compared to the work in [109] in the context of ZFB. This is a direct result of the larger search space that results from the use of the order-dependent encoding of DPC. Hence, additional function evaluations are required to compensate for that larger search space. In general, we find

Table 3.1: Runtime comparison of genetic and exhaustive search scheduling algorithms in terms of number of utility function evaluations required

(N_S,K)	Genetic Algorithm ($N_p \times N_g$)	Exhaustive Search (General Case Utility Function)	Exhaustive Search (Proportional Fairness)	Exhaustive Search (Maximum Throughput)
(2,10)	10×5 = 50	100	55	57
(4,10)	10×10 = 100	5860	385	409
(2,20)	10×10 = 100	400	210	212
(4,20)	20×10 = 200	123520	6195	6219

that doubling either the number of generations or the population size for the GA relative to that in [109] yields good results. (Recall from the previous section that doubling either is equivalent for the performance and convergence.)

The final two columns in Table 3.1 are applicable only for the proportional fairness[1] and maximum throughput criteria. Knowledge of the form of their utility functions can be exploited to reduce the complexity of an ES. In the case of the proportional fairness criterion, the utility metric is a weighted sum rate, where the weight of each user is the reciprocal of their average throughput. Because of this, and since the MAC capacity is convex, the maximization of that metric will lie on the boundary of the MAC capacity region, and thus also on the boundary of the DPC capacity region, as the two are coincident. Furthermore, the optimal decoding order on the MAC is known *a priori* as being the relative increasing order of the values of the user weights [82]. In other words, the larger the weight of a given user is, the later in the decoding order on the MAC should that user be positioned, with the highest weight being positioned last. This is intuitively explained. Users in later positions in the decoding order experience less interference overall, because the effect of users earlier in the ordering is removed. The user that is decoded last experiences no interference at all. Thus, it is logical to put the user with the largest weight in that position. The rate that is achieved by that user shall be the largest possible compared to any other order, and hence, its effect on maximizing the weighted sum rate would be the greatest. With the transformations in [24], this means that on the BC, the user with the largest weight should be encoded first. Thus, with the optimal decoding / encoding order known beforehand, it is therefore only necessary for the ES to search through all combinations of users (Eqn. (3.1)) instead of all ordered selections (Eqn. (3.2)), thereby reducing the complexity.

[1] Strictly speaking, the column for PF also applies to any utility function in the form of a weighted sum rate, provided that the weights do not depend on the instantaneous user rates.

In the case of the maximum throughput criterion, as previously mentioned, the maximum sum rate on the MAC and BC does not depend on the decoding / encoding order. Although the individual user rates change depending on the order, any given encoding order will result in the same value of the utility function. Choosing a specific order, if necessary, is therefore based on secondary criteria. This fact can be used to reduce the complexity of the ES. The search can first seek through all possible combinations of users to find the specific group that maximizes the utility function. Once that group is found, it can then search through all possible orderings of just those users to break the tie.

The "general case" column of the table refers to some arbitrary utility function. In this general case, it is assumed that the utility function is such that the encoding order affects the function value, and there is no easy way to know beforehand what the optimal encoding order is. One such example might be if the utility function is a non-linear function of the user rates. In such a case, the exhaustive search must indeed search through all possible selections and orders of users.

The runtime of the GA is significantly less dependent on the number of active users K, the number of transmit antennas M_T, and the number of scheduled users N_S. In particular, decent results were obtained when the number of function evaluations increased linearly with N_S and approximately linearly with K. However, it is important to note that, based on the results in Figure 3.7, the rate at which the sum-throughput increases with the number of active users is not as large for the GA as it is for the ES. This fact indicates that the number of function evaluations in the GA needs to increase at a rate greater than linearly with K to match the curvature of the ES curve. This could be accomplished either through an increased population size or a larger number of generations. The more the function evaluations that the GA performs, the closer the performance will be to that of an ES. Hence, as more generations or a larger population size would likely be required with larger pools of users to maintain the current level of performance, additional complexity for larger K would result.

There are, however, some steps that could be taken to reduce that complexity in practice. For example, it may be possible to determine beforehand one or two users that are particularly likely to be scheduled. This is quite reasonable, since we already assume that the base station has channel state information with which to make scheduling decisions. The likely candidates may include a certain user with a channel that is consistently known to be good (e.g. with low path loss or shadowing, or a channel matrix

with a high norm). Alternatively, this could be a user who has not been scheduled for a while and may be approaching some maximum delay or minimum throughput threshold. In such cases, the initial population could be seeded with chromosomes that include those users. This would result in a better starting location for the search and, hence, less time to converge, thus improving the GA performance. A similar situation would exist if the channel gains were correlated between adjacent scheduling instances. While we assumed in our simulations that the channel gains were independent between scheduling intervals, in practice, there will be some temporal correlation in the physical channel[1]. In such a case, the optimal scheduling solution might not significantly change between the decision intervals. Hence, the initial population for a given scheduling instance could be seeded with some of the best chromosomes from the previous instance, again leading to a reduced convergence time and improved performance.

Nevertheless, for the analyzed scenarios, a simple linear dependence of the GA runtime on K and N_S already yields throughput results close to that of the ES and without its associated combinatorial complexity. In fact, if the change in the number of active users is relatively small (e.g. a few users join or leave the system), the number of function evaluations can likely be kept constant with no significant effect on the GA performance.

There is a large runtime reduction for the GA compared to the optimal algorithm. For $N_S = 2$ and $K = 10$, there is not much of a reduction to be seen relative to an ES with the MT and PF criteria. However, for a general case utility function, the runtime with the GA is reduced by half. Increasing K to 20, the GA runtime is about half that of the MT and PF exhaustive searches, and a quarter of that in the general case. With $(N_S,K) = (4,10)$ and $(4,20)$, there is a runtime reduction by a factor of about 4 and 31, respectively, for both the maximum throughput and proportional fairness criteria. Compared with a general case ES, the runtime is reduced by a factor of 58.6 and 617.6, respectively.

3.6 Conclusion

In this chapter, we have investigated the use of genetic algorithms for scheduling in multiuser single-carrier and multi-carrier MIMO systems with DPC. We have described a GA representation that can account for both the selection of users and the encoding order of those users. This representation can be used just as easily in any system with linear or non-linear precoding where the encoding order of the users will affect the users'

[1] We can neglect the effect of correlation in this work, since its presence would not affect the comparison between the scheduling methods.

performance and / or the utility function. For the case of DPC, the representation is also independent of the number of receive antennas at each user.

We considered a system with a base station with M_T transmit antennas, which schedules transmissions using DPC to a pool of K active users, each having N_R receive antennas, and analyzed the performance of the GA relative to the optimal performance of an exhaustive search. It was observed that the GA performed quite well, obtaining on average about 94–99% of the optimal utility function value with the maximum throughput scheduling criterion. In terms of SNR, the GA was approximately 0.5 dB inferior to the ES. The relative performance of the GA to the ES was basically independent of both the system SNR and the number of receive antennas at each mobile. The results for the proportional fairness scheduling criterion were much the same, with the GA again being about 0.5 dB away from optimal.

Extending the analysis to a multi-carrier scenario, the relative performance of the GA was basically the same as with a single carrier. The use of OFDM with four subcarriers resulted in a gain in sum-throughput by a factor of about 1.2 to 1.36 for MT scheduling due to OFDM's increased bandwidth efficiency compared to single-carrier transmission. For both MT and PF scheduling, the performance gain was about 1.5 to 3 dB when using OFDM. The multi-carrier system also was seen to schedule more users simultaneously, leading to decreased packet delays overall. In our multi-carrier work, we simplified the scheduling optimization problem by allocating equal transmit power to each of the subcarriers. It may be possible to further improve the performance by performing a joint scheduling and power optimization over the subcarriers instead of using equal power allocation.

Examining the convergence, it was shown how quickly the GA converged to a very good solution (although not necessarily the global optimum). For the cases examined, a maximum of 10 generations was sufficient to obtain good results. This short convergence time also resulted in a greatly reduced runtime compared to the ES. The required calculations were reduced by a factor of up to 31 for the two examined scheduling criteria, and up to a factor of 617.6 for a more general case utility function and exhaustive search.

Despite the very good performance of the GA seen in this chapter, it is possible to further improve its performance by the careful tuning of some of the aspects of the genetic algorithm. The next chapter examines this tuning in more detail.

Chapter 4

Impact on and Improvement of the Convergence of GA Scheduling from Parameter Tuning and Change in Crossover Method

4.1 Introduction

It was seen in the previous chapter that our genetic algorithm for scheduling in MIMO systems with DPC performed quite well relative to an exhaustive search. That GA was based in part on a genetic algorithm used in [109] to perform scheduling in a system that used ZFB. As such, we adapted some of the concepts in [109] to our GA scheduling algorithm. One of the most notable of these was an adaptive mutation rate. As already mentioned, the parameters used within [109] still provided good performance in the context of DPC scheduling. However, it also raises the question: Are those parameters used for ZFB still the best overall to be used with DPC? Or, would a change in some of the values of those parameters lead to a further improved performance? In this chapter, we examine this issue and demonstrate that tuning the parameter values within the adaptive mutation rate can indeed lead to a significant improvement in the convergence of the DPC genetic algorithm. The average number of iterations required by the GA to converge to a given percentage of the optimum utility function value can be reduced to less than a third of that required by the original parameter values. We also briefly examine the impact of using a different crossover method than the one-point crossover used in the previous chapter. Our contributions for this chapter have appeared in [125], [126].

4.2 Problem Description

In the previous chapter, we used an adaptive mutation rate in our GA, which we repeat here for convenience:

$$p_m = \frac{1}{\beta_1 + \beta_2 \, \sigma_G / \mu_G}. \tag{4.1}$$

In the above equation, σ_G and μ_G are the standard deviation and the mean of the fitness for the population of chromosomes during each generation, while β_1 and β_2 are constants. Earlier, we used the values of $\beta_1 = 1.2$ and $\beta_2 = 10$.

In this chapter, we examine more closely the effect of the parameter values for β_1 and β_2 in (4.1) on the convergence of the genetic algorithm. The initial motivation for this work first came from an examination of the mutation rates seen during the operation of our GA in the previous chapter. We observed that the rate upon the random initialization of the population tended to be in the range of 0.2–0.6. Higher values for p_m were seen for a larger number of active users K and with a larger number of transmit antennas M_T (or equivalently, the maximum number of scheduled users N_S). Distributions of the mutation rate for the initial population of our GA are shown in Figure 4.1. The case of an SNR of 10 dB is shown. We also examined the case of 0 dB; in that case, the curves look essentially the same, except they are shifted to the left on the x-axis by about 0.05 units.

These values are somewhat higher than what is typically seen in a GA. A mutation rate about an order of magnitude lower (e.g. around 0.01) is more common [112],[127]. When combined with elitism, a higher mutation rate is not as much of a problem, as there is no danger of destroying the previous best solution. However, too high of a mutation

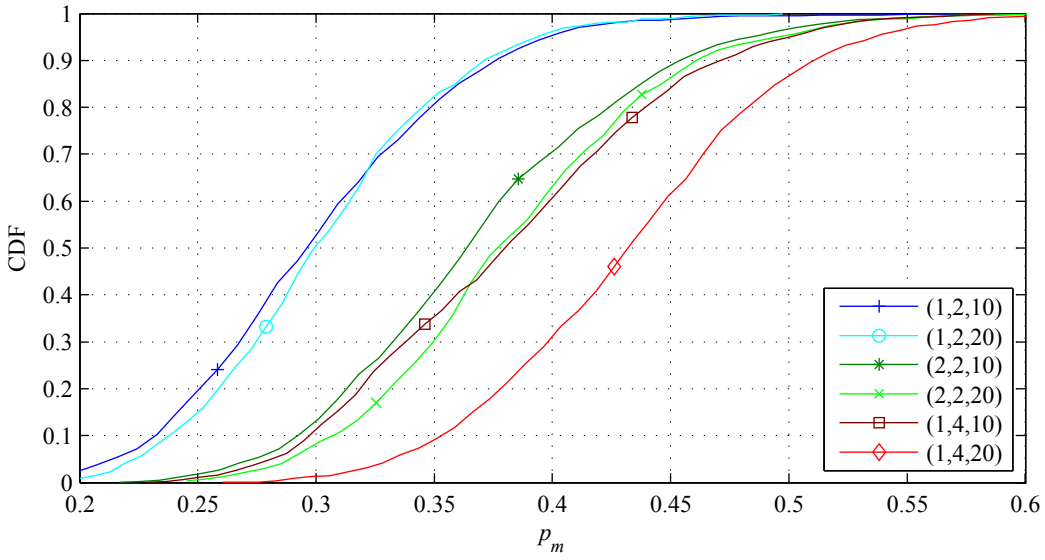

Figure 4.1: Distributions of the adaptive mutation rate for the random initial first generation of the genetic scheduling algorithm for DPC with $\beta_1 = 1.2$ and $\beta_2 = 10$, an SNR of 10 dB, and various values of (N_R,M_T,K).

rate can still slow the convergence of a GA, due to the destruction of good solution characteristics in the chromosomes that were obtained during previous generations.

In summary, we suspect these adaptive mutation rates are likely too high. Thus, we generally are interested in increasing the parameters β_1 and β_2 to create a lower mutation rate, and analyzing the effect on the convergence of the GA.

4.3 Simulation Setup and Results

4.3.1 Simulation Setup

The simulation setup and system model in this work are nearly identical to that in the previous chapter. We still consider a base station with M_T transmit antennas, a transmit power limitation of P, and perfect channel knowledge. The base station schedules transmissions using capacity-achieving DPC to a pool of K users requesting service, each with N_R receive antennas. The users experience statistically identical path loss, noise, shadowing, and Rayleigh block fading conditions. However, we only consider a single-carrier system now, as it was seen in the previous chapter that the single-carrier and multi-carrier GA performance relative to an exhaustive search were much the same.

The key difference in this chapter is that β_1 and β_2 are varied instead of being kept constant, and the change in the convergence of the GA is examined. We focus primarily on the GA convergence with the maximum throughput scheduling criterion (i.e., the utility function $G_{MT} = \sum_k R_k$) at an SNR of 10 dB. However, we comment on other scenarios (i.e., convergence at different SNRs and under the proportional fairness scheduling criterion) in Section 4.4.2.

4.3.2 Simulation Results

Regardless of the specific value of N_R, M_T, or K, we can observe an overall trend in the simulation results as the values for β_1 and β_2 change. Notably, there is no single optimal operational point for either β_1 or β_2. Instead, there is an optimal range of values for both parameters where the average time for the GA to converge is both minimal and approximately equal. Outside of this range, smaller β values give too large of a mutation rate, while larger β values give too small a rate; either case increases the convergence time of the GA. In Figure 4.2 and Figure 4.3, we examine the effect on the convergence rate for (N_R,M_T,K) = (1,2,10), when one of β_1 or β_2 is varied while the other is kept constant. These figures show the probability of the GA having found the optimal utility

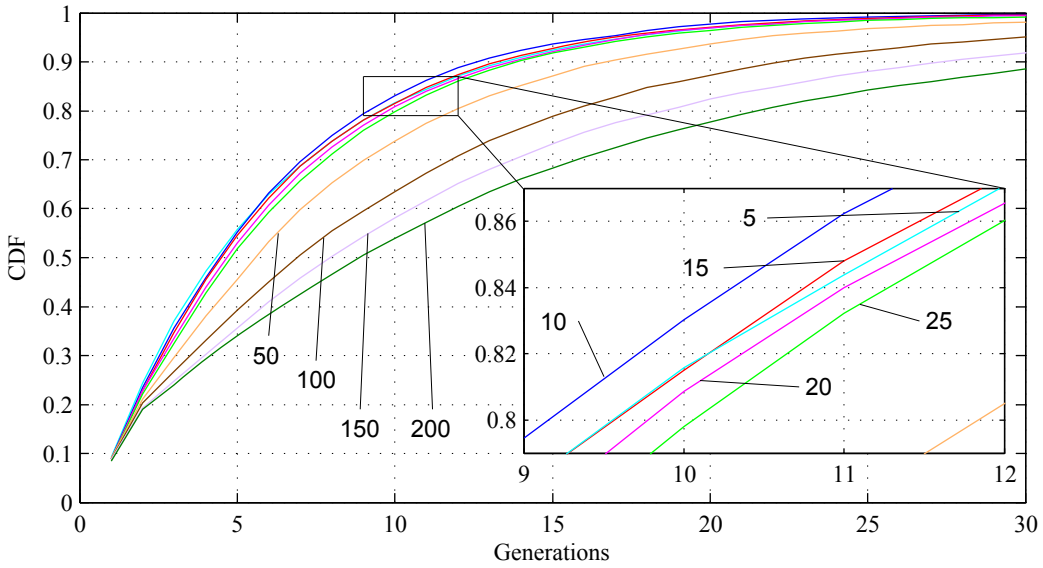

Figure 4.2: Distributions of number of generations required to converge to optimal utility function value for $(N_R,M_T,K) = (1,2,10)$ at an SNR of 10 dB; $\beta_1 = 1.2$ (constant), β_2 variable.

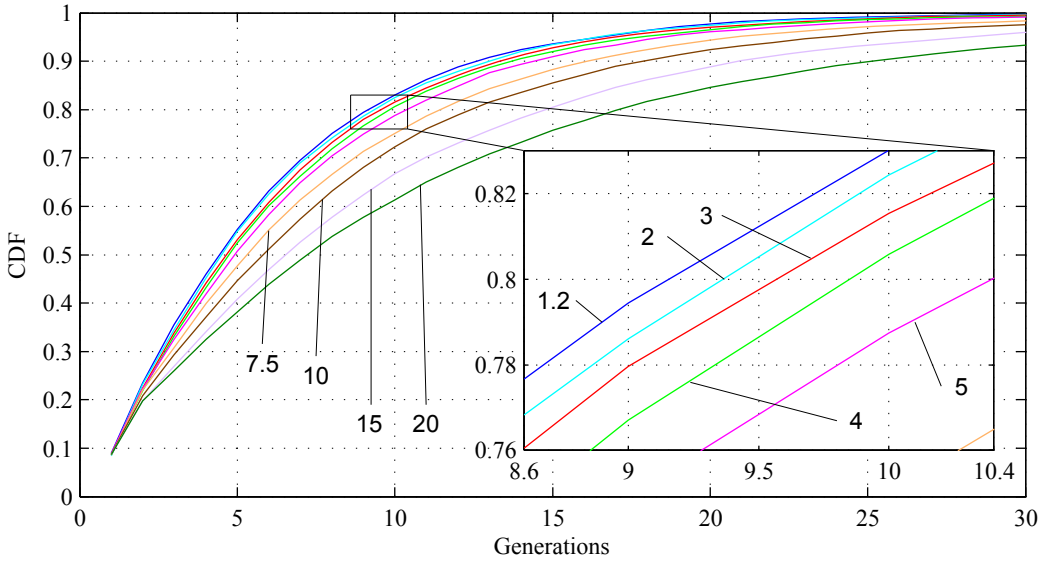

Figure 4.3: Distributions of number of generations required to converge to optimal utility function value for $(N_R,M_T,K) = (1,2,10)$ at an SNR of 10 dB; β_1 variable, $\beta_2 = 10$ (constant).

function value after the specified number of generations. The distributions in convergence time are very similar for β_2 = 5–25 with a constant $\beta_1 = 1.2$, and for β_1 = 1.2–4 while β_2 is constant at 10. Of the values tested, $(\beta_1,\beta_2) = (1.2,10)$, as we used in the previous chapter, results in the fewest generations required for the algorithm to converge to the optimal utility function value, but only by a small margin.

We also observe that the closer the algorithm is to the optimal solution, the more pronounced of an effect that changes to β_1 and β_2 have. We now consider the case of

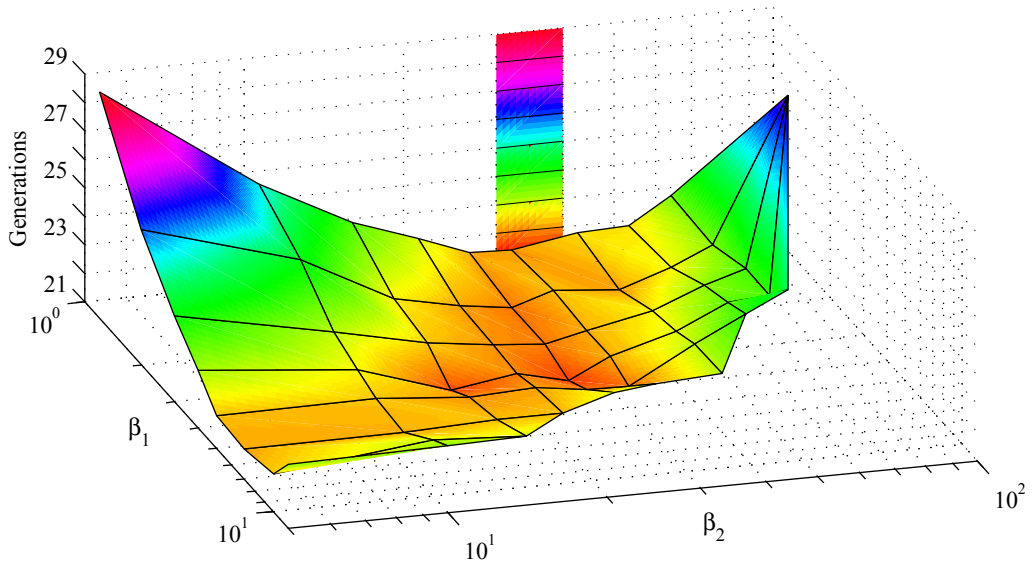

Figure 4.4: Number of generations required to converge on average to within 1% of optimal utility function value as a function of β_1 and β_2 with $(N_R,M_T,K) = (1,2,20)$ and an SNR of 10 dB.

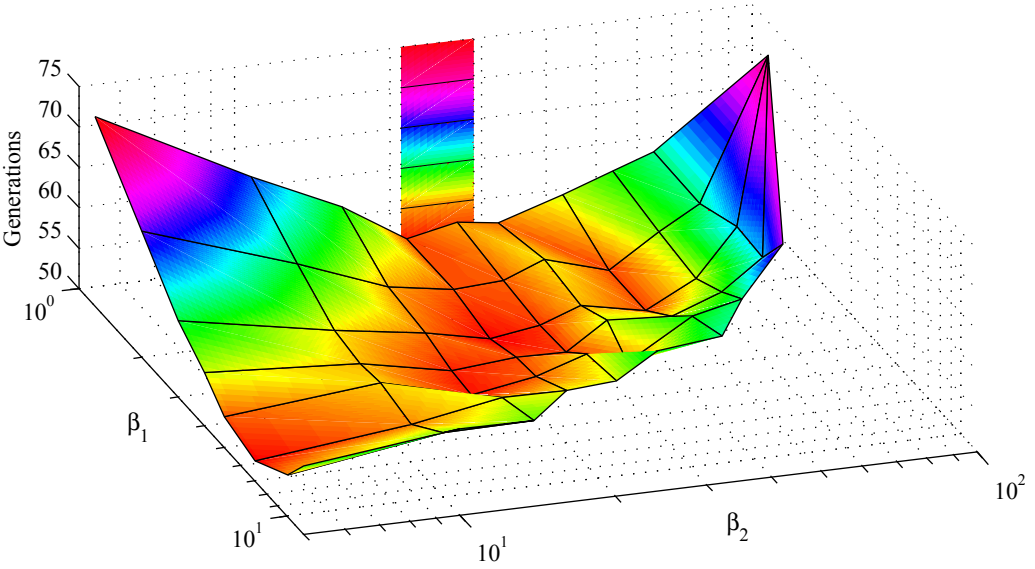

Figure 4.5: Number of generations required to converge on average to within 0.1% of optimal utility function value as a function of β_1 and β_2 with $(N_R,M_T,K) = (1,2,20)$ and an SNR of 10 dB.

$(N_R,M_T,K) = (1,2,20)$. Figure 4.4 shows the number of generations required for the algorithm to converge on average to within 1% of the optimal utility function value, while Figure 4.5 shows the number of generations to converge on average to within 0.1% of the optimal value. On an absolute scale, there is clearly a larger impact from changing the parameters on the algorithm convergence time at the 0.1% level than there is at the 1% level.

Specifically, in Figure 4.4, we vary the values of β_1 between 1.2 and 12 and β_2 between 5 and 100. Within this range, the average time to converge to within 1% of the optimal utility function value changes by about 7–8 generations from the longest time to the shortest. However, that same range for β_1 and β_2 causes a change of about 20–21 generations at the 0.1% convergence level in Figure 4.5. Our simulations indicate a further increase in the impact of the parameter values on the convergence time when the algorithm is even closer on average to the optimal solution. However, a more detailed observation past this point would be unwarranted and likely inaccurate, owing to an increasing lack of unconverged cases / samples (i.e., those that have not yet found the optimal value) in our Monte Carlo simulations. The simulations consider the average over 10000 independent runs. When considering convergence to 0.1% of optimum, on average about 390 of the runs have not yet converged to the exact optimum value within the generations shown in Figure 4.5. This number of runs is still statistically significant. In comparison, we could consider a convergence even closer to optimum, such as to within 0.01%. In that case, though, only about 40 runs on average have not converged. Having that few cases runs the risk of yielding an inaccurate average, so it is best not to consider closer convergence levels[1].

Interestingly, if we consider the relative change in the convergence time instead of the absolute difference, the level of convergence has much smaller of an effect. For instance, moving from the highest point on the graphs of Figure 4.4 and Figure 4.5 at $(\beta_1,\beta_2) = (1.2,5)$ to the lowest point yields a reduction in the average number of generations by a factor of about 0.72 at both levels. That is, the number of generations at the lowest point is about 0.72 times the number of generations at the highest point for both graphs. This fact can also be noted simply in observing the similar overall contour of the two figures.

We also note two additional observations from the two figures. First, the optimal value for (β_1,β_2) is no longer around (1.2,10), compared to the earlier case in Figure 4.3 with $K = 10$. Second, the region around the minimum values of the graph fluctuates somewhat. This is due to the stochastic way in which the GA converges. The algorithm

[1] A good rule of thumb, similar to when calculating bit or symbol error rates, is to have 100 statistical samples in order to be sure of a proper average. In the case of average bit or symbol error rate calculations, this means to have at least 100 bits or symbols that are in error out of however many are considered in total. In our case, this means to have at least 100 instances that have not in fact located the optimal selection of users to maximize the utility function, in order to find how far on average the GA is from the optimum utility function value.

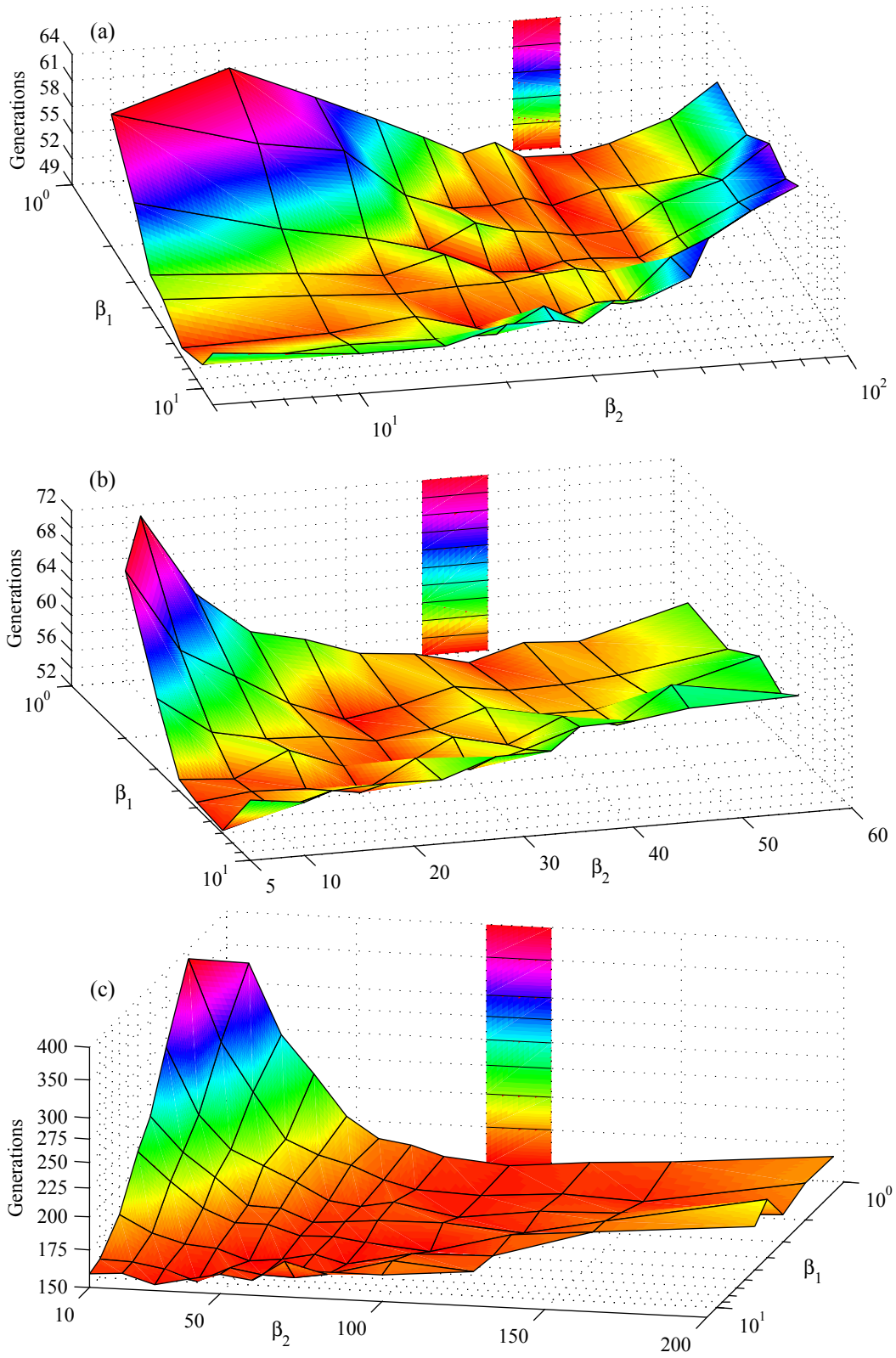

Figure 4.6: Number of generations required to converge on average to within 0.1% of optimal utility function value as a function of β_1 and β_2 with 10 dB SNR and various (N_R,M_T,K). (a) (N_R,M_T,K) = (2,2,20). (b) (N_R,M_T,K) = (1,4,10). (c) (N_R,M_T,K) = (1,4,20).

may occasionally get extremely "lucky" and find the optimal solution in just a few generations. At other times, it can be "unlucky" and not find the optimal solution even after many generations. These outliers skew the mean to a small degree, and the small fluctuations in the graph are the result.

The number of transmit antennas (or scheduled users) and / or the number of users in the pool also has an effect on the proper selection of the β values, as seen in Figure 4.6. In general, as M_T or K increases, at least one of the β values must also increase to compensate. The dependence on M_T can be seen most clearly when comparing Figure 4.6(a) and (c), while the dependence on K is easiest seen when comparing Figure 4.6(b) and (c). The dependences can also be seen, although less clearly, when comparing Figure 4.2 / Figure 4.3 with Figure 4.5 (for K) and with Figure 4.6(b) (for M_T). However, notably, a change in the number of receive antennas per user N_R does not significantly affect the convergence of the GA. This is seen in comparing Figure 4.5 to Figure 4.6(a). The lowest points in both graphs occur around the same values of β_1 and β_2. This last fact is not that surprising; as we discussed in the previous chapter, the number of iterations required for either GA scheduling or an exhaustive search is independent of the number of receive antennas. Thus, one could expect that the convergence of the GA would still be largely independent of N_R for other values of β_1 and β_2 as well.

Consider the specific case of $(N_R,M_T,K) = (1,4,20)$ shown in Figure 4.6(c). With the original values of $(\beta_1,\beta_2) = (1.2,10)$ as used in the previous chapter, it takes approximately 550 generations to converge on average to within 0.1% of the optimal utility function value. (Note that this value is off the top of the z-axis in Figure 4.6(c).) However, by increasing the parameter values to somewhere in the vicinity of $(\beta_1,\beta_2) = (7,70)$, the required number of generations drops to approximately 155. This is less than 30% of the time with the original values, and clearly illustrates the importance of properly tuning the β values, particularly as K and M_T increase. We demonstrate this further in Figure 4.7, which compares how close on average the GA is to the optimal utility function value as a function of the number of generations for two sets of β values. It can be seen that there is a significant reduction in convergence time over a wide range of levels of convergence.

As we have noted above, rather than one specific operating point, there is instead a range of β values for which the rate of convergence of the GA is approximately the same. This can be seen to some degree in the earlier figures. We demonstrate this more clearly

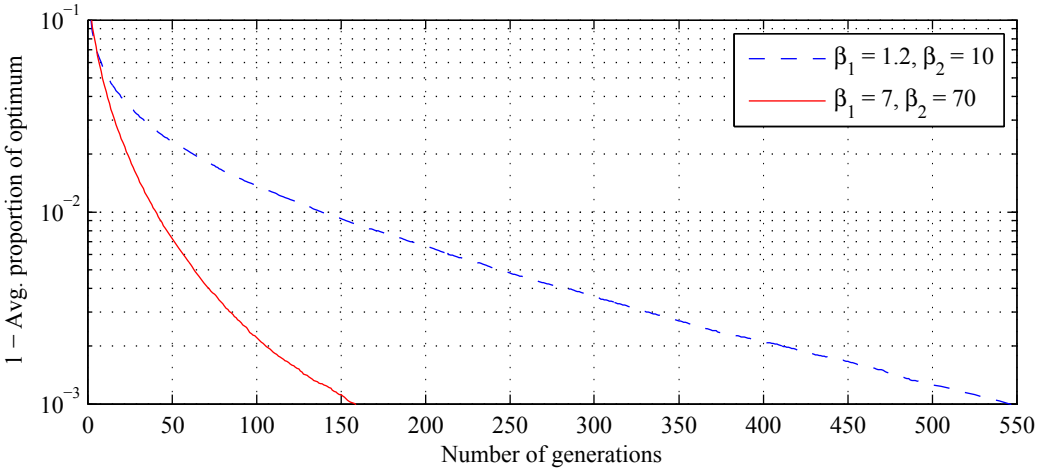

Figure 4.7: Average convergence of GA vs. number of generations at an SNR of 10 dB, with $(N_R,M_T,K) = (1,4,20)$ and two sets of values for β_1 and β_2.

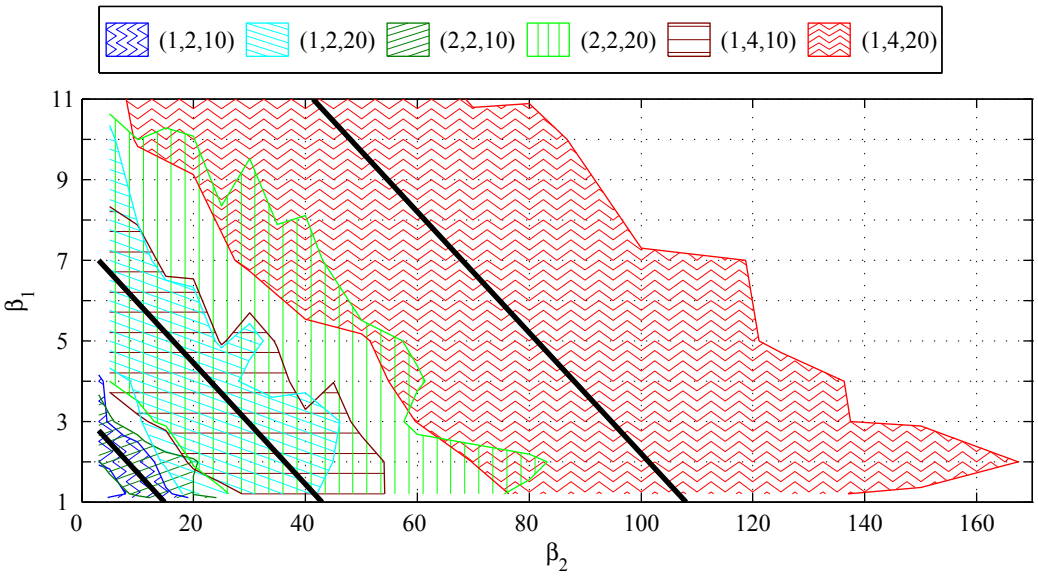

Figure 4.8: Ranges of (β_1,β_2) for which the number of generations for the GA to converge to within 0.1% of the optimum is within 5% of the minimum convergence time.

in Figure 4.8, which depicts the range of β values for which the number of generations required to converge to 0.1% of the optimal value is within 5% of the minimum convergence time obtained over all values of β_1 and β_2. (Note that the ranges are very similar for other levels of convergence as well, just as was seen when comparing Figure 4.4 and Figure 4.5.) There are two key observations from Figure 4.8. First, the ranges for $(N_R,M_T,K) = (1,2,10)$ and $(2,2,10)$, as well as for $(1,2,20)$, $(2,2,20)$, and $(1,4,10)$, overlap. This provides additional confirmation that the specific number of receive antennas per user has little effect on the GA convergence. Rather, the primary effect of N_R is actually

to somewhat increase the range of usable values of β_1 and β_2. This increased range with N_R is a result of the increased receive diversity due to the additional antennas. The increased diversity decreases the variation in the capacity of the various users, thereby decreasing the variation in the solutions represented by the chromosomes (σ_G in Eqn. 4.1), and hence making the GA less sensitive to the β values. (This reduced variation in capacity was also observed in [62], where the phenomenon is referred to as "channel hardening".) More importantly, though, the overlap in the ranges implies that doubling the number of users K has about the same effect on the proper β values as doubling the number of transmit antennas M_T does.

Second, note that each of the ranges has approximately the same slope. This implies that β_1 and β_2 could be expressed as a linear function of each other, as indicated by the black lines in Figure 4.8. Those three lines can be expressed by $\beta_1 + 0.15\beta_2 = c$, where c = 3.227, 7.45, and 17.2, from left to right respectively. Interestingly, those values of c are multiples of each other, i.e., $7.45/3.227 \approx 17.2/7.45 \approx 2.309$. Hence, we see that as either K or M_T doubles for each of the cases, the constant c scales proportionally. This indicates a linear relationship between the log of c and the log of the product KM_T. Thus, based on the relationship between β_1, β_2, M_T, K, and c seen in Figure 4.8, after performing a simple least-squares polynomial coefficient best fit (shown in Appendix C), it would seem that for any value of K and M_T, one could select any value of β_1 and β_2 that satisfies the following constraints to reduce the average convergence time to within 5% of the minimum:

$$\beta_1 + 0.15\beta_2 = \left(KM_T/7.5773\right)^{1.2071}, 1.1 \le \beta_1 \le 11, \beta_2 \ge 3\,. \tag{4.2}$$

The above equation might also hold for a larger range of β_1 and β_2, but this would require further analysis. Nonetheless, this equation gives a very simple method to adapt the mutation rate parameters to changing system characteristics, such as users entering or leaving the system. We also note that given the fairly wide range of useful values seen in Figure 4.8, the specific values used for β_1 and β_2 do not necessarily have to precisely satisfy the equation. For example, for simplicity of implementation, one could quantize the values to, say, the nearest integer or nearest multiple of 0.5, and still obtain a very good convergence rate. In such a case, it is then probably better to select a value for β_1 and solve for β_2, since the range of values that β_2 can take on is larger.

As a side comment, from a practical standpoint, the lowest value β_1 can safely take on is exactly 1. It is possible, though quite unlikely, that the genetic algorithm could

create a set of chromosomes that all have the same fitness. In that case, the variance of the population's fitness would become zero, and β_2 would have no effect on the adaptive mutation rate. In this event, the mutation rate effectively becomes $1/\beta_1$. Were β_1 any smaller than 1, this would mean the probability of mutation would be greater than 100%, which is impossible. To be on the safe side, we consider only a minimum value of β_1 of 1.1, to allow for a margin of error, and a smaller maximum possible value for p_m.

4.4 Further Discussion

4.4.1 Interpretation of Equation for Parameter Values

Overall, there is a certain "best" operating range for the mutation rate that likely depends on many factors, including the channel, population size, the specific form of the chromosomes, and so forth. The simulations of the previous section showed that the effect of the parameter values becomes greater the closer the algorithm is to the optimal utility function value, and that the values of β_1 and β_2 should depend on the number of users in the pool and the number of transmit antennas (or equivalently, the maximum possible number of simultaneously scheduled users). Specifically, as either K or M_T increases, so too should β_1 and / or β_2. In all, these results are not that surprising, and their reasons can be explained somewhat intuitively.

To begin, the mutation operation exists to force genetic diversity in the population to avoid the algorithm stalling at local optima. It is thus most important and influential in later generations. Earlier, when there is already much diversity due to the random initialization, the crossover operator instead affects the GA's initial convergence the most. Therefore, obviously the parameter values and thus the mutation rate will have the most notable effect the closer the algorithm is to the optimal value, as seen in the simulations.

In general, mutations are intended to have a fairly small effect on the chromosomes, so that they mostly maintain the characteristics of a good solution, while still guiding the GA away from local optima. While too small of a mutation rate can slow the convergence rate due to the algorithm becoming stuck at local optima, too large of a mutation rate also risks slowing the convergence. In the following, we focus mostly on the head of the chromosome, as the selection of scheduled users usually has more of an impact on the maximization of the utility function than the order in which those users are encoded does.

As the pool of users increases in size, so too does the length of the chromosomes. Consequently, for a given mutation probability, the longer chromosome has a higher

probability of experiencing *some* mutation. The probability of the head of the chromosome *not* mutating is $(1-p_m)^K$, which decreases exponentially with K. By the same reasoning, as K increases, it is therefore more likely for a chromosome to experience a mutation of multiple bits. The more bits that are mutated, the more likely it is that the chromosome will be harmfully impacted. The most harmful mutation that can occur is if a particularly good user who should be scheduled becomes de-selected. This could happen directly as a result of mutation of that user's bit in the chromosome. It could also happen indirectly during the correction step; if too many users are scheduled, that user could be randomly de-selected[1] while reducing the weight of the head to M_T. Consequently, one can expect that as K increases, the mutation rate should be decreased somewhat to compensate. This can be accomplished by increasing the β values with K.

In the case of an increase in M_T, more transmit antennas means more users can be scheduled simultaneously. Hence, more '1's would exist in the heads of the chromosomes. For a given mutation rate, there is therefore a higher chance that a '1' will be toggled to a '0'. As mentioned above, de-scheduling a good user is a particularly harmful mutation. More generally, de-selecting any user from being scheduled will very often result in a significant drop in the system capacity and the value of the utility function, especially with DPC. While it is usually not necessary to schedule more than M_T users, scheduling fewer than M_T users will almost always result in a significant reduction in the system capacity [75]. Doing so means that all the available degrees of freedom for scheduling and resource allocation are not being exploited. The exception to this may be if the SNR is low or that user happened to be experiencing bad channel conditions. However, in general, to avoid this scenario, the mutation rate should again be lowered as M_T increases to compensate for the increased likelihood of de-scheduling a user.

The relationship between β_1, β_2, K, and M_T as found by the simulations is reasonably logical. Intuitively, for some given ratio σ_G/μ_G, one might expect or desire that p_m should change such that, whatever the value of K is, the probability of no mutation occurring at all in (the head of) a chromosome should remain approximately constant. That way, the probability of a detrimental mutation would also be approximately constant. Hence, we would have $(1-p_m)^K \approx Z$, where Z is some constant. For $p_m << 1$, this can be approximated as $1 - Kp_m \approx Z$, or $Kp_m \approx Y$, where $Y = 1 - Z$. Substituting in Eqn. (4.1), we

[1] During correction, if the weight w of the head is greater than M_T, each scheduled user has an equally likely 1 in w chance of being de-selected.

get $\beta_1 + \beta_2 \sigma_G/\mu_G \approx K/Y$ for some ratio σ_G/μ_G. Thus, one could expect a nearly linear relationship between β_1, β_2, and K. Similarly, one might intuitively expect a change in p_m such that the probability of de-selecting none of the scheduled users is approximately constant, for whatever the value of M_T may be. Since the mutation operation acts independently on each bit of the chromosome, this probability is $(1-p_m)^{M_T}$. Thus, by a similar derivation to that above, we would end up with $\beta_1 + \beta_2 \sigma_G/\mu_G \approx M_T/A$ for some ratio σ_G/μ_G and some constant A.

This intuitive line of thought is not too different from the simulation results and Eqn. (4.2). The value of 0.15 in (4.2) most likely represents some sort of average for the ratio σ_G/μ_G. However, the power of 1.2071 on the KM_T term indicates that the probability of no mutation occurring should not quite be constant, but rather grow to a small degree with increasing K and M_T. What is most likely happening is that the equation is providing some sort of middle ground between keeping the probability of not mutating fixed with K and M_T, and yet still dealing with the combinatorially increasing search space. The fact that the encoding order bits can also be mutated would also have an effect.

We expect that Eqn. (4.2) could possibly be used with precoding methods other than DPC (e.g. ZFB). However, in such cases, M_T may have to be replaced with N_S, the maximum possible number of simultaneously scheduled users, if $N_S \neq M_T$. In other precoding methods, there may not be an encoding order to consider, if any particular order results in the same user rates, like in ZFB and BD. Alternatively, the encoding order might actually affect the utility function value, in contrast to maximum throughput scheduling for DPC. However, in most of these cases, it is likely to be which users are scheduled, rather than their encoding order, that has the largest effect on the utility function value. Since we have seen that Eqn. (4.2) is largely based off the head of the chromosome (i.e., the scheduled users), the equation should still work reasonably well with those different precoding methods.

4.4.2 Other SNRs and Utility Functions

The work we have presented in this chapter has focused on convergence results for only the maximum throughput scheduling criterion at an SNR of 10 dB. However, the results are much the same for other SNRs. The change in the convergence rate with β_1 and β_2 is very similar at, for example, an SNR of 0 dB. We demonstrate this in Figure 4.9. We have considered the case of $(N_R,M_T,K) = (2,2,20)$ with parameter values of $(\beta_1,\beta_2) =$

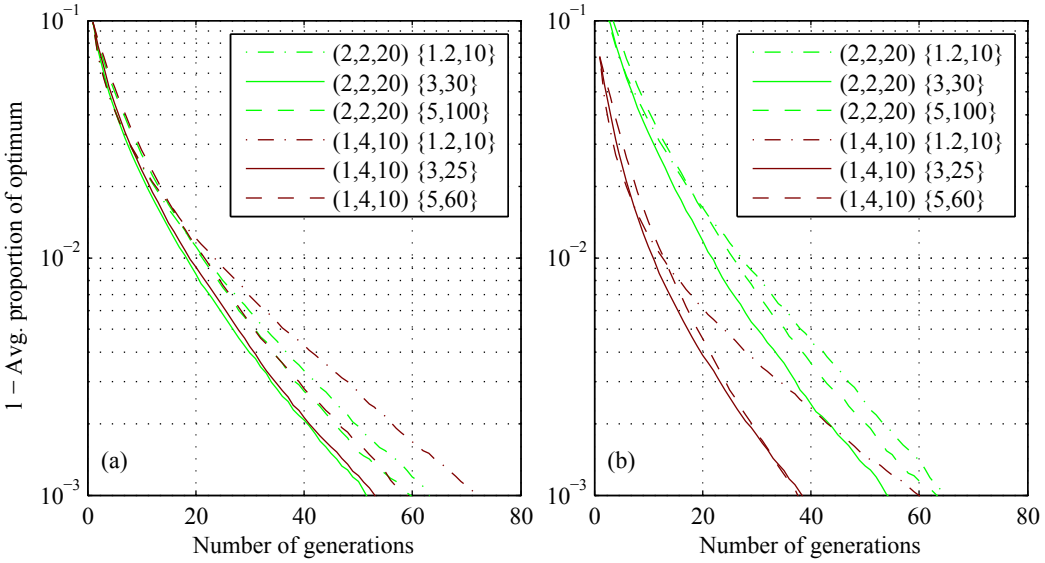

Figure 4.9: Comparison of convergence of GA for $(N_R,M_T,K) = (2,2,20)$ and $(1,4,10)$ when changing $\{\beta_1,\beta_2\}$ at different SNRs. (a) SNR = 10 dB. (b) SNR = 0 dB.

(1.2,10), (3,30), and (5,100), and the case of $(N_R,M_T,K) = (1,4,10)$ with parameter values of $(\beta_1,\beta_2) =$ (1.2,10), (3,25), and (5,60). These β values were selected since, in their respective cases, the first set (1.2,10) are the original values used in the previous chapter, the middle set are values that reduce the average convergence time to about the minimum, and the last set are values that are too large at 10 dB, which result in an increased convergence time.

It can be seen that the change in the convergence time with changes in β_1 and β_2 is largely the same at 0 dB as it is at 10 dB. For the (2,2,20) case, parameter values of (1.2,10) and (5,100) result in a lengthened convergence time, while (3,30) improves the convergence. The (1,4,10) case is a bit more interesting. It is again seen that values of (1.2,10) result in a prolonged convergence time, while values of (3,25) provide an improvement. However, values of (5,60) also result in a very similar convergence rate as (3,25). A likely cause for this is that the values (3,25) and (5,60) are probably close to the "edges" of the region that provides a good convergence rate at 0 dB. This further implies that at 0 dB, the regions might be somewhat larger than at 10 dB; the values (5,60) fall outside the (1,4,10) region for 10 dB in Figure 4.8, but appear to be not as bad at 0 dB. The values of (3,25) are a bit off the line established by Eqn. (4.2); values of (3,30) or (4,23) would be closer. Nevertheless, it is overall still clear that Eqn. (4.2) and values thereabout still provide a notable improvement in convergence even at other SNRs. This result is not that surprising, given the results of our work in the previous chapter. In that

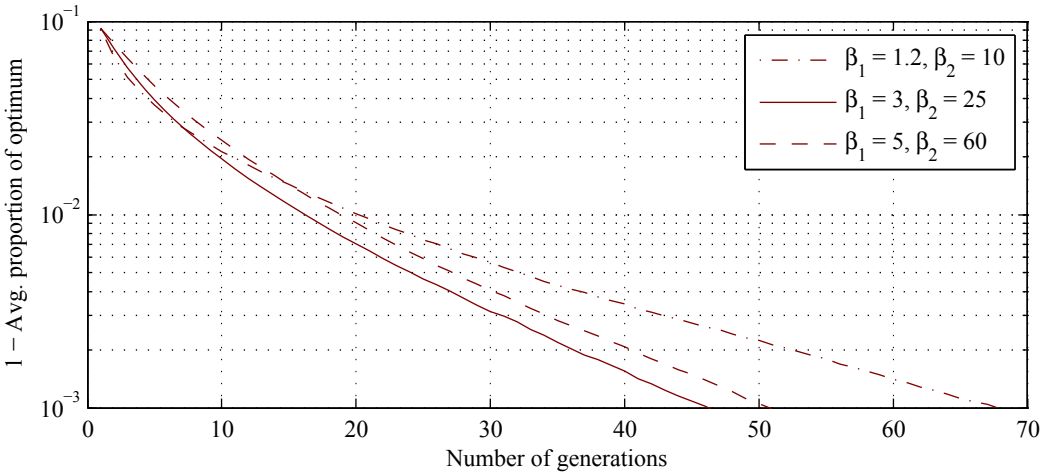

Figure 4.10: Average convergence of GA scheduling algorithm with the proportional fairness criterion, an SNR of 10 dB, and various values for β_1 and β_2.

work, for a fixed β_1, β_2, and number of generations, the GA converged to approximately the same percentage of the optimal utility function value regardless of the SNR. Thus, seeing almost no SNR dependence for other β values can be reasonably expected.

We have also performed some brief simulations using the proportional fairness (PF) scheduling criterion. As we explained in the previous chapter, simulating the PF criterion requires much more effort, since it requires average statistics to be built up before useful data can be obtained. Thus, we limit our focus to the case of $(N_R,M_T,K) = (1,4,10)$ with the same parameter values as for the 0 vs. 10 dB results earlier. The average convergence of the PF scheduling algorithm is shown in Figure 4.10, in terms of the distance away from the optimal utility value G_{PF}, as given in Eqn. (3.15). The results also show a similar overall change in the convergence rate with changing β values, but with a small difference in the rate of convergence compared to the maximum throughput criterion. That is, the convergence with the PF criterion tends to be slightly faster. It also appears that the same β values that produce a minimum convergence time for the maximum throughput criterion result in a convergence time close to the minimum for the PF criterion as well. There is very little difference in the overall look of the graph in Figure 4.10 and that in Figure 4.9(a).

4.5 Uniform Crossover

Another possibility for improving the performance of the genetic algorithm is to change some of the details of the steps it uses. To that end, we have also investigated replacing the one-point crossover method in our GA. Numerous studies over the years

have indicated that the one-point crossover is not necessarily the best operator. Rather, it has overall been seen that the two-point crossover and the uniform crossover perform better [128],[129],[130],[131]. The two-point crossover is much the same as the one-point crossover, except that instead two crossover points are defined at random, and the bits between those points are exchanged between the parents. This operation still occurs with probability p_c. In uniform crossover, each bit in the chromosome has the same probability to be exchanged between the parents. Most often, the probability of an exchange of each bit is 50% in uniform crossover; each bit is as equally likely as not to be exchanged. We have chosen to examine the uniform crossover, as studies have indicated that uniform crossover works better than the two-point crossover when using smaller population sizes [128],[129].

We compare the convergence of the GA using the one-point and the uniform crossover in Figure 4.11 and Figure 4.12. Both the original β values from the previous chapter and improved β values found from Eqn. (4.2) earlier this chapter are examined. We have also considered the cases of (N_R,M_T,K) = (1,2,10) and (2,2,10). However, we have found no discernable difference between the two crossover methods in those two cases, for either set of β values. Thus, we have not graphed these cases, as the graphs overlap and cannot easily be distinguished from each other.

Examining the figures, it can be seen that there is mostly no significant change in performance for the crossover methods when the original β values are used. It is not until we consider the improved β values that we can start to see a difference. Nonetheless, the improvement is initially small. There is still no significant change in convergence when using the uniform crossover with (N_R,M_T,K) = (1,2,20) in Figure 4.11(a). There is a small convergence improvement with the uniform crossover with (N_R,M_T,K) = (2,2,20) in Figure 4.11(b), but not until an average convergence of about 0.5% away from the optimal utility function value. Even then, the improvement in convergence time is at best about 1 to 3 generations, although this does represent a relative improvement in convergence time by about 4–8%.

The improvement in convergence time becomes more significant when examining the M_T = 4 cases. The improvement in convergence time with the uniform crossover can be seen sooner in Figure 4.12(a), where again the improvement is at maximum about 8%. We note again, though, than on an absolute scale, this only represents a savings of about 1–4 generations. The most significant improvement is seen in Figure 4.12(b) for

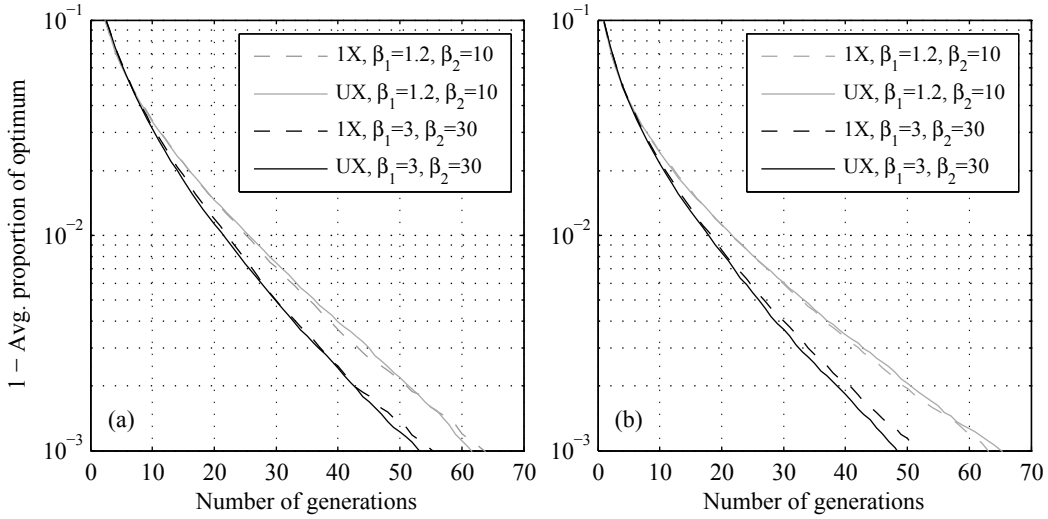

Figure 4.11: Comparison of GA convergence with 1-point crossover (1X) and uniform crossover (UX) operators, each with two sets of β parameter values. (a) $(N_R,M_T,K) = (1,2,20)$. (b) $(N_R,M_T,K) = (2,2,20)$.

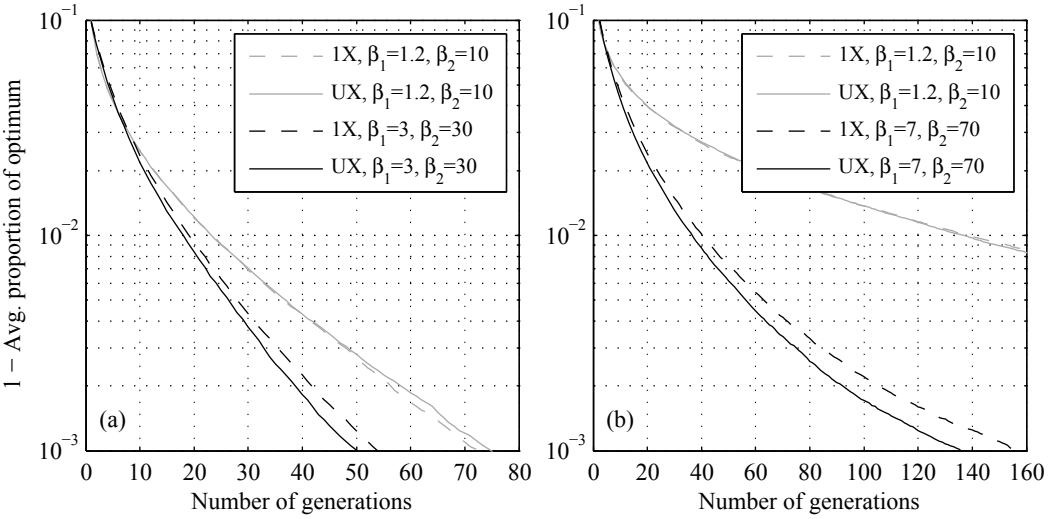

Figure 4.12: Comparison of GA convergence with 1-point crossover (1X) and uniform crossover (UX) operators, each with two sets of β parameter values. (a) $(N_R,M_T,K) = (1,4,10)$. (b) $(N_R,M_T,K) = (1,4,20)$.

$(N_R,M_T,K) = (1,4,20)$. At an average convergence level of 0.1%, the convergence time drops by about 20 generations, or about 13%.

Overall, it can be seen that there is some improvement in convergence that can be obtained by the use of the uniform crossover operator, but these gains are a bit limited, especially when compared to the improvements seen when adjusting the parameter values for β_1 and β_2. Furthermore, there are additional considerations in using the uniform crossover method. First, the uniform crossover is more complicated than the one-point

crossover. A random number must be generated for every bit in the chromosome, to determine if that bit is exchanged, whereas only a single random crossover location must be generated for the one-point crossover. This overhead is not too bad; it is mostly negligible when compared to the complexity of calculating the utility function value. It is even less of a worry if the uniform crossover probability is 50%; in that case, only random binary values need to be generated. More important is the fact that our GA scheduling algorithm only operates for a limited number of generations. In that limited amount of time, the amount of improvement that can be obtained from the uniform crossover is small. The gain in using uniform crossover, both in terms of the additional relative throughput increase and the relative convergence time saved, is about half an order of magnitude smaller than the gain by adjusting the β values. More improvement would be attained if the algorithm were to run longer.

Upon reflection, the relatively small gains obtained by using the uniform crossover operator instead of the one-point crossover operator are not that surprising, nor is the fact that significant gains are not seen until $M_T = 4$. The chromosomes used for scheduling are rather "sparse" in content. Since normally $K >> M_T$ in wireless systems, the vast majority of the bits in the GA chromosome will be '0's. This means that regardless of if the GA exchanges a large block of bits simultaneously, or does so one by one, most of the bits that are exchanged will be '0's for other '0's. Thus, there is effectively not much difference between the two methods. More gain being seen at $M_T = 4$ stems from the fact that there are more '1's in the chromosome that can possibly be exchanged for what is more likely to be a '0' in the other parent. The gain also comes from the fact that the users will be swapped individually and independently rather than in a block. For example, if the crossover point in the one-point crossover is just before user k, any and all users k to K that are scheduled will be swapped between the parents. More users on average will be in that block to be swapped with larger M_T. On average, in the long term, half of the bits in a chromosome can be expected to be swapped with both methods. In the uniform crossover, these bits are evenly distributed throughout the chromosome, while in the one-point crossover, the bits are all on average in the second half of the chromosome. This may not always be bad, but the simulation results indicate that independent swapping can lead to a small improvement in performance. Given the results of the simulations, we expect that there may be an even larger gain to be found for even larger values of M_T, for which even more users can be scheduled simultaneously.

4.6 Conclusion

In this chapter, we have examined the impact of the adaptive mutation rate parameters on the convergence rate of genetic scheduling algorithms. We have observed that there is a reasonably wide range of values that result in a similar, near-minimum convergence time for the algorithm. However, we have also seen that it is important to tune the parameters to ensure the algorithm is operating within that range. In one case, tuning the parameters resulted in the number of generations required for the algorithm to converge dropping to less than 30% of the number required when the original parameter values from the prior chapter were used.

We have also seen that the proper values for the parameters are dependent on both the number of users K and the number of transmit antennas M_T, but less so on the number of receive antennas per user N_R. We have proposed a simple equation that is linear in β_1 and β_2 to tune the parameters for changing K and M_T. This equation was seen to work well for various signal-to-noise ratios, and for both the maximum throughput and proportional fairness scheduling criteria.

The effect of using a uniform crossover method in the GA instead of the previously used one-point crossover method was also examined. There is some improvement in the convergence of the GA to be obtained, but nowhere near as much as when tuning the adaptive mutation rate parameter values. The gain with uniform crossover is the most when M_T is larger and the genetic algorithm is allowed to run for more generations.

There is another potential detail of the GA where adjustment could lead to an improvement in performance. This is in the probability of crossover p_c. We have used the value $p_c = 1$ throughout our work. This value is at the upper limit for p_c, and means that a crossover always occurs. It is possible that tuning this value may also improve the GA performance. The specific value of p_c may also impact the proper values for the mutation rate and / or the values for β_1 and β_2. Examining the impact of tuning p_c along with p_m on the scheduling performance would be an interesting area for possible future work.

Chapter 5

Genetic, Greedy, and Hybrid Scheduling Algorithms for Block Diagonalization and Successive Zero-Forcing

5.1 Introduction

In the previous chapter, we examined GA scheduling for MIMO systems employing dirty paper coding (DPC). This allowed us to focus solely on the effect of scheduling in an environment that would, scheduling issues aside, achieve the maximum possible capacity. Since unfortunately DPC is currently impractical for implementation, lower complexity precoding methods are of interest. Thus, we now shift our attention to scheduling methods for block diagonalization (BD) [46] and successive zero-forcing (SZF) [50] precoding. Both precoding methods are linear and thus of significantly lower complexity than DPC, though naturally their capacity is lower as well. In this chapter, we evaluate the performance of genetic and greedy scheduling algorithms for BD and SZF. We also compare the complexity of those algorithms in relation to existing algorithms and an exhaustive search. Finally, we also propose hybrid algorithms of the genetic and greedy algorithms that combine the characteristics of the two, which result in increased performance without an increase in the order of complexity. Our contributions in this chapter have appeared in [132],[133],[134].

5.2 System Model with Linear Precoding

The system model we use is largely the same as in the previous chapters on GA scheduling for DPC. The main difference now is that rather than encoding each user's signal with DPC, the transmitted signal for each user k is instead preprocessed with a transmit beamforming matrix $\mathbf{W}_k \in \mathbb{C}^{M_T \times N_k}$. If user k has N_k receive antennas, the $N_k \times 1$ received signal $\mathbf{y}_k$ at each user then becomes:

$$\mathbf{y}_k = \mathbf{H}_k \sum_{j=1}^{K_0} \mathbf{W}_j \mathbf{s}_j + \mathbf{n}_k . \tag{5.1}$$

$\mathbf{H}_k \in \mathbb{C}^{N_k \times M_T}$ is the channel matrix for user k, $\mathbf{s}_k \in \mathbb{C}^{N_k \times 1}$ is the data symbol vector intended for user k, and $\mathbf{n}_k \in \mathbb{C}^{N_k \times 1}$ is additive white Gaussian noise with variance $E\left\{\mathbf{n}_k \mathbf{n}_k^H\right\} = \sigma_n^2 \mathbf{I}_{N_k}$. K_0 is the number of users that the system transmits to simultaneously. Otherwise, the system details are largely the same as in the previous chapters.

5.2.1 Block Diagonalization

Block diagonalization (BD) [46] was discussed briefly in Chapter 2. We expand upon the details here.

BD is designed to completely eliminate all existing multiuser interference (MUI) at the transmitter. It does so by designing the beamforming matrices such that $\mathbf{H}_k \mathbf{W}_j = \mathbf{0}$ for all $k \neq j$. This in effect decouples the multiuser broadcast channel into parallel equivalent single-user channels. The received signal from Eqn. (5.1) then becomes $\mathbf{y}_k = \mathbf{H}_k \mathbf{W}_k \mathbf{s}_k + \mathbf{n}_k$. If we consider the aggregate channel matrix:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1^T & \mathbf{H}_2^T & \cdots & \mathbf{H}_{K_o}^T \end{bmatrix}^T , \tag{5.2}$$

and the aggregate precoding matrix:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 & \cdots & \mathbf{W}_{K_0} \end{bmatrix}, \tag{5.3}$$

then the overall product **HW** will have a block-diagonal structure (from which BD takes its name).

Let us define the aggregate matrix $\tilde{\mathbf{H}}_k$ as follows:

$$\tilde{\mathbf{H}}_k = \begin{bmatrix} \mathbf{H}_1^T \cdots \mathbf{H}_{k-1}^T \, \mathbf{H}_{k+1}^T \cdots \mathbf{H}_{K_0}^T \end{bmatrix}^T \in \mathbb{C}^{\sum_{j=1, j\neq k}^{K_0} N_j \times M_T} ; \tag{5.4}$$

that is, the concatenation of all channel matrices except that of user k. The zero-MUI constraint is satisfied by requiring that $\mathbf{W}_k$ fall in the null space of $\tilde{\mathbf{H}}_k$. This naturally implies that $\tilde{\mathbf{H}}_k$ has a null space of dimension greater than zero. This will be satisfied if $\text{rank}\left(\tilde{\mathbf{H}}_k\right) = \tilde{r}_k < M_T$. Thus, up to K_0 users can be supported simultaneously with BD if $\max\left\{\tilde{r}_1, \tilde{r}_2, \cdots, \tilde{r}_{K_0}\right\} < M_T$ [46]. We assume that the fading both between any two users and between the antennas of any given user is independent. Thus, each channel matrix will be of full rank, which is equal to N_k (assuming $N_k < M_T$). In this case, the maximum number

of users K_0 that can be supported [1] can also be determined by the constraint $\sum_{j=1, j\neq k}^{K_0} N_j < M_T, \forall k$. In the event that all users have the same number of receive antennas N, this further simplifies to $K_0 = \lceil M_T/N \rceil$, where $\lceil \cdot \rceil$ is the ceiling function.

Let the singular value decomposition (SVD) of $\tilde{\mathbf{H}}_k$ be denoted as $\tilde{\mathbf{H}}_k = \tilde{\mathbf{U}}_k \tilde{\mathbf{D}}_k \tilde{\mathbf{V}}_k^H = \tilde{\mathbf{U}}_k \tilde{\mathbf{D}}_k \left[\tilde{\mathbf{V}}_k^1 \; \tilde{\mathbf{V}}_k^0 \right]^H$, where $\tilde{\mathbf{V}}_k \in \mathbb{C}^{M_T \times M_T}$. $\tilde{\mathbf{V}}_k^1$ contains the first $\tilde{r}_k$ right singular vectors, while $\tilde{\mathbf{V}}_k^0$ contains the remaining $M_T - \tilde{r}_k$ right singular vectors. The columns of $\tilde{\mathbf{V}}_k^0$ form an orthonormal basis for the null space of $\tilde{\mathbf{H}}_k$. Constructing the beamforming matrices from the columns of $\tilde{\mathbf{V}}_k^0$ will satisfy the zero-MUI requirement. The multiuser channel will then be decoupled into the following equivalent parallel single-user channels:

$$\mathbf{H}_{k,e} = \mathbf{H}_k \tilde{\mathbf{V}}_k^0 . \tag{5.5}$$

With a transmit power constraint P at the base station, the throughput achievable by block diagonalization is obtained by maximizing:

$$R_{BD} = \max_{\mathbf{Q}_k : \mathbf{Q}_k \succeq \mathbf{0}} \sum_{k=1}^{K_0} \log_2 \left| \mathbf{I} + \frac{1}{\sigma_n^2} \mathbf{H}_{k,e} \mathbf{Q}_k \mathbf{H}_{k,e}^H \right| , \tag{5.6}$$

such that $\sum_{k=1}^{K_0} Tr\left(\mathbf{Q}_k\right) \leq P$. $\mathbf{Q}_k$ is the square, positive semidefinite transmit covariance matrix for the equivalent channel $\mathbf{H}_{k,e}$, with dimensions equal to the number of columns in $\tilde{\mathbf{V}}_k^0$. The matrices $\mathbf{Q}_k$ can be obtained by the well-known waterfilling solution over the equivalent block-diagonal channel matrix $\mathbf{H}_e = blkdiag\left(\mathbf{H}_{1,e}, \mathbf{H}_{2,e}, \cdots, \mathbf{H}_{K_0,e}\right)$ with the power constraint P [46].

While $\tilde{\mathbf{V}}_k^0$ can be obtained through an SVD, a method to obtain $\tilde{\mathbf{V}}_k^0$ through a QR decomposition, which is more computationally efficient and numerically stable than an SVD, was presented in [138].

[1] Even more users can be supported if the transmitter can account for the filter matrices $\mathbf{M}_k$ or antenna selection at the receivers. The transmitter can instead consider the null spaces of the effective matrices $\mathbf{M}_k\mathbf{H}_k$ when performing the null space operation [46],[48],[135],[136],[137]. This process is known as coordinated beamforming. However, such techniques are beyond the scope of this work.

5.2.2 Successive Zero-Forcing

Successive zero-forcing (SZF) [50] is similar in some aspects to BD. However, unlike BD, it does not completely null all MUI. As its name implies, the precoding is performed successively, and so an encoding order must be defined. For a given set of K_0 users with an encoding order π, the received signal for each user k in Eqn. (5.1) can be expanded as[1] [50]:

$$\mathbf{y}_{\pi(k)} = \mathbf{H}_{\pi(k)}\left(\mathbf{W}_{\pi(k)}\mathbf{s}_{\pi(k)} + \sum\nolimits_{i<k}\mathbf{W}_{\pi(i)}\mathbf{s}_{\pi(i)} + \sum\nolimits_{i>k}\mathbf{W}_{\pi(i)}\mathbf{s}_{\pi(i)}\right) + \mathbf{n}_{\pi(k)}. \tag{5.7}$$

In SZF, the precoding matrix $\mathbf{W}_{\pi(k)}$ is designed such that it lies in the null space of the aggregate channel $\bar{\mathbf{H}}_{k-1}$ of the $k-1$ previously precoded users' channels (in contrast to the null space of all other users with BD):

$$\bar{\mathbf{H}}_{k-1} = \left[\mathbf{H}_{\pi(1)}^T\ \mathbf{H}_{\pi(2)}^T \cdots \mathbf{H}_{\pi(k-1)}^T\right]^T. \tag{5.8}$$

With this null space constraint, we have that $\mathbf{H}_{\pi(k)}\mathbf{W}_{\pi(i)} = \mathbf{0}$ for all $i > k$. Thus, the third term in the sum of Eqn. (5.7) is cancelled, and the equation reduces to:

$$\mathbf{y}_{\pi(k)} = \mathbf{H}_{\pi(k)}\left(\mathbf{W}_{\pi(k)}\mathbf{s}_{\pi(k)} + \sum\nolimits_{i<k}\mathbf{W}_{\pi(i)}\mathbf{s}_{\pi(i)}\right) + \mathbf{n}_{\pi(k)}. \tag{5.9}$$

SZF of K_0 users' channels is possible[2] if $\text{rank}\left(\bar{\mathbf{H}}_{K_0-1}\right) < M_T$. If we assume full-rank channel matrices as we did for BD, this constraint becomes $\sum_{k=1}^{K_0-1} N_{\pi(k)} < M_T$. Furthermore, if all users have the same number of receive antennas N, this finally becomes $K_0 = \lceil M_T/N \rceil$, just as in BD.

Taking the SVD of (5.8) yields $\bar{\mathbf{H}}_{k-1} = \bar{\mathbf{U}}_{k-1}\bar{\mathbf{D}}_{k-1}\bar{\mathbf{V}}_{k-1}^H = \bar{\mathbf{U}}_{k-1}\bar{\mathbf{D}}_{k-1}\left[\bar{\mathbf{V}}_{k-1}^1\ \bar{\mathbf{V}}_{k-1}^0\right]^H$, where $\bar{\mathbf{V}}_{k-1} \in \mathbb{C}^{M_T \times M_T}$. Similar to BD, $\bar{\mathbf{V}}_{k-1}^1$ holds the first $\text{rank}\left(\bar{\mathbf{H}}_{k-1}\right)$ right singular vectors, while $\bar{\mathbf{V}}_{k-1}^0$ contains the remaining $\bar{v}_k = M_T - \text{rank}\left(\bar{\mathbf{H}}_{k-1}\right)$ singular vectors. $\bar{\mathbf{V}}_0^0 \triangleq \mathbf{I}_{M_T}$ by definition. The columns of $\bar{\mathbf{V}}_{k-1}^0$ are then an orthonormal basis for the null space of $\bar{\mathbf{H}}_{k-1}$, from which the precoding matrices $\mathbf{W}_{\pi(k)}$ can be constructed.

---

[1] It is assumed from here on that $\sigma_n^2 = 1$ for all users.

[2] As with BD, coordinated beamforming with cooperation between the transmitter and the receivers is possible to increase the number of supportable users, but this is again outside the scope of this work.

Under the assumption that the signals transmitted from the base station are Gaussian-distributed [46],[50], then for a specific set of K_0 users and a specific encoding order $\pi = \pi_i$ for those users, the maximum achievable rate for each of those users is given by:

$$R_{\pi_i(k)} = \log_2 \frac{\left| \mathbf{I} + \mathbf{H}_{\pi_i(k)} \left(\sum_{j=1}^{k} \overline{\mathbf{V}}_{j-1}^{0} \mathbf{B}_{\pi_i(j)} \left(\overline{\mathbf{V}}_{j-1}^{0} \right)^H \right) \mathbf{H}_{\pi_i(k)}^H \right|}{\left| \mathbf{I} + \mathbf{H}_{\pi_i(k)} \left(\sum_{j=1}^{k-1} \overline{\mathbf{V}}_{j-1}^{0} \mathbf{B}_{\pi_i(j)} \left(\overline{\mathbf{V}}_{j-1}^{0} \right)^H \right) \mathbf{H}_{\pi_i(k)}^H \right|}, \tag{5.10}$$

where the precoder input covariance matrices $\mathbf{B}_{\pi_i(k)}$ and the channel input covariance matrices $\mathbf{Q}_{\pi_i(k)}$ are defined such that $\mathbf{Q}_{\pi_i(k)} = \mathbf{W}_{\pi_i(k)} \mathbf{W}_{\pi_i(k)}^H = \overline{\mathbf{V}}_{k-1}^{0} \mathbf{B}_{\pi_i(k)} \left(\overline{\mathbf{V}}_{k-1}^{0} \right)^H$.

The achievable sum rate of SZF precoding for a given user order π_i is:

$$R_{SZF}^{\pi_i} = \max_{\left\{ \mathbf{Q}_{\pi_i(k)} \right\}_{k \in \{1,\cdots,K_0\}} : \mathbf{Q}_{\pi_i(k)} \succeq 0, \sum_k Tr\left(\mathbf{Q}_{\pi_i(k)} \right) \le P} \sum_{k=1}^{K_o} R_{\pi_i(k)} \, . \tag{5.11}$$

The maximum achievable sum rate R_{SZF} of SZF precoding is then obtained by maximizing (5.11) over all $K_0!$ possible user orders:

$$R_{SZF} = \max_{\pi_i, i=1,2,\cdots,K_0!} R_{SZF}^{\pi_i} \, . \tag{5.12}$$

Solving equations (5.10)–(5.12) to determine the optimal covariance matrices can be quite difficult, as the problem is not convex. However, for a given encoding order, the authors in [50] have proposed a suboptimal numerical technique based on DPC covariance optimization to solve Eqn. (5.11). The method involves using the sum-power iterative waterfilling method of [107] to find the optimal covariance matrices for the multiple access channel (MAC), using the MAC to BC transformations in [24] to obtain covariance matrices for the broadcast channel under DPC, and then projecting those matrices to the SZF null spaces to obtain $\mathbf{Q}_{\pi_i(k)}$. We use this suboptimal method in this chapter. Full details on the technique can be found in Appendix D.

There has been very little work in the literature on scheduling and ordering issues for SZF. The number of possible ordered user subsets (given by Eqn. (3.2)) and the fact that each ordering provides a different sum rate (in notable contrast to DPC) make the problem quite difficult. A further simplified covariance optimization scheme, along with a suboptimal user ordering algorithm to be used in conjunction, was proposed in [139]. However, that algorithm only performs well in the low-SNR regime. SZF has also been examined in the context of clustered network MIMO systems in [140]. However, to the best of our knowledge, no other work in the literature has dealt with this topic.

5.3 Scheduling Algorithms

5.3.1 Genetic Algorithms

The genetic scheduling algorithms we use are by and large the same as what is described in Chapter 3. Some of the details have been modified, though, to account for scheduling in the context of BD and SZF.

To begin, for BD scheduling, rather than attempting to optimize the DPC sum rate, we are instead maximizing the sum of the rates of the scheduled users, as given by Eqn. (5.6). Thus, the fitness of the chromosomes is found from (5.6) when optimal covariance matrices are used. In this scenario, the encoding order is no longer a factor in the scheduling decision or the sum-rate maximization. Thus, the tail of the GA chromosomes is no longer necessary, so we simply remove it. In that respect, the chromosome representation becomes the same as that used in [109]. We are no longer concerned with the encoding order when dealing with GA elitism, either. Thus, we modify the breeding process as follows. During each generation of the GA, rather than only create a new population of $N_p - 2$ chromosomes by breeding, we do create a full set of N_p chromosomes. Then, denoting C^* as the best chromosome from the previous generation, if C^* does not already exist in the new population, the chromosome with the worst fitness in the new population is replaced by C^*, provided that the fitness of that worst chromosome is lower than that of C^*.

For SZF, the details of the GA operation are almost identical to what is described in Chapter 3. The only difference is that the fitness of a chromosome is now defined by Eqn. (5.11) for the users and encoding order of that chromosome. We use the SZF covariance method described earlier and in Appendix D when determining the fitness.

For both precoding methods, we adopt the improved adaptive mutation rate parameters from the previous chapter[1]. In the equation for β_1 and β_2, we replace the M_T term with K_0, the maximum number of users that can be scheduled using either precoding method. We also ignore the upper bound on β_1, since we shall be considering a wider range for the number of active users K, and that upper bound may become too restrictive and degrade the performance. We wish to choose β values somewhere in the middle of

[1] However, we do not adopt the uniform crossover operator, but continue to use the one-point crossover. This is mostly because the majority of the work in this chapter was completed before we decided to examine the crossover methods. However, this is still justified in that we saw in the previous chapter there was little additional gain in performance to be obtained by using the uniform crossover with limited generations, compared to the large gain seen by adjusting β_1 and β_2.

Table 5.1: Adaptive mutation rate parameter values used for varying numbers of active users in pool (K) and varying numbers of simultaneously supportable users (K_0)

(K,K_0)	Maximum Possible Value for β_1	Adaptive Mutation Rate Parameter Values	
		β_1	β_2
(8,2)	2.02	1.5	6.5
(10,2)	2.78	2	8
(16,2)	5.42	3	18
(20,2)	7.00	4	23
(30,2)	11.70	6.5	38
(40,2)	16.75	9	55
(50,2)	22.07	11.5	73.5
(70,2)	33.35	17	112
(100,2)	51.54	26	173
(8,4)	5.42	3	18
(10,4)	7.00	4	23
(16,4)	12.69	7	41
(20,4)	16.75	9	55
(30,4)	27.61	14.5	90.5
(40,4)	39.26	20	131
(50,4)	51.54	26	173
(70,4)	77.59	39	260
(100,4)	119.58	60	400

the range imposed by the constraints $\beta_1 \geq 1.1$ and $\beta_2 \geq 3$. Note that the lower limit on β_2 imposes an upper limit on β_1 when Eqn. (4.2) is solved (and vice versa); that limit is dependent on the values for K and K_0. In choosing the values, we choose a value for β_1 approximately in the middle of the range of its allowable values, then solve for β_2. Since it was seen in the previous chapter that the values do not have to fall exactly on the line defined by Eqn. (4.2), we generally also quantize the values of β_1 and β_2 to the nearest multiple of 0.5. Table 5.1 shows the values used for β_1 and β_2 in this chapter.

In deciding for how long to allow the GA to run, recall from Chapter 3 that it is the product $N_p \times N_g$ that appears to determine the performance more than the individual values. This was seen in the near-identical performance for $K = 10$ and $K_0 = 4$ with $(N_p,N_g) = (10,10)$ and (20,5). With this in mind, the values for N_p and N_g used in Chapter 3 seem to provide a performance reasonably close to optimal. Thus, it would be logical to continue with the same overall trend for those values in this work. Previously, for $(K_0,K) = (2,10)$, (2,20), (4,10), and (4,20), we had the equivalent of $(N_p,N_g) = (10,5)$, (10,10), (20,5), and (20,10), respectively. Continuing this pattern in this work, we have decided to set $N_p = 5K_0$ and $N_g = \lceil K/2 \rceil$. Note that in this work, we only use even values for K_0, so N_p is also even. However, since the chromosomes are bred in pairs (two parents

create two children), there could be issues if K_0, and thus N_p, was odd. To remedy this, one could discard one of the children created during breeding, such as the child with the lowest fitness, or more simply just discard one of the final two created. Alternatively, one could enforce an even-numbered population by instead using, for example, $N_p = 2\times\lceil 5K_0/2 \rceil$.

5.3.2 Greedy Algorithms

Along with the GAs described above, we also consider two so-called "greedy" scheduling algorithms and their performance. Specifically, we consider the algorithms proposed by Sigdel and Krzymień in [141] for BD and [142] for SZF. These algorithms use a Frobenius-norm (F-norm) based metric in their scheduling decisions. The metrics consider a combination of channel gains and orthogonality. Specifically, at each step of the algorithm, the metric largely favors to be scheduled those users with the largest projection of their channel matrix into the null space of the users selected in earlier steps. In the BD algorithm, the projections of the previously selected users onto each potential user in the current step are also a factor in the metric. For the SZF algorithm, projections of each potential user to the null space of the aggregate channel (as given by (5.8)) for each prior step are also considered in the metric. Thus, the algorithms overall generally aim to select users whose channel matrices are the closest to orthogonal to each other. This way, data directed to a given user will already largely fall in the null space of the other users, and so the beamforming vectors for that user will be mostly aligned with the channel matrix of that user, which improves the system throughput.

The greedy BD and SZF algorithms share some similarities to an F-norm projection-based scheduling algorithm for BD proposed in [69]. However, the algorithms differ in two key areas. First, the scheduling metrics are less complex than those in [69]. Second, and most importantly, the greedy algorithms also use an intermediate grouping step to "filter out" some of the active users in the pool. The algorithms, in each step, first consider the projection of the channel matrices for each remaining user in the pool to the subspace of the channel matrices of those users already selected. If the spatial correlation of the users is not below a certain threshold ξ (i.e., the channels are not sufficiently close to orthogonal), those users are removed from the pool of users to select from, and do not proceed to have their full scheduling metric calculated. This intermediate grouping process can reduce the complexity of the scheduling process, but also creates a large

Table 5.2: Simplified greedy user scheduling algorithm for BD

1. Initialize: $i = 1$; $\mathcal{U} = \{1,2,\ldots,K\}$; $\mathcal{U}_s = \{\varnothing\}$.
   Find user $u_1 = \arg\max_{k\in\mathcal{U}} \|\mathbf{H}_k\|_F^2$.
   Set: $\mathcal{U}_s = \mathcal{U}_s \cup \{u_1\}$; $\mathbb{U}_1 = \mathcal{U} \setminus \{u_1\}$.
2. Set: $i = i + 1$.
   Find projector matrix: $\mathcal{P}_i^{\perp} = \mathbf{I}_{M_T} - \mathbf{V}_{i-1}^H \mathbf{V}_{i-1}$, where $\mathbf{V}_{i-1}$ is the row basis of $\mathbf{H}(\mathcal{U}_s)$,
   and $\mathbf{H}(\mathcal{U}_s) = \left[\mathbf{H}_{u_1}^T \; \mathbf{H}_{u_2}^T \cdots \mathbf{H}_{u_{i-1}}^T\right]^T$.
3. **if** $|\mathcal{U}_s| < K_0$ **then**
   Find $\mathbb{U}_i = \left\{ k \in \mathbb{U}_i, k \notin \mathcal{U}_s \left| \frac{\|\mathbf{H}_k \mathbf{V}_{i-1}^H\|_F}{\|\mathbf{H}_k\|_F \|\mathbf{V}_{i-1}\|_F} < \xi \right. \right\}$.
   if $|\mathbb{U}_i| > 0$ **then**
   Select user $u_i = \arg\max_{k\in\mathbb{U}_i} \left(\underbrace{\|\mathbf{H}_k \mathcal{P}_i^{\perp}\|_F^2}_{\text{backward projection}} + \underbrace{\sum_{k'\in\mathcal{U}_s} \|\mathbf{H}_{k'} \mathcal{P}_k^{\perp}\|_F^2}_{\text{forward projections}} \right)$, where $\mathcal{P}_k^{\perp} = \mathbf{I} - \hat{\mathbf{V}}_k^H \hat{\mathbf{V}}_k$,
   and $\hat{\mathbf{V}}_k$ is the row basis of $\mathbf{H}_k$.
   Set: $\mathcal{U}_s = \mathcal{U}_s \cup \{u_i\}$; $\mathbb{U}_i = \mathbb{U}_i \setminus \{u_i\}$.
   Go to Step 2.
   end if
   else Exit.

dependence of the performance of the algorithms on the value for the threshold ξ. [141] discusses the selection of the value for ξ in more detail.

In the case of SZF, there is also the matter of the order in which to encode the selected users. In [142], it is proposed to simply encode the users in the same order as they are selected by the greedy algorithm. Despite being suboptimal, the results of [142] indicate that ordering performs close to the optimal ordering obtained through an exhaustive search.

We outline the details of Sigdel and Krzymień's greedy algorithm (GrA) for BD in Table 5.2, while the GrA for SZF is described in Table 5.3. For notation, $\mathcal{U} = \{1,2,\ldots,K\}$ denotes the set of active users requesting service, $\mathcal{U}_s$ denotes the set of scheduled users, $\mathbb{U}_i$ denotes an intermediate set of users in step i, and $|\mathcal{U}_s|$ denotes the cardinality of the set $\mathcal{U}_s$.

Table 5.3: Simplified greedy user scheduling algorithm for SZF

1. Initialize: $i = 1$; $\mathcal{U} = \{1,2,\ldots,K\}$; $\mathcal{U}_s = \{\varnothing\}$.
   Find user $u_1 = \arg\max_{k\in\mathcal{U}} \|\mathbf{H}_k\|_F^2$.
   Set: $\mathcal{U}_s = \mathcal{U}_s \cup \{u_1\}$; $\mathbb{U}_1 = \mathcal{U} \setminus \{u_1\}$.
2. Set: $i = i + 1$.
   Define: $\bar{\mathbf{H}}(\mathcal{U}_s) = \left[\mathbf{H}_{u_1}^T\ \mathbf{H}_{u_2}^T \cdots \mathbf{H}_{u_{i-1}}^T\right]^T = \bar{\mathbf{U}}_i \bar{\mathbf{D}}_i \left[\bar{\mathbf{V}}_i^1\ \bar{\mathbf{V}}_i^0\right]^H$.
3. **if** $|\mathcal{U}_s| < K_0$ **then**

   Find $\mathbb{U}_i = \left\{ k \in \mathbb{U}_{i-1}, k \notin \mathcal{U}_s \left| \frac{\|\mathbf{H}_k \bar{\mathbf{V}}_i^1\|_F}{\|\mathbf{H}_k\|_F \|\bar{\mathbf{V}}_i^1\|_F} < \xi \right. \right\}$.

   if $|\mathbb{U}_i| > 0$ **then**

   Select user $u_i = \begin{cases} \arg\max_{k\in\mathbb{U}_i} \|\mathbf{H}_k \bar{\mathbf{V}}_i^0\|_F^2 & \text{if } i = 2, \\ \arg\max_{k\in\mathbb{U}_i} \dfrac{\|\mathbf{H}_k \bar{\mathbf{V}}_i^0\|_F^2}{\sum_{j=2}^{i-1} \|\mathbf{H}_k \bar{\mathbf{V}}_j^0\|_F^2} & \text{otherwise.} \end{cases}$

   Set: $\mathcal{U}_s = \mathcal{U}_s \cup \{u_i\}$; $\mathbb{U}_i = \mathbb{U}_i \setminus \{u_i\}$.
   Go to Step 2.
   end if
else Exit.

5.3.3 Hybrid Algorithms

Lastly, we consider two hybrid algorithms of the genetic and greedy algorithms described in the previous sections. The first is a seeded genetic algorithm, which we shall call Hybrid Algorithm 1 (HA1). Rather than initialize the chromosomes of the GA completely at random, instead, the first step of the GrA is performed to find the user[1] with the largest channel F-norm. This user is then seeded into the chromosomes of the initial population; that is, the bit corresponding to that user in the head of each chromosome is set to '1', forcing the GA to initially consider that user. The remaining users are selected randomly for each chromosome, as is the encoding order for those users in the case of SZF. The remainder of the operation of the GA is otherwise the same as previously described. This seeding was one of the possible ways to improve the GA performance mentioned in Chapter 3.

[1] It is also possible to seed more than one user (for example, the two users with the largest channel norms) into the chromosomes. However, as we show in Section 5.5.3, seeding just the top user is the best choice.

The second hybrid algorithm (HA2) starts by running the GrA as normal. However, rather than ending there, the algorithm then encodes those users (and encoding order, for SZF) into one chromosome, which replaces one of those randomly initialized at the start of the GA. The GA then runs, but for fewer generations than normal in order to help reduce the overall complexity. Due to elitism in the GA, the hybrid algorithm can perform no worse than the GrA alone.

There are a few details that can be adjusted in both of these hybrid algorithms. For instance, one can adjust how much to seed in the first algorithm, and how many generations to run the GA for in the second. We examine these details more closely later in Sections 5.5.3 and 5.5.4.

5.4 Complexity Analysis

In this section, we compare the complexity of the genetic and greedy algorithms in terms of the number of flops required. A flop is a real-valued floating point operation; an addition, multiplication, or division is each 1 flop. A complex-valued addition and multiplication take 2 and 6 flops, respectively. In general, most matrix operations require about an equal number of multiplications and additions. Thus, we assume that complex-valued operations need 4 times the flops as the real-valued ones. For the analysis, we assume $N_k = N$ for all k, $K_0 = \lceil M_T/N \rceil$, and that the algorithms schedule the maximum of K_0 users. Since $K_0 = \lceil M_T/N \rceil = M_T/N + \zeta$, for some constant $0 \le \zeta < 1$, K_0 grows with the same order of complexity as M_T/N.

5.4.1 Complexity of Various Matrix Operations

For an $m \times n$ complex-valued matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$, we list the complexity of various matrix operations required for the genetic and greedy scheduling algorithms.

- Multiplying an $m \times n$ matrix by an $n \times p$ matrix requires $8mnp$ flops [143][1].
- The squared F-norm $\|\mathbf{A}\|_F^2$ requires a total of $4mn$ flops [69].
- A Gram-Schmidt orthogonalization (GSO) of $\mathbf{A}$ uses $8m^2n - 2mn$ flops [69].
- The (inverse) p^{th} root $\mathbf{A}^{1/p}$ or $\mathbf{A}^{-1/p}$ of an $n \times n$ matrix requires about $\left(112 + \frac{4}{3}(p-1)\right)n^3$ flops [145]. In particular, the (inverse) square root will require $\frac{340}{3}n^3$ flops.

[1] Multiplying square $n \times n$ matrices can in fact be done with complexity $\mathcal{O}(n^{2.376})$ instead of $\mathcal{O}(n^3)$ [144]. However, we are uncertain if this complexity reduction still applies in general when multiplying rectangular matrices. Regardless, we show shortly that operations other than matrix multiplications dominate the order of complexity of the scheduling algorithms anyway.

- The determinant $|\mathbf{A}|$ of an $n \times n$ matrix is calculated by first performing an LU decomposition ($\mathbf{A} = \mathbf{LU}$), with a complexity of $\frac{8}{3}n^3$ flops [143]. The determinant is then the product of the n diagonal entries of $\mathbf{U}$. Thus, $|\mathbf{A}|$ has a total complexity of $\frac{8}{3}n^3 + 6n$ flops.
- A QR decomposition of an $m \times n$ matrix, $m \geq n$, to find $\mathbf{R} \in \mathbb{C}^{m \times n}$ and $\mathbf{Q} \in \mathbb{C}^{m \times m}$ requires a total of $16m^2n - 8mn^2 + \frac{8}{3}n^3$ flops [143].
- Waterfilling over j eigenmodes requires a maximum of $2j^2 + 6j$ flops [69].
- A full SVD ($\mathbf{A} = \mathbf{UDV}^H$) of an $m \times n$ matrix, $m \geq n$, requires $16m^2n + 32mn^2 + 36n^3$ flops [143]. If $m < n$, the complexity can be approximated by instead taking the SVD of $\mathbf{A}^H$. The $\mathbf{U}$ and $\mathbf{V}$ of $\mathbf{A}$ will be the $\mathbf{V}$ and $\mathbf{U}$ of $\mathbf{A}^H$, respectively. Furthermore, the complexity can be reduced if not all of $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ are required. For example, if only the singular values $\mathbf{D}$ are required, then only $16mn^2 - \frac{16}{3}n^3$ flops are needed [143].

The primary GA function of selecting users through crossover and mutation is mostly bit manipulation operations with the chromosomes, which are of negligible computational complexity. The complexity instead lies in calculating the utility function for the selection of users represented by each chromosome. On the other hand, the complexity of the greedy algorithm comes mainly from F-norm calculations and matrix multiplications.

5.4.2 Complexity of Genetic Algorithm for Block Diagonalization

The utility function (i.e., the fitness of each chromosome) is the sum-throughput for the selected users as given by Eqn. (5.6). However, it is not necessary to completely calculate the transmit covariance matrices to find the sum-throughput. The following steps are taken for each chromosome: First, for each of the K_0 users, find the null space basis vectors $\tilde{\mathbf{V}}_k^0$ of $\tilde{\mathbf{H}}_k$. Second, obtain the effective channel matrices $\mathbf{H}_{k,e}$ as given by Eqn. (5.5). Third, find the singular values of each $\mathbf{H}_{k,e}$. Fourth, waterfill over the eigenvalues of all scheduled users to get the power allocated to each eigenmode. Fifth, calculate the sum-throughput from the eigenvalues and power allocations. The complexity of each step is as follows.

Step 1: To find $\tilde{\mathbf{V}}_k^0$ for user k, it is not necessary to perform an SVD of the $(K_0 - 1)N \times M_T$ matrix $\tilde{\mathbf{H}}_k$. Instead, one may perform a QR decomposition of $\tilde{\mathbf{H}}_k^H$ to save on computational complexity. Let us express $\tilde{\mathbf{H}}_k^H = \mathbf{QR} = [\mathbf{Q}_1\ \mathbf{Q}_0]\mathbf{R}$. Then $\mathbf{Q}_0$, the rightmost $M_T - (K_0 - 1)N$ columns of $\mathbf{Q}$, are orthonormal basis vectors for the null space

of $\tilde{\mathbf{H}}_k$. The complexity of the K_0 QR decompositions is $K_0[\ 16M_T^2\left(K_0N-N\right)\ -\ 8M_T(K_0N-N)^2+\frac{8}{3}(K_0N-N)^3]$. After expansion and simplification, the complexity of the first step is $\mathcal{O}(M_T^3K_0)$.

Step 2: Finding the equivalent channel matrices involves K_0 multiplications of an $N\times M_T$ matrix with an $M_T\times(M_T-K_0N-N)$ matrix, for complexity $8K_0NM_T(M_T-K_0N+N)$ $\approx \mathcal{O}(8K_0NM_T^2\ -8K_0N^2M_T(M_T/N)+8K_0N^2M_T)=\mathcal{O}(M_TK_0N^2)$, as the first two terms cancel.

Step 3: Finding the singular values of each $N\times(M_T-K_0N+N)$ matrix $\mathbf{H}_{k,e}$ requires $16N(M_T-K_0N+N)^2-\frac{16}{3}(M_T-K_0N+N)^3$ flops. Thus, for K_0 users and their effective channel matrices, and replacing K_0N with $M_T+\zeta N$, we get a complexity of $\mathcal{O}(N^3K_0)$, as the M_T terms cancel.

Step 4: The rank of each of the effective matrices $\mathbf{H}_{k,e}$ is $r=M_T-K_0N+N$. Thus, there are a total of K_0r eigenmodes. First, K_0r multiplications are required to square the (real-valued) singular values into eigenvalues for $\mathbf{H}_{k,e}\mathbf{H}_{k,e}^H$, then waterfilling requires $2(K_0r)^2+6K_0r$ flops. As in the previous step, if we replace the K_0N in r with $M_T+\zeta N$, we find a total complexity of $\mathcal{O}(N^2K_0^2)\approx\mathcal{O}(M_TNK_0)$.

Step 5: The sum-throughput can be found by $\log_2\prod_{i=1}^{K_0r}\left(1+\lambda_ip_i/\sigma_n^2\right)$, where λ_i are the eigenvalues and p_i are the associated waterfilling power allocations for each eigenvalue from Step 4. This requires 1 addition, multiplication, and division for each of the K_0r terms, K_0r more multiplications for the product of the terms, and a $\log_2$ calculation, whose complexity can be neglected here[1]. Thus, the final step requires $4K_0r$ flops, which is of order $\mathcal{O}(NK_0)$.

Thus, one fitness calculation is $\mathcal{O}(M_T^3K_0)+\mathcal{O}(M_TK_0N^2)+\mathcal{O}(N^3K_0)+\mathcal{O}(M_TNK_0)+$ $\mathcal{O}(NK_0)$. Overall, based on the highest order term, this is therefore $\mathcal{O}(M_T^3K_0)$. The GA calculates this metric $N_p\times N_g$ times. Recall from earlier that we use $N_p=5K_0$ and $N_g=\lceil K/2\rceil$ for the GA. Thus, the entire GA scheduling process for BD is $\mathcal{O}(KK_0^2M_T^3)$.

[1] Strictly speaking, the $\log_2$ operation is not even necessary. We could have also used the product in Step 5 as the fitness for the GA, since the log function is monotonically increasing. However, since the log function is of a much lower order of complexity (a single calculation for each chromosome) compared to everything else, it is trivial to include it.

5.4.3 Complexity of Genetic Algorithm for Successive Zero-Forcing

This utility function (fitness) for each chromosome is the sum rate for the selected users and encoding order as given by Eqns. (5.10) and (5.11). For ease of notation, we assume without loss of generality that $\pi(j) = j$. To calculate the sum rate, the following steps are taken: First, find the covariance matrices $\mathbf{P}_i$ for the dual MAC with the iterative algorithm in [107]. Second, convert the MAC matrices $\mathbf{P}_i$ to BC matrices $\mathbf{\Sigma}_i$ for DPC, as in [24]. Third, convert the DPC matrices $\mathbf{\Sigma}_i$ to SZF covariance matrices $\mathbf{Q}_i$ as in [50]. Fourth, calculate the sum rate from (5.10) and (5.11). More details for each step can be found in Appendix D. The complexity of each step is as follows.

Step 1: The MAC covariance $\mathbf{P}_i$ matrices are first initialized to some value. Then, during each iteration of the algorithm, for each user i, an effective channel matrix $\mathbf{G}_i \in \mathbb{C}^{N \times M_T}$ is calculated by $\mathbf{G}_i = \mathbf{H}_i \left(\mathbf{I} + \sum_{j \neq i} \mathbf{H}_j^H \mathbf{P}_j \mathbf{H}_j \right)^{-1/2}$. For all K_0 users, $\mathbf{H}_j^H \mathbf{P}_j \mathbf{H}_j$ can be calculated and stored, using $8K_0(M_T N^2 + M_T^2 N)$ flops. For the first user, the matrix $\mathbf{Z}_i = \mathbf{I} + \sum_{j \neq i} \mathbf{H}_j^H \mathbf{P}_j \mathbf{H}_j$ can be calculated using $M_T + (K_0 - 1)(2 M_T^2)$ flops. For the remaining $K_0 - 1$ users, $\mathbf{Z}_i$ can be calculated recursively by $\mathbf{Z}_{i+1} = \mathbf{Z}_i + \mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i - \mathbf{H}_{i+1}^H \mathbf{P}_{i+1} \mathbf{H}_{i+1}$, using a total of $(K_0 - 1)(4 M_T^2)$ flops. Finding the inverse square root of each $M_T \times M_T$ matrix $\mathbf{Z}_i$ requires $\frac{340}{3} M_T^3$ flops. The final multiplication by $\mathbf{H}_i$ requires an additional $8 M_T^2 N$ flops per user. Out of this entire operation, it can be seen the highest order calculation is that of the inverse square root. Thus, the calculation of $\mathbf{G}_i$ for K_0 users is $\mathcal{O}(M_T^3 K_0)$.

Once the effective channel matrices are computed, new covariance matrices are calculated from the block-diagonal channel formed from $blkdiag\left(\mathbf{G}_1, \cdots, \mathbf{G}_{K_0} \right)$. First, the $N \times N$ matrices $\mathbf{G}_i \mathbf{G}_i^H$ are formed using a total of $8K_0 M_T N^2$ flops. Then, an SVD is performed to obtain $\mathbf{G}_i \mathbf{G}_i^H = \mathbf{U}_i \mathbf{D}_i \mathbf{V}_i^H$. However, since only the matrices $\mathbf{U}_i$ and $\mathbf{D}_i$ are required from the SVD, only $48N^3$ flops are required per SVD instead of $84N^3$ if we also needed $\mathbf{V}_i$ [143]. There are a total of $K_0 N$ eigenvalues λ_i; waterfilling over the eigenmodes thus requires $2(K_0 N)^2 + 6K_0 N$ flops. From the resulting eigenmode power allocations p_i, the sum rate can be found (for the purposes of determining algorithm convergence) with $4K_0 N$ flops by $\log_2 \prod_{i=1}^{K_0 N} \left(1 + \lambda_i p_i / \sigma_n^2 \right)$. Updated covariance matrices

are found from $\mathbf{S}_i = \mathbf{U}_i \mathbf{\Lambda}_i \mathbf{U}_i^H$, where the $\mathbf{\Lambda}_i$ matrices are formed from the eigenmode power allocations. Since the $\mathbf{\Lambda}_i$ are real diagonal matrices, $\mathbf{S}_i$ can each be found using $8N^3 + 2N^2$ flops. Finally, the $\mathbf{P}_i$ (in the general case) are updated for the next iteration $n+1$ by $\mathbf{P}_i^{(n+1)} = \frac{1}{K_0}\mathbf{S}_i + \frac{K_0 - 1}{K_0}\mathbf{P}_i^{(n)}$, using a total of $6K_0N^2$ flops. For this entire updating procedure, the highest order calculation is multiplying $\mathbf{G}_i\mathbf{G}_i^H$, which is $\mathcal{O}(K_0 M_T N^2)$.

Thus, for one iteration of the MAC iterative waterfilling algorithm, it is clear that the highest order calculation is that of the inverse square root. Thus, the entire algorithm has complexity $\alpha\mathcal{O}(M_T^3 K_0)$[1], where α is the number of iterations required for the algorithm to converge sufficiently for the purposes of scheduling. From the figures in [107], $\alpha = 5$ is usually sufficient for the algorithm to converge very close to the sum rate, and should be enough for our scheduling requirements.

Step 2: For the MAC to BC calculations, we have $\mathbf{A}_j = \mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j-1} \mathbf{\Sigma}_i \right) \mathbf{H}_j^H \in \mathbb{C}^{N \times N}$ and $\mathbf{B}_j = \mathbf{I} + \sum_{i=j+1}^{K_0} \mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i \in \mathbb{C}^{M_T \times M_T}$, where $\mathbf{\Sigma}_i$ are the DPC BC covariance matrices. All $\mathbf{B}_j$ can be recursively calculated, starting with $\mathbf{B}_{K_0} = \mathbf{I}_{M_T}$. The remaining $K_0 - 1$ matrices require $8M_TN^2 + 8M_T^2N + 2M_T^2$ flops each to multiply $\mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i$ and add to the previously calculated $\mathbf{B}_j$. A similar running sum can also be kept for the $\mathbf{\Sigma}_i$ (once they have been calculated) in $\mathbf{A}_j$. Each $\mathbf{A}_j$ for $j > 1$ (as $\mathbf{A}_1 = \mathbf{I}$) then requires $2M_T^2 + 8M_T^2N + 8M_TN^2 + N$ flops to calculate. Calculation of $\mathbf{A}_j^{1/2}$ and $\mathbf{A}_j^{-1/2}$, $j > 1$, each require $\frac{340}{3}N^3$ flops, while each $\mathbf{B}_j^{-1/2}$, $j < K_0$, requires $\frac{340}{3}M_T^3$ flops. The product $\mathbf{B}_j^{-1/2}\mathbf{H}_j^H\mathbf{A}_j^{-1/2} \in \mathbb{C}^{M_T \times N}$ generally requires $8M_T^2N + 8M_TN^2$ flops, except for $j = 1$ and K_0, where knowing $\mathbf{A}_1$ and $\mathbf{B}_{K_0}$ (and hence their inverse square roots) are identity matrices reduces the complexity to just $8M_T^2N$ and $8M_TN^2$, respectively. Finding the SVD $\mathbf{B}_j^{-1/2}\mathbf{H}_j^H\mathbf{A}_j^{-1/2} = \mathbf{F}_j\mathbf{\Lambda}_j\mathbf{G}_j^H$, where $\mathbf{F}_j \in \mathbb{C}^{M_T \times N}$ and $\mathbf{G}_j \in \mathbb{C}^{N \times N}$, requires $16M_T^2N + 32M_TN^2 + 36N^3$ flops for each j. $\mathbf{\Sigma}_j$ is then found by $\mathbf{\Sigma}_j = \mathbf{B}_j^{-1/2}\mathbf{F}_j\mathbf{G}_j^H\mathbf{A}_j^{1/2}\mathbf{P}_j\mathbf{A}_j^{1/2}\mathbf{G}_j\mathbf{F}_j^H\mathbf{B}_j^{-1/2}$. Let $\mathbf{T}_j = \mathbf{A}_j^{1/2}\mathbf{G}_j\mathbf{F}_j^H\mathbf{B}_j^{-1/2}$, and note that $\mathbf{B}_j$ and $\mathbf{A}_j$ (and their (inverse) square roots) are all Hermitian. Calculating $\mathbf{T}_j$ requires $8M_T^2N + 8M_TN^2$ flops to compute at $j = 1$, $8N^3 + 8M_TN^2$ flops at $j = K_0$, and $8M_T^2N +$

[1] This would therefore also be the order of complexity for calculating the fitness of a GA chromosome for maximum throughput scheduling under DPC.

$8M_TN^2 + 8N^3$ flops otherwise. Then, $\mathbf{\Sigma}_j = \mathbf{T}_j^H \mathbf{P}_j \mathbf{T}_j$, requiring a further $8M_T^2 N + 8M_TN^2$ flops to compute. Overall, it can be seen that the highest order term in the MAC to BC transformations is the $\frac{340}{3} M_T^3$ flops required to calculate $\mathbf{B}_j^{-1/2}$. $(K_0 - 1)$ of these roots are calculated, so the complexity of Step 2 overall is $\mathcal{O}(M_T^3 K_0)$.

Step 3: In converting the BC covariance matrices $\mathbf{\Sigma}_j$ to SZF matrices $\mathbf{Q}_j$, first, $\mathbf{Q}_1 = \mathbf{\Sigma}_1$. For the remaining $K_0 - 1$ users, $\bar{\mathbf{V}}_{j-1}^0 \in \mathbb{C}^{M_T \times (M_T - (j-1)N)}$ must be calculated for each of the aggregate matrices $\bar{\mathbf{H}}_{j-1} \in \mathbb{C}^{(j-1)N \times M_T}$. Like for the BD null space vectors, this can be done with a QR decomposition instead of an SVD. The complexity of these calculations are $\sum_{j=2}^{K_0} [\, 16M_T^2 N\,(j-1) - 8M_TN^2(j-1)^2 + \frac{8}{3} N^3(j-1)^3] = \sum_{n=1}^{K_0-1} [\, 16M_T^2 Nn - 8M_TN^2n^2 + \frac{8}{3} N^3n^3] = 8M_T^2 NK_0(K_0-1) - \frac{8}{6} M_TN^2(K_0-1)(K_0)(2K_0-1) + \frac{8}{3} N^3[(K_0-1)K_0/2]^2$. Recall that K_0 has the same order as M_T/N, or equivalently, N has the same order as M_T/K_0. Substituting $N = M_T/K_0$ in the above[1], we obtain a complexity of $6K_0 M_T^3 - \frac{16}{3} M_T^3 - \frac{2}{3} M_T^3 K_0^{-1}$, so the order of complexity of finding the null space vectors is $\mathcal{O}(M_T^3 K_0)$.

Next, $K_0 - 1$ products $\bar{\mathbf{V}}_{j-1}^0 \left(\bar{\mathbf{V}}_{j-1}^0\right)^H$ must be calculated, with each product using $8M_T^2 [M_T - (j-1)N]$ flops. The products thus use a total of $\sum_{j=2}^{K_0} 8M_T^2 [M_T - (j-1)N] = \sum_{n=1}^{K_0-1} [\, 8M_T^3 - 8M_T^2 nN] = 8M_T^3 (K_0-1) - 4M_T^2 NK_0(K_0-1)$ flops. Again substituting $N = M_T/K_0$, we get a complexity of $4M_T^3 K_0 - 4M_T^3$, for an order of $\mathcal{O}(M_T^3 K_0)$. For users $j = 2$ to $K_0 - 1$, $\mathbf{Q}_j = \bar{\mathbf{V}}_{j-1}^0 \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{\Sigma}_j \bar{\mathbf{V}}_{j-1}^0 \left(\bar{\mathbf{V}}_{j-1}^0\right)^H$, resulting in a further total complexity of $16(K_0 - 2) M_T^3$ flops. For the final user $j = K_0$, first the effective channel matrix $\mathbf{H}_{eff} = \left(\mathbf{I} + \mathbf{H}_{K_0} \left(\sum_{j=1}^{K_0-1} \mathbf{Q}_j\right) \mathbf{H}_{K_0}^H\right)^{-1/2} \mathbf{H}_{K_0} \bar{\mathbf{V}}_{K_0-1}^0 \left(\bar{\mathbf{V}}_{K_0-1}^0\right)^H$ is calculated. This involves $2M_T^2 (K_0 - 1)$ flops to add the $\mathbf{Q}_j$, $N + 8M_T^2 N + 8M_T N^2$ flops to form $\mathbf{I} + \mathbf{H}_{K_0} \left(\sum_{j=1}^{K_0-1} \mathbf{Q}_j\right) \mathbf{H}_{K_0}^H$, $\frac{340}{3} N^3$ flops to find the inverse square root, and $8M_T^2 N + 8M_TN^2$ flops to multiply the root by $\mathbf{H}_{K_0} \bar{\mathbf{V}}_{K_0-1}^0 \left(\bar{\mathbf{V}}_{K_0-1}^0\right)^H$. Once $\mathbf{H}_{eff}$ is found, first $\mathbf{H}_{eff}^H \mathbf{H}_{eff}$ is calculated using $8M_T^2 N$ flops, then the product is waterfilled with power constraint $P - \sum_{j=1}^{K_0-1} Tr(\mathbf{Q}_j)$ to find a

[1] Strictly, this should be $N = M_T/(K_0 - \zeta)$, but neglecting ζ does not affect the order of complexity.

temporary covariance matrix $\bar{\mathbf{Q}}_{K_0}$. Calculating the power constraint requires $(K_0 - 1)M_T$ flops, since the diagonal elements of $\mathbf{Q}_j$ are real. The waterfilling involves an SVD requiring **U** and **D**, but not **V**, so $48\,M_T^3$ flops are needed. **D** has N non-zero eigenvalues (since $\mathbf{H}_{eff}$ is $N \times M_T$), so the waterfilling takes $2N^2 + 6N$ flops. Then $\bar{\mathbf{Q}}_{K_0} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H$, where $\mathbf{\Lambda}$ is a real diagonal matrix formed from the waterfilling eigenmode power allocations, requiring $8\,M_T^3 + 2M_TN$ flops, exploiting the structure of $\mathbf{\Lambda}$ to reduce complexity. Finally, $\mathbf{Q}_{K_o} = \bar{\mathbf{V}}^0_{K_0-1}\left(\bar{\mathbf{V}}^0_{K_0-1}\right)^H \bar{\mathbf{Q}}_{K_0} \bar{\mathbf{V}}^0_{K_0-1}\left(\bar{\mathbf{V}}^0_{K_0-1}\right)^H$ is found using $16\,M_T^3$ flops. Within Step 3, there are several calculations that require $\mathcal{O}(M_T^3 K_0)$, so this is the overall complexity of this step.

Step 4: The SZF sum rate of Eqns. (5.10) and (5.11) can be rewritten as $\log_2 \prod_{j=1}^{K_0} \left[\left| \mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j} \mathbf{Q}_i \right) \mathbf{H}_j^H \right| \Big/ \left| \mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j-1} \mathbf{Q}_i \right) \mathbf{H}_j^H \right| \right]$. The determinant in the numerator must be calculated K_0 times, while the denominator must be found $K_0 - 1$ times (the value at $j = 1$ is 1). A running sum of the $\mathbf{Q}_i$ can be used for each j; overall, this sum will require $2K_0\,M_T^2$ flops. Having the sum, the terms within the determinant functions take $8\,M_T^2 N + 8M_TN^2 + N$ flops to calculate for each j. Finding the determinant of those $N \times N$ terms then requires $\frac{8}{3}N^3 + 6N$ flops. Finally, having the (real) determinant values, dividing them for each j and finding the overall product would require an additional $2K_0 - 1$ flops, plus one $\log_2$ operation, which we ignore. Thus, the calculation of the SZF sum rate requires $(2K_0 - 1)(8\,M_T^2 N + 8M_TN^2 + 7N + \frac{8}{3}N^3 + 1) + 2K_0\,M_T^2$ flops. Hence, Step 4 has an overall complexity of $\mathcal{O}(M_T^2 K_0 N)$.

Thus, one fitness calculation requires $\mathcal{O}(M_T^3 K_0)$, from Steps 1 to 3. The GA calculates this metric $N_p \times N_g$ times, where $N_p = 5K_0$ and $N_g = \lceil K/2 \rceil$. Thus, the entire GA scheduling process for SZF is $\mathcal{O}(KK_0^2 M_T^3)$. This is the same as for BD, but it is clear from the overall description that the SZF scheduling certainly must have a larger constant in front of that $KK_0^2 M_T^3$ term.

5.4.4 Complexity of Greedy Algorithm for Block Diagonalization

In this section we determine the complexity of the BD GrA. This work was done in collaboration with Dr. Shreeram Sigdel [132]. The complexity of each step is as follows.

Step 1: The F-norm of the $N \times M_T$ channel matrix is calculated for each of the K users, requiring $4KNM_T$ flops.

Step 2: For $i \geq 2$, the projection matrix requires a GSO of the $(i-1)N \times M_T$ matrix $\mathbf{H}(\mathcal{U}_s)$ to obtain the row basis $\mathbf{V}_{i-1}$, then a multiplication and subtraction to get $\mathcal{P}_i^{\perp} = \mathbf{I}_{M_T} - \mathbf{V}_{i-1}^H \mathbf{V}_{i-1}$. The total number of flops required is $8(i-1)^2N^2M_T - 2(i-1)NM_T + 8(i-1)NM_T^2 + M_T = 8(i-1)^2N^2M_T + (i-1)(8NM_T^2 - 2NM_T) + M_T$.

Step 3: Finding the $\left\|\mathbf{H}_k \mathbf{V}_{i-1}^H\right\|_F$ term uses a multiplication and F-norm and thus requires $(8M_T + 4)N^2(i-1)$ flops. $\left\|\mathbf{H}_k\right\|_F$ can be reused from Step 1, at the cost of some memory storage. $\left\|\mathbf{V}_{i-1}^H\right\|_F$ has negligible complexity, since its rows are orthonormal; hence $\left\|\mathbf{V}_{i-1}^H\right\|_F = \sqrt{(i-1)N}$, which is negligible to compute, as is the division for the normalization. This correlation calculation is done for each of the $|\mathbb{U}_i|$ users. Thus, the complexity of the intermediate grouping is $|\mathbb{U}_i|(8M_T + 4)N^2(i-1)$ flops. The projection matrices $\mathcal{P}_k^{\perp}$ only need to be calculated one time at $i = 2$, then can be reused for larger i. Each of the $|\mathbb{U}_2|$ projection matrices requires a GSO to find $\hat{\mathbf{V}}_k$, a matrix multiplication, and a subtraction from the identity, yielding a complexity of $|\mathbb{U}_2|(8N^2M_T - 2NM_T + 8NM_T^2 + M_T)$ flops. The forward and backward projections combined require a total of i matrix multiplications, F-norm calculations, and real additions. The projections are done for each of the $|\mathbb{U}_i|$ users, so the total complexity is $i|\mathbb{U}_i|(8NM_T^2 + 4NM_T + 1)$ flops.

Thus, the total complexity of the greedy algorithm is $4KNM_T + |\mathbb{U}_2|(8N^2M_T - 2NM_T + 8NM_T^2 + M_T) + \sum_{i=2}^{K_0}\{8(i-1)^2N^2M_T + (i-1)(8NM_T^2 - 2NM_T) + M_T + |\mathbb{U}_i| \times [(i-1)(8M_TN^2 + 4N^2) + i(8NM_T^2 + 4NM_T + 1)]\}$. After simplification, it can be found that the complexity of the GrA for BD is $\mathcal{O}(M_T^2N\sum_{i=2}^{K_0} i|\mathbb{U}_i|)$. $|\mathbb{U}_i|$ is a random variable in the range $0 \leq |\mathbb{U}_i| \leq (K-i+1)$, and its value depends on the threshold ξ. In the worst case, the greedy algorithm must search over $(K-i+1)$ users in Step 3. (The most likely reason

for this to occur is if $\xi = 1$.) Thus, although the search is generally simpler, the worst-case complexity of the BD GrA is $\mathcal{O}(M_T^2 N \sum_{i=2}^{K_0} i(K-i+1))$. After further simplification, we find that the GrA complexity is $\mathcal{O}(KK_0^2 M_T^2 N) \approx \mathcal{O}(KK_0 M_T^3)$.

Thus, overall, the GrA complexity for BD is lower than that of the GA by a factor of K_0. However, it should be noted that the GrA will still require one calculation of (5.6) to find the sum rate for selected users, whereas for the GA, most of that calculation has already been carried out. This is in essence the calculation for a single GA chromosome, so the GrA must carry out the one additional step of complexity order $\mathcal{O}(K_0 M_T^3)$.

In comparison, the complexity of an exhaustive search is found by calculating (5.6) for all possible groups of K_0 out of K users. Its complexity is thus $\mathcal{O}\left(\binom{K}{K_0} K_0 M_T^3\right)$ [69].

5.4.5 Complexity of Greedy Algorithm for Successive Zero-Forcing

In this section, we find the complexity of the GrA for SZF. The complexity of each step is as follows.

Step 1: This step is identical to Step 1 of the BD GrA; it requires $4KNM_T$ flops.

Step 2: For $i \geq 2$, the matrix $\bar{\mathbf{H}}(\mathcal{U}_s)$ is $(i-1)N \times M_T$. Performing an SVD of this matrix requires $16 M_T^2 N(i-1) + 32 M_T N^2 (i-1)^2 + 36 N^3 (i-1)^3$ flops.

Step 3: The calculation of the correlation for the intermediate grouping is the same as for the BD GrA. It thus has complexity $|\mathbb{U}_{i-1}|(8M_T+4)N^2(i-1)$ flops. For the user selection metric, the denominator need not be fully calculated for each i. Instead, a running sum can be kept and updated with each i, at the expense of the storage of at most $|\mathbb{U}_2|$ real scalars. At each i, the term $\left\|\mathbf{H}_k \bar{\mathbf{V}}_i^0\right\|_F^2$ requires $(8M_T+4)N[M_T-(i-1)N]$ flops to compute the multiplication and F-norm for each user in $|\mathbb{U}_i|$. Then, 1 flop is required to divide by the sum in the denominator, followed by 1 flop to update the sum for the next i; we neglect these final 2 flops. Thus, the complexity of Step 3 is $|\mathbb{U}_i|(8M_T+4)N \times [M_T-(i-1)N]$ flops.

Therefore, the total complexity of the greedy algorithm for SZF is $4KNM_T + \sum_{i=2}^{K_0} \{ 16 M_T^2 N(i-1) + 32 M_T N^2 (i-1)^2 + 36 N^3 (i-1)^3 + |\mathbb{U}_{i-1}|(8M_T+4)N^2(i-1) + |\mathbb{U}_i| \times (8M_T+4)N[M_T-(i-1)N] \}$. After some simplification of the above equation, we find the

highest order terms are $(8M_T+4)N^2 \sum_{n=1}^{K_0-1}[\, n\,|\mathbb{U}_i| - n\,|\mathbb{U}_{n+1}|\,]$ and $(8M_T+4)M_TN\times \sum_{i=2}^{K_0}|\mathbb{U}_i|$. Expanding the sum in the first term gives $\sum_{n=1}^{K_0-1}[\, n|\mathbb{U}_i| - n|\mathbb{U}_{n+1}|\,] = (K_0-1)\times |\mathbb{U}_{K_0}| + \sum_{n=1}^{K_0-1}|\mathbb{U}_n|$. This makes the first term equal to $(8M_T+4)N^2(K_0-1)\,|\mathbb{U}_{K_0}| + (8M_T+4)N^2 \sum_{n=1}^{K_0-1}|\mathbb{U}_n|$. Thus, the order of complexity of the two terms is $\mathcal{O}(M_TN^2K_0|\mathbb{U}_{K_0}| + M_TN^2\sum_{n=1}^{K_0-1}|\mathbb{U}_n|)$ and $\mathcal{O}(M_T^2N\sum_{i=2}^{K_0}|\mathbb{U}_i|)$, respectively. The first half of the first term is approximately equivalent to $\mathcal{O}(M_T^2N|\mathbb{U}_{K_0}| + \ldots)$. Since $M_T > N$, the second term is of higher order, so the entire GrA for SZF has a complexity of $\mathcal{O}(M_T^2N\sum_{i=2}^{K_0}|\mathbb{U}_i|)$. Just as with the GrA for BD, $|\mathbb{U}_i|$ is a random variable in the range $0 \le |\mathbb{U}_i| \le (K-i+1)$, whose value depends on the threshold ξ. Thus, in the worst case, $|\mathbb{U}_i| = (K-i+1)$, and the order of complexity becomes $\mathcal{O}(M_T^2N\sum_{i=2}^{K_0}(K-i+1))$. After further simplification, we finally find the worst-case complexity for the GrA is $\mathcal{O}(KK_0M_T^2N) \approx \mathcal{O}(KM_T^3)$.

Thus, overall, the GrA complexity for SZF is lower than that of the GA by a factor of K_0^2. However, just like for BD, the GrA must afterwards find the covariance matrices and sum rate for the selected users and encoding order, whereas the GA has already done this. This additional step has complexity order $\mathcal{O}(K_0M_T^3)$, the same as a single SZF GA chromosome fitness calculation.

In comparison, the complexity of an exhaustive search is found by finding the sum rate for all possible groups and encoding orders of up to K_0 out of K users. Its complexity is thus $\mathcal{O}\left(K_0!\binom{K}{K_0}K_0M_T^3\right) \approx \mathcal{O}\left(K^{K_0}K_0M_T^3\right)$.

5.4.6 Complexity of Hybrid Algorithms

As already seen in the previous two sections, the complexity of finding the user with the best channel F-norm is $4KNM_T$. The idea of the first hybrid algorithm (HA1) is to use that user as a seed in the GA chromosome, but otherwise operate the GA as normal. The complexity of finding that user is of lower order than the overall GA. Thus, the order of complexity of HA1 is $\mathcal{O}(KK_0^2M_T^3)$ for both BD and SZF, the same as for the GA.

Table 5.4: Summary of the complexity orders of different user scheduling algorithms

Algorithm	Complexity Order (BD)	Complexity Order (SZF)
Greedy	$\mathcal{O}\left(M_T^2 N \sum_{i=2}^{K_0} i\lvert\mathbb{U}_i\rvert\right)$	$\mathcal{O}\left(M_T^2 N \sum_{i=2}^{K_0} \lvert\mathbb{U}_i\rvert\right)$
Greedy (worst case)	$\mathcal{O}\left(KK_0^2 M_T^2 N\right) \approx \mathcal{O}\left(KK_0 M_T^3\right)$	$\mathcal{O}\left(KK_0 M_T^2 N\right) \approx \mathcal{O}\left(KM_T^3\right)$
Genetic	$\mathcal{O}\left(KK_0^2 M_T^3\right)$	$\mathcal{O}\left(KK_0^2 M_T^3\right)$
Hybrid 1	$\mathcal{O}\left(KK_0^2 M_T^3\right)$	$\mathcal{O}\left(KK_0^2 M_T^3\right)$
Hybrid 2	$\mathcal{O}\left(KK_0 M_T^3\right)$	$\mathcal{O}\left(KM_T^3\right)$
Exhaustive Search	$\mathcal{O}\left(\binom{K}{K_0} K_0 M_T^3\right)$	$\mathcal{O}\left(K_0!\binom{K}{K_0} K_0 M_T^3\right) \approx \mathcal{O}\left(K^{K_0} K_0 M_T^3\right)$

The worst-case complexity of the GrA was found to be $\mathcal{O}(KK_0M_T^3)$ for BD and $\mathcal{O}(KM_T^3)$ for SZF. To reduce the complexity of HA2, the idea is that rather than run the GA for a number of generations which is dependent on K, instead we run for a constant number of generations after running the GrA. This removes the factor of K from the order of complexity of the GA, making its order less than that of the GrA. The order of complexity for the GA portion would be $\mathcal{O}(K_0^2M_T^3)$. Hence, the highest order term, and thus the overall order of complexity for HA2, is the same as for the GrA, i.e., $\mathcal{O}(KK_0M_T^3)$ for BD and $\mathcal{O}(KM_T^3)$ for SZF, in the worst case. Obviously, the total complexity will be higher than the GrA alone. However, if the generations for the GA are restricted, the added computations should not be very significant.

We summarize the complexity orders of all the algorithms in Table 5.4.

5.5 Simulation Results

In this section, we present simulation results demonstrating the performance of our scheduling algorithms. A performance comparison of the exhaustive search, genetic algorithm (GA), and greedy algorithm (GrA) for both BD and SZF are presented first. The simulations and examination of the greedy algorithms were done in collaboration with Dr. Shreeram Sigdel [132],[133],[134]. For the GrA, the optimal correlation threshold ξ (that maximizes the sum rate) was used, and determined as in [141] through simulation. The optimal case exhaustively searches through all possible user combinations (and encoding orders in case of SZF). For reference, when examining the BD performance, we also consider the sum rate for the two scheduling algorithms

proposed in [69]. We label these algorithms "SCAHE" after the initials of the authors' names. One of the two, based on F-norms and projections, has already been mentioned. The BD GrA is similar to this algorithm, but is less complex; the SCAHE metric is more complicated (its forward projections for each $k' \in \mathcal{U}_s$ (see Table 5.2) also consider the null spaces of the other previously scheduled users) and the algorithm considers all active users (i.e., no threshold is used). The second SCAHE algorithm is based on capacity. During each iteration, the algorithm selects the user from those remaining in the pool that maximizes the sum rate when scheduled together with the users selected in the previous iterations. The two algorithms both have complexity $\mathcal{O}(KK_0^2M_T^3)$ [69]. Later, we examine the performance of the two hybrid algorithms.

5.5.1 Block Diagonalization

Figure 5.1 shows the performance versus the number of active users K of the scheduling algorithms for a case where $M_T = 4$, $N_k = N = 2$ for all K users, and $K_0 = 2$, at an SNR of 5 dB and 10 dB, where the SNR is defined as P/σ_n^2 (see Eqn. (5.6)). It is observed that the GrA performs better than the GA for low SNR, but the GA outperforms the GrA at higher SNR. The reasons for this are as follows. Since the GrA starts by selecting the user with the maximum channel F-norm, the transmit power allocated to that strongest user will be the highest. The second user may in fact receive much less power than the first user (in the case of two selected users presented here). At low SNR (as P approaches 0), scheduling a single user can become optimal, resulting in a better sum rate than scheduling two users. In such a case, zero-forcing across users is not beneficial as the gain of the projected channel of the second selected user will be low. (In other words, $\mathbf{H}_{eff}$ for the second user will be poor.) On the other hand, it is by no means guaranteed that the GA will select the user with the maximum channel gain, due to its stochastic nature. This reduces the GA throughput at lower SNR. Both algorithms perform well compared to the optimal performance of the exhaustive search, achieving about 93−98% of the best possible throughput, with a gap less than 1 bit/s/Hz from optimal. We have performed additional simulations at 0, 15, and 20 dB, which we relegate to Appendix E for space and presentation considerations. These supplementary results indicate that the GrA continues to be better at the lower SNR of 0 dB, while at 15 and 20 dB, the GA

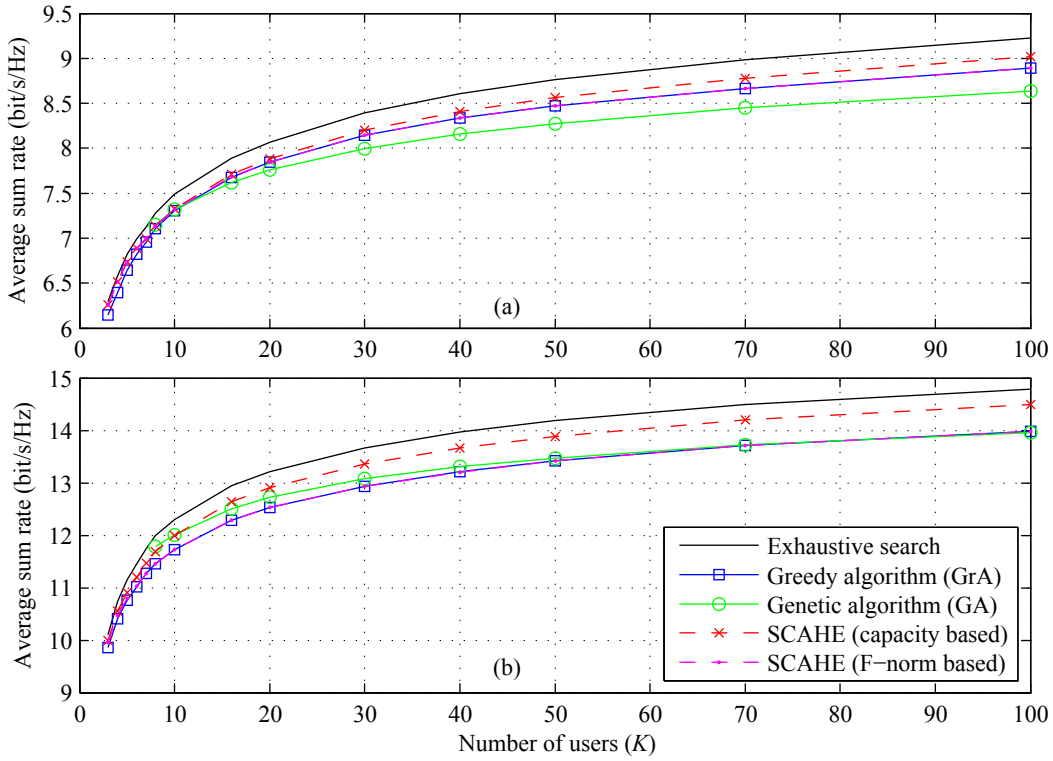

Figure 5.1: Performance vs. K of exhaustive search, greedy, genetic, and SCAHE scheduling algorithms for BD; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. (a) SNR = 5 dB, and (b) 10 dB.

shows further improvement over the GrA[1]. For reference, we also show results for the SCAHE algorithms. It can be seen that the GrA performs essentially identically to the SCAHE F-norm algorithm at all SNR levels. However, the SCAHE capacity algorithm still performs better than both the GrA and GA. We note, though, that the complexity of the GrA is lower than the GA and both SCAHE algorithms by a factor of K_0, as discussed in Section 5.4.4.

Similarly, we perform the simulations for $M_T = 8$, $N_k = N = 2$, and $K_0 = 4$; the results for SNR = 5 dB, 10 dB and 15 dB are given in Figure 5.2. Further results for 0 dB and 20 dB are in Appendix E. We first note that the results are different than in our work in [132]. In that earlier work, the results for the GrA were erroneously for a non-optimized threshold in Step 3 of the algorithm (i.e., using $\xi = 1$). With the optimized threshold in

[1] The performance of current cellular systems is limited by out-of-cell interference; while the interference-free SNR might be high, the SINR can be much lower. While some gains in spectral efficiency are still possible from MIMO spatial multiplexing, they are nowhere near as significant as the gains at high SNR / SINR. However, MIMO system designs employing network coordination (e.g. [146]) can mitigate or eliminate much of this intercell interference, allowing the system to indeed operate in the high-SNR regime. Thus, our simulation results at higher SNRs may have the most relevance for future MIMO systems. (For example, [146] considers an SNR of 18 dB.)

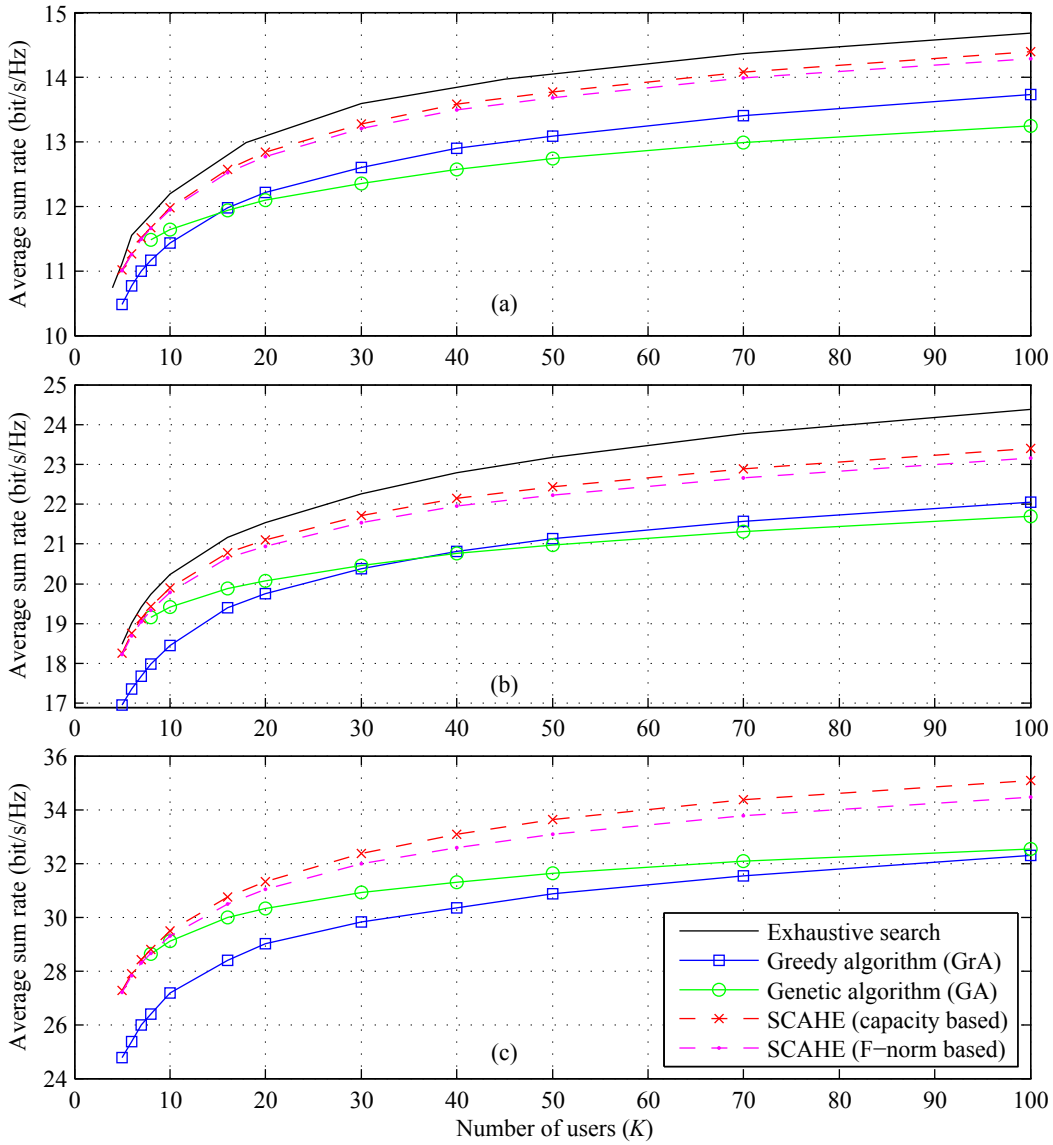

Figure 5.2: Performance vs. K of exhaustive search, greedy, genetic, and SCAHE scheduling algorithms for BD; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) SNR = 5 dB, (b) 10 dB, and (c) 15 dB.

this work, the GrA performance is improved, and hence the analysis of the simulation results leads to somewhat different conclusions.

We observe a similar trend as in the $M_T = 4$ case. Namely, the GrA has a better performance at low SNR, but the GA has a better performance at higher SNR. Interestingly, we also now more clearly observe a crossover in performance as the SNR increases. Specifically, at 10 dB, the GA performs better at low K, while the GrA performs better at high K; the sum rate of the two algorithms becomes equal somewhere between $K = 30$ and 40. At low SNR, this trend again comes down to scheduling the user with the best channel with the GrA, whereas this is not guaranteed for the GA. In

comparison, at high SNR, orthogonality between users is a more important factor. It is generally better to find users whose channel matrices are closer to orthogonal to each other, rather than first focus on the one user with the best channel gain, since all the users' channel gains are good. Since the GrA is biased towards that one user, who may no longer be the best choice, its performance suffers relative to the GA.

We also observe that the GA tends to perform better at lower K than the GrA. This is since at lower K, orthogonality is also quite important. With a small user pool, finding users that are relatively close to orthogonal is difficult compared to at large K, due to multiuser diversity. Multiuser diversity and basic probability theory dictates that the more users that there are to choose from, the more likely it will be that there are users whose channels are close to orthogonal. More specifically, at lower K, it is therefore more difficult to find users that are orthogonal to the one specific user with the best channel F-norm. In the GrA, the small number of users to select from, combined with the use of the threshold, can in fact often result in the GrA scheduling fewer users than is appropriate. Thus, some of the system's resources and degrees of freedom in the transmitter are wasted. In comparison, while the GrA might exclude certain users because of its threshold and comparisons to the best individual user, the GA does not exclude any particular users. Often, at low K, finding a good set of users that are closer to orthogonal, where that set does not include the user with the best channel, results in a better overall sum rate than if that user must be included. Thus, for these reasons the GA performs better than the GrA when the user pool is small. At large K, a good user set that includes the best user is far more likely, and the GrA is no longer in danger of scheduling too few users, so the GrA performs better at large K.

Further, we note that the performance of the GA does not increase as rapidly with K; an increasing performance gap between the GA and the exhaustive search is observed as K increases. A similar gap was observed in Chapter 3, for scheduling under DPC precoding. Possible ways to compensate for the gap in practice were also discussed in that chapter, including seeding the initial population and preserving chromosomes between scheduling instances if temporal correlation in the channels exists. Nevertheless, this increasing gap results in the crossover in performance between the GrA and the GA with K at mid-range SNRs, since the GrA does not have that same increasing gap; its growth in sum rate with K is the about the same as that for the exhaustive search. Both algorithms have much lower complexity than the exhaustive search, yet still yield decent throughput relative to the best possible, although not as close to the best as for $K_0 = 2$. For

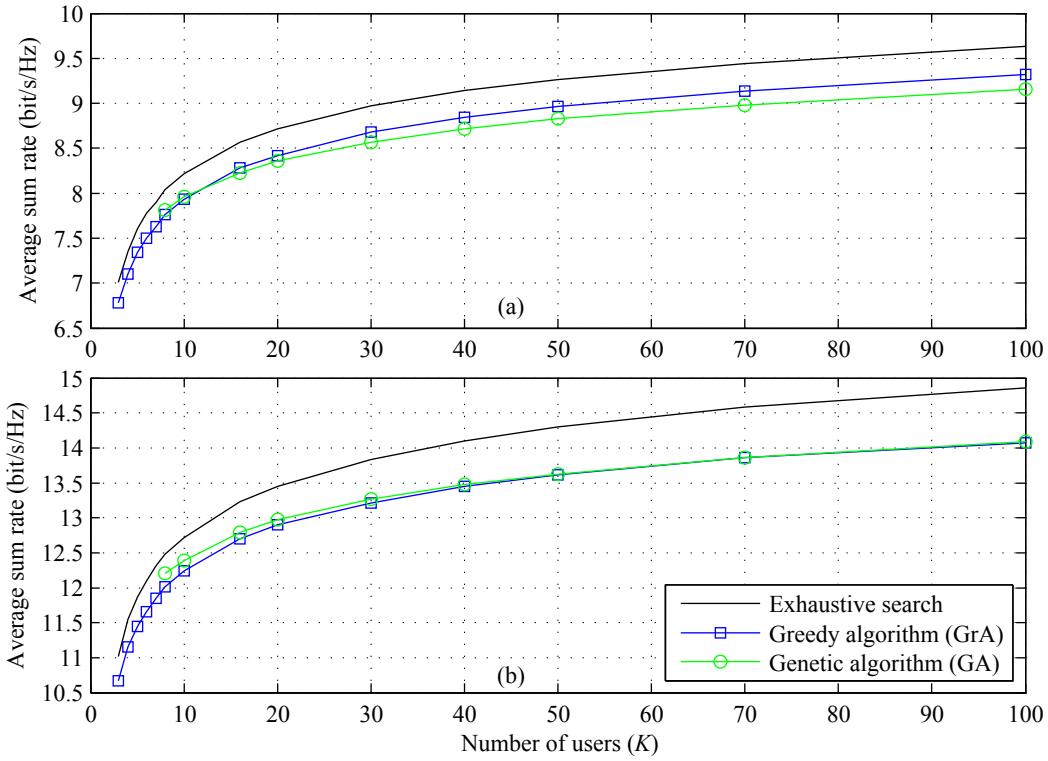

Figure 5.3: Performance vs. K of exhaustive search, greedy, and genetic scheduling algorithms for SZF; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$. (a) SNR = 5 dB, and (b) 10 dB.

5 and 10 dB, the algorithms provide, at worst, no less than about 90% of the optimal sum rate. Finally, we note the two SCAHE algorithms perform better than the GrA and GA at $M_T = 8$, though they both trade off complexity for that additional performance compared to the GrA.

5.5.2 Successive Zero-Forcing

Figure 5.3 shows the SZF performance for $M_T = 4$. We see much the same overall trend as was seen for BD precoding. For small K (e.g. $K < 15$ at SNR = 5 dB) the GA outperforms the GrA, whereas for large K the GrA outperforms the GA. At SNR = 10 dB, the performance of the GrA and GA has been found to be essentially identical for $K > 30$; hence the plots in Figure 5.3(b) are not clearly distinguishable. Additional simulations (results shown in Appendix E) also demonstrated that at 0 dB, the GrA outperformed the GA for all K, while at higher SNRs (i.e., 15 and 20 dB), the GA outperformed the GrA. It is also observed that the proposed algorithms perform very close to the exhaustive search. The performance of both algorithms is less than 0.8 bit/s/Hz inferior to optimal, achieving about 95−98% of the sum rate of an exhaustive search.

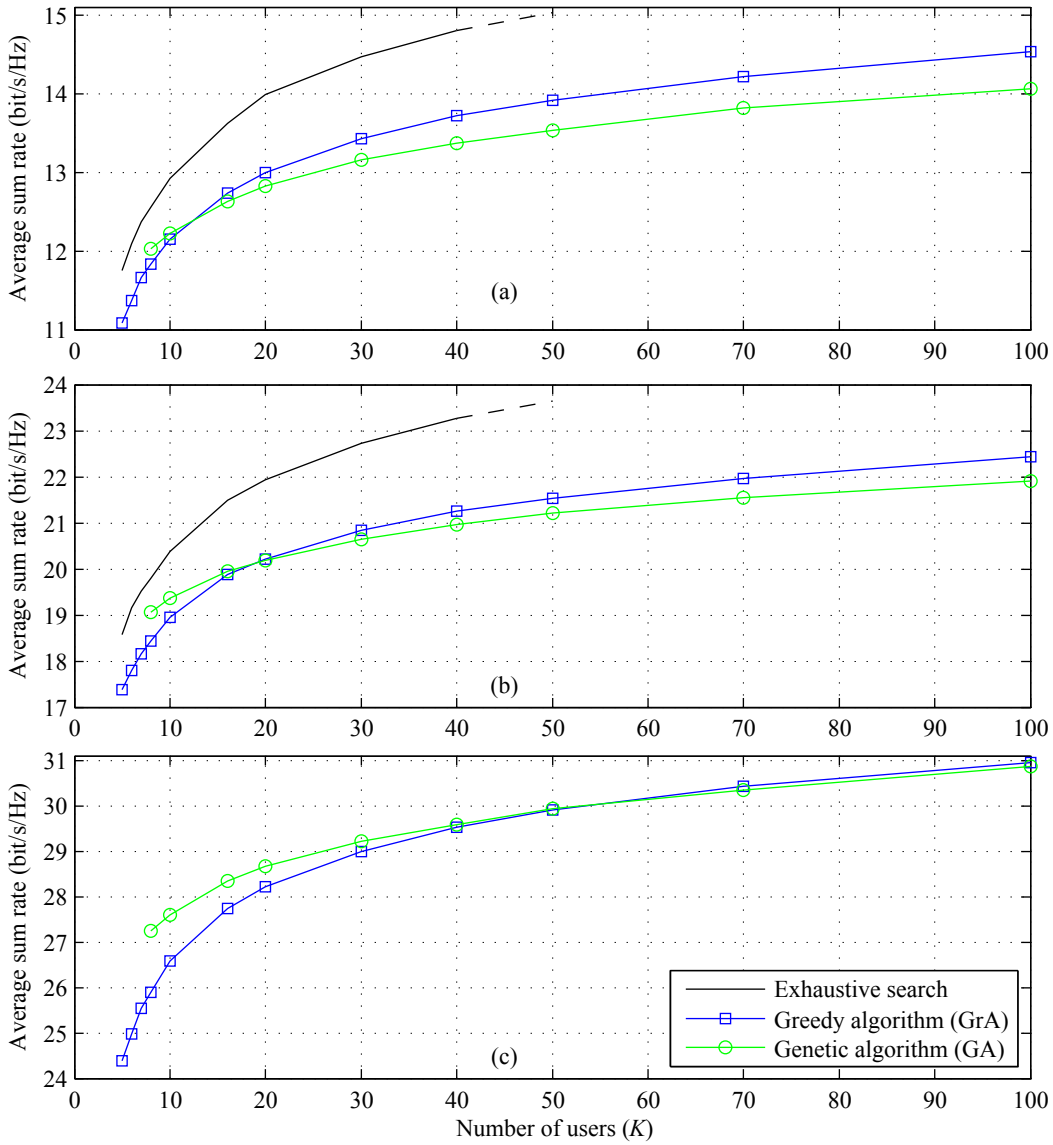

Figure 5.4: Performance vs. K of exhaustive search, greedy, and genetic scheduling algorithms for SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) SNR = 5 dB, (b) 10 dB, and (c) 15 dB.

Similar results are observed for $M_T = 8$ in Figure 5.4. A crossover of the performance curves of the GrA and GA can be seen from these figures, just like what was seen for BD. However, the crossover occurs at a lower value of K compared to BD (e.g. about $K = 20$ for SZF at 10 dB compared to about $K = 35$ for BD). The reasons for this are similar to those for BD. With the use of the threshold, the GrA tends to be biased toward too few users at low K, and toward scheduling the maximum number of users at higher K. The GA has no such bias one way or the other. We note that when scheduling multiple users simultaneously, the cancellation of multiuser interference can become a significant factor. Thus, it is often best to schedule less than the maximum servable number of users in

order to maximize the sum rate for small K, since there is not a large enough pool of users to select from to help reduce multiuser interference through pre-existing user channel orthogonality. Nevertheless, selecting too few users also reduces the throughput, as degrees of freedom in the system are wasted. As was seen with BD, at low K, both scheduling too few users due to the threshold, and being forced to schedule the user with the best channel norm instead of looking more at orthogonal users, causes the GrA performance to suffer relative to the GA. The reverse is true at higher K, since it is more likely for the best user to be in a good orthogonal group due to multiuser diversity. However, finding orthogonal users is not as important as in BD, since SZF only nulls the interference of a user on previously encoded users, not on all other users. Hence, we see the crossover in performance occur at a smaller value of K compared to BD. It would be interesting to examine this crossover in performance between the algorithms further, to find some sort of theoretical analysis as to why the crossover occurs at a specific value of K. However, since the GrA throughput is quite dependent on the threshold ξ, and because the GA is stochastic in nature, this would be extremely difficult, if not impossible.

We examine the number of users scheduled by each of the algorithms in Figure 5.5. The graphs reinforce the statements made in the previous paragraph. At low K ($K = 10$), it is seen that for both BD and SZF, the GrA tends to schedule far fewer users than what an exhaustive search indicates should be appropriate. The GrA tends to schedule just two users a little over 10% of the time in SZF, and even more often in BD. The GrA in fact even schedules only a single user on rare occasions. Thus, available transmit resources are being wasted. In comparison, the GA schedules a (more appropriately) larger number of users at low K, and so its throughput is better. By a more mid-range value of K ($K = 40$), the number of users simultaneously scheduled by both the GrA and the GA are much closer to each other. We also note that in comparing BD to SZF, all the algorithms tend to schedule more users simultaneously when using SZF at all K, as a result of the less stringent null space constraints.

Looking back once more at Figure 5.4, we again see that the rate of increase in throughput of the GA with K does not match that of the exhaustive search, while the GrA does, which also contributes to the crossover. In fact, the sum rate of any beamforming (including BD and SZF) is known to grow as $\log(\log K)$ [41]. In Figure 5.6, we plot the throughput performance of the algorithms vs. $\log(\log K)$ for SZF at 5 and 10 dB and $M_T = 8$. We see that the exhaustive search curve is linear, as expected. More importantly, it is observed that the GrA curve is also linear, with about the same slope as the exhaustive

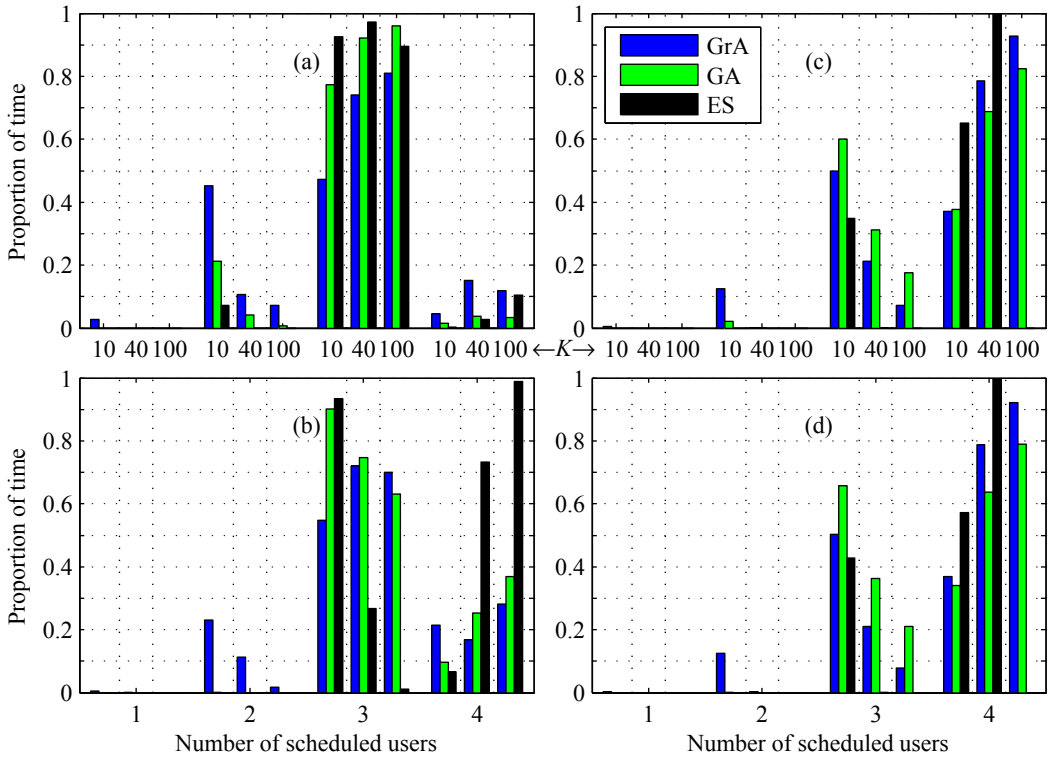

Figure 5.5: Average proportion of time that a given number of users out of a maximum of K_0 is scheduled using various algorithms for $M_T = 8$, $N_k = N = 2$, and $K_0 = 4$, with $K = 10$, 40, and 100. (a) BD, SNR = 5 dB. (b) BD, SNR = 10 dB. (c) SZF, SNR = 5 dB. (d) SZF, SNR = 10 dB.

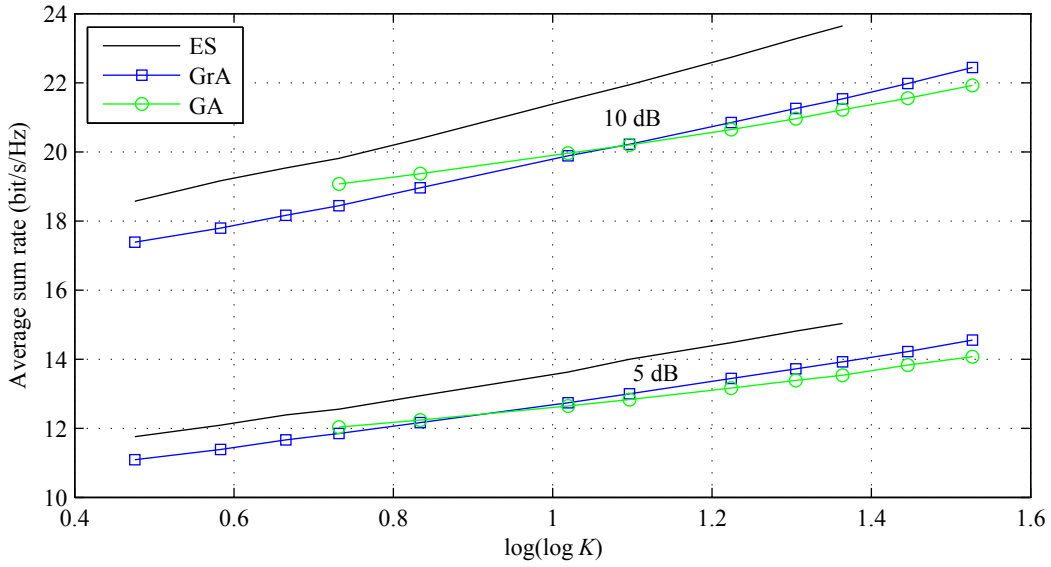

Figure 5.6: Performance of exhaustive search, greedy, and genetic scheduling algorithms for SZF vs. log(log K); $M_T = 8$, $N_k = N = 2$, $K_0 = 4$; SNR = 5 dB and 10 dB.

search, meaning that the GrA achieves about the same growth in throughput versus K due to multiuser diversity as the exhaustive search. In other words, the gap in the GrA performance versus the optimum is approximately constant as K increases. This result

can also be seen in the earlier figures, but not as clearly, especially in Figure 5.4, where the exhaustive search curve does not extend to $K = 100$. It is also observed that the GA curve seems close to being linear, but that its slope is less steep, thus resulting in the widening gap in performance compared to the optimum. However, there still is a slight upward curve to the GA plots. This indicates a possibility that the rate of increase in the gap with the ES performance may slow at even larger K. As K asymptotically becomes very large, the GA might achieve a constant gap from the exhaustive search as well.

The performance of both proposed algorithms is still quite close to that of an exhaustive search, though not quite as close as for $M_T = 4$. The worst performance of the algorithms is still not less than 90% of the optimal throughput, at least up to $K = 40$. Full exhaustive search results are not available for larger K, due to the combinatorially increasing complexity. Simulating for even larger K would take a prohibitively long time.

5.5.3 Hybrid Algorithm 1

In this section, we focus on the $M_T = 8$ cases, as these are the cases where the most improvement in sum rate relative to the exhaustive search can be obtained. Full results are available in Appendix E. We first investigate the various options for seeding the best F-norm user(s) into the chromosomes. We note that the calculations to find the maximum F-norm will have already been carried out, so how often that result is used from that point on will have no further impact on the complexity. We hence examine the effect of seeding the "top" user (with the highest channel F-norm) into one chromosome, two chromosomes, and the entire population of chromosomes for the GA. Furthermore, it is trivial when finding the maximum F-norm to keep track of the top two users instead of just one. Thus, we also examine seeding the top two users each into one chromosome, each into two chromosomes, and seeding into the entire population, where one user is seeded into half of the population's chromosomes, and the second into the other half.

When initializing the GA, the algorithm first creates a random set of users (and a random encoding order for those users, in the case of SZF). The algorithm then checks if the desired seed user has been scheduled in the chromosome. If so, the GA proceeds to the next chromosome. Otherwise, the user drops the last user that was scheduled in the chromosome and inserts the seed user as the first user in the scheduled group. For example, if a chromosome indicates to schedule the users {18,2,12,7}, while user 5 is the seed, the GA would drop user 7 and insert user 5 at the front, creating the group

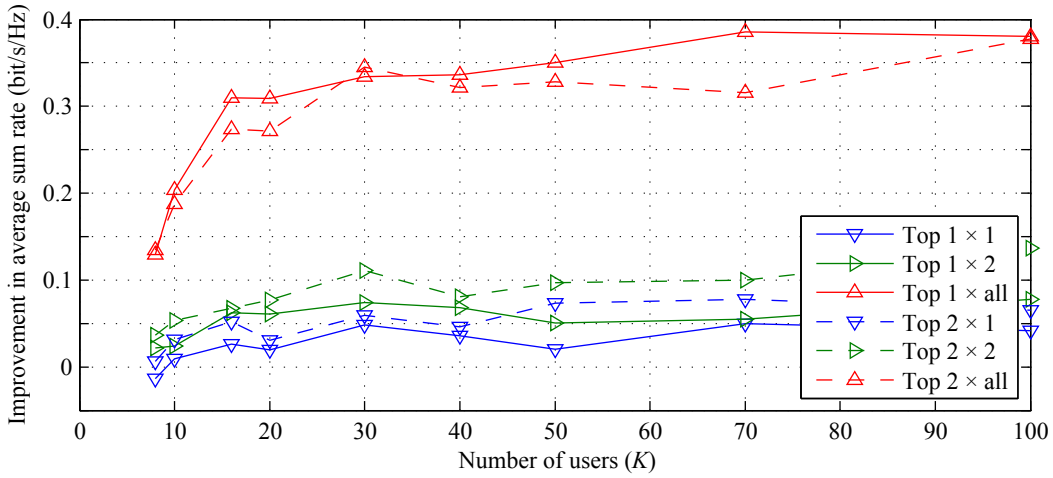

Figure 5.7: Improvement in average sum rate vs. K of hybrid algorithm 1 over the unseeded GA while seeding the top n users into c chromosomes of the GA population. $M_T = 8$, $N_k = N = 2$, $K_0 = 4$; SNR = 10 dB.

{5,18,2,12}. In the case of SZF, this also means the seed user will be encoded first, just as with the SZF greedy algorithm.

We show simulation results in Figure 5.7 for one BD case with $M_T = 8$, $N_k = N = 2$, and $K_0 = 4$ at an SNR of 10 dB. The figure shows the improvement in average sum rate. It can be seen that seeding the top user in just one or two chromosomes has an almost negligible effect on the sum rate. The increase is less than 0.1 bit/s/Hz. However, when the top user is seeded in the entire population, the effect is more significant. The sum rate increases by about 0.3 to 0.4 bit/s/Hz. Seeding the top two users into just one or two chromosomes also has a mostly negligible effect, although seeding two users does seem to improve the performance very slightly. This extra improvement is likely within the error bounds of the simulation, though, since we are dealing with hundredths of a bit/s/Hz. Seeding the top two users throughout the population seems to have no significant difference in effect; if anything, seeding two users may even occasionally worsen the performance. Thus, it appears that the best option is to seed the single top user with the best channel F-norm into the entire GA population.

With this in mind, simulation results for HA1 with the "seed top user in all chromosomes" strategy are seen in Figure 5.8 for BD and SZF. It is observed that for the fairly small increase in the complexity required to seed the GA chromosomes, there is a reasonable, though slight, increase in the performance of HA1 compared to the GA. The gain in throughput is not so large at smaller K; the sum rate increases by about 0.1 bit/s/Hz. This is expected from the discussions in the previous sections; at low K, the GA already performs well, so there is not as much gain to be achieved in the first place. The

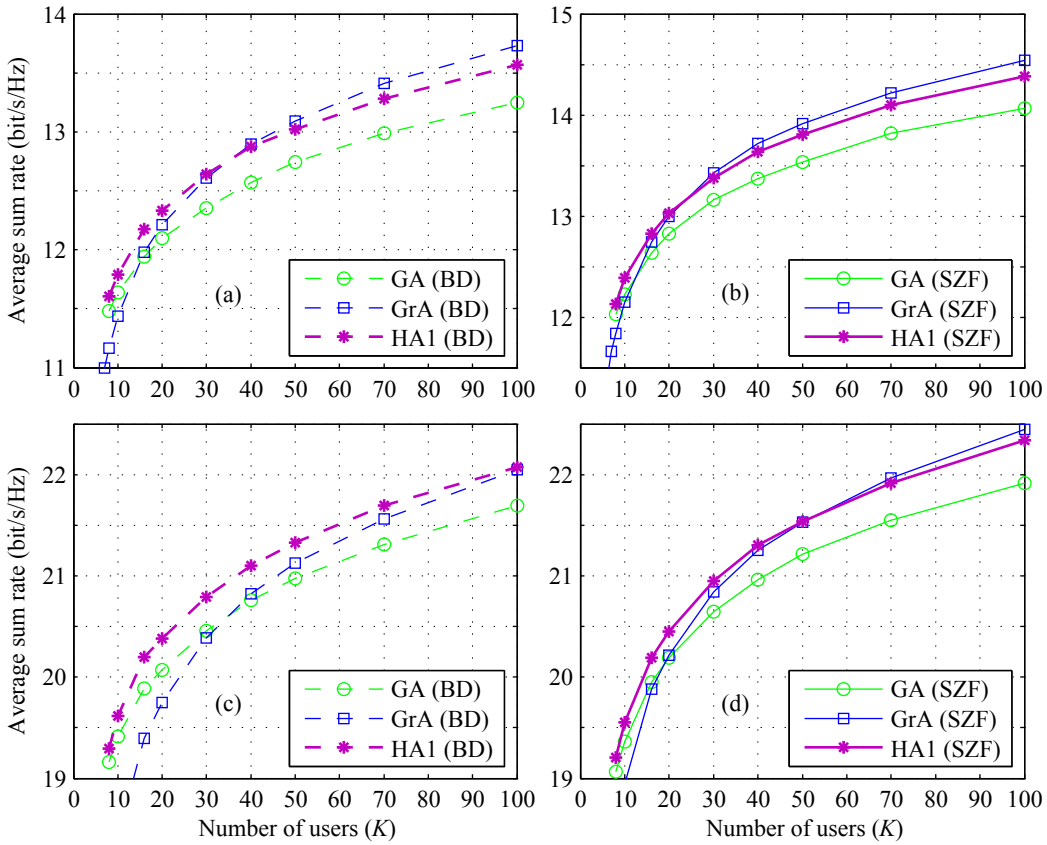

Figure 5.8: Performance of hybrid algorithm 1 vs. K for BD and SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) BD, SNR = 5 dB; (b) SZF, SNR = 5 dB; (c) BD, SNR = 10 dB; and (d) SZF, SNR = 10 dB.

GA in HA1 does see some benefit, though, since it is not forced to keep the user with the best channel; the breeding process could remove that user from contention if it is not included in the best subset of users that will maximize the utility function, but will keep it if it is. HA1 is also not required to keep that top user in the first encoding position for SZF, either. The throughput gain is higher at larger K. The throughput goes up by about 0.3 bit/s/Hz at an SNR of 5 dB in Figure 5.8(a) and (b), respectively, and by about 0.4 bit/s/Hz at 10 dB in Figure 5.8(c) and (d). Essentially the same increase in performance is seen for both BD and SZF. In fact, for both precoding methods, the seeding of the GA improves its performance such that it now approximately equals or even exceeds that of the GrA at 10 dB. This is compared to the crossover in the GA and GrA performance seen without the seeding. However, it should also be noted that the gain in performance amounts to at best about an extra 2% in throughput, so in relative terms, the gain is rather minor.

5.5.4 Hybrid Algorithm 2

In analyzing the performance of HA2, we again focus on the $M_T = 8$ cases, since they best illustrate the effect of adding the GA to the GrA; full results are in Appendix E. First, we examine the best number of generations for which to run the GA. Figure 5.9 shows the case for BD at an SNR of 10 dB, letting the GA run for 2, 5, and 10 generations. It can be seen that the biggest relative gain to the GrA comes after just two generations. The gain is largest at lower K, which is to be expected; the GA was earlier seen to perform better at low K when compared to the GrA. We would expect the GA portion of HA2 to also perform relatively better at higher SNR for similar reasons. Running the GA for 5 and 10 generations does unsurprisingly further improve the average sum rate of HA2. However, there appear to be diminishing returns at this point. For instance, the gain in throughput relative to the GrA at K around 16 to 20 when running 2 generations is the same gain in throughput seen when increasing the generations from 2 to 10. In other words, the same addition in sum rate is seen when the complexity of the GA portion of HA2 quintuples (or when adding 8 more generations) as was seen when the first two generations were added. For this reason, and also because we wish to keep the added complexity from the GA in HA2 as low as reasonably possible, we have decided to continue to use only 2 generations in HA2 for the remainder of the work on this algorithm.

Figure 5.10(a) and (b) show the performance for BD and SZF, respectively, at an SNR of 10 dB. It is observed that HA2 essentially acts asymptotically yielding the better of the GrA and GA performances. At lower K, the HA2 performance is essentially the

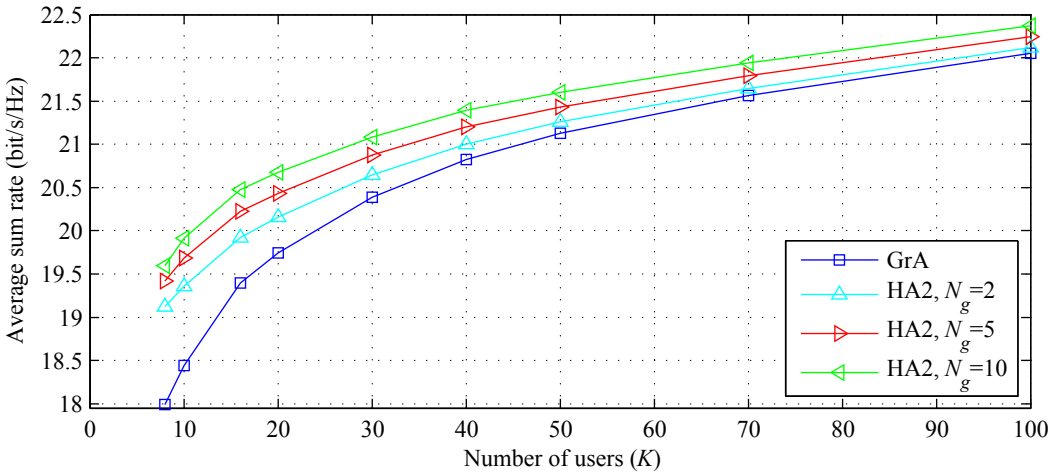

Figure 5.9: Performance of hybrid algorithm 2 vs. K for BD while letting the GA within HA2 run for N_g generations; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$, SNR = 10 dB.

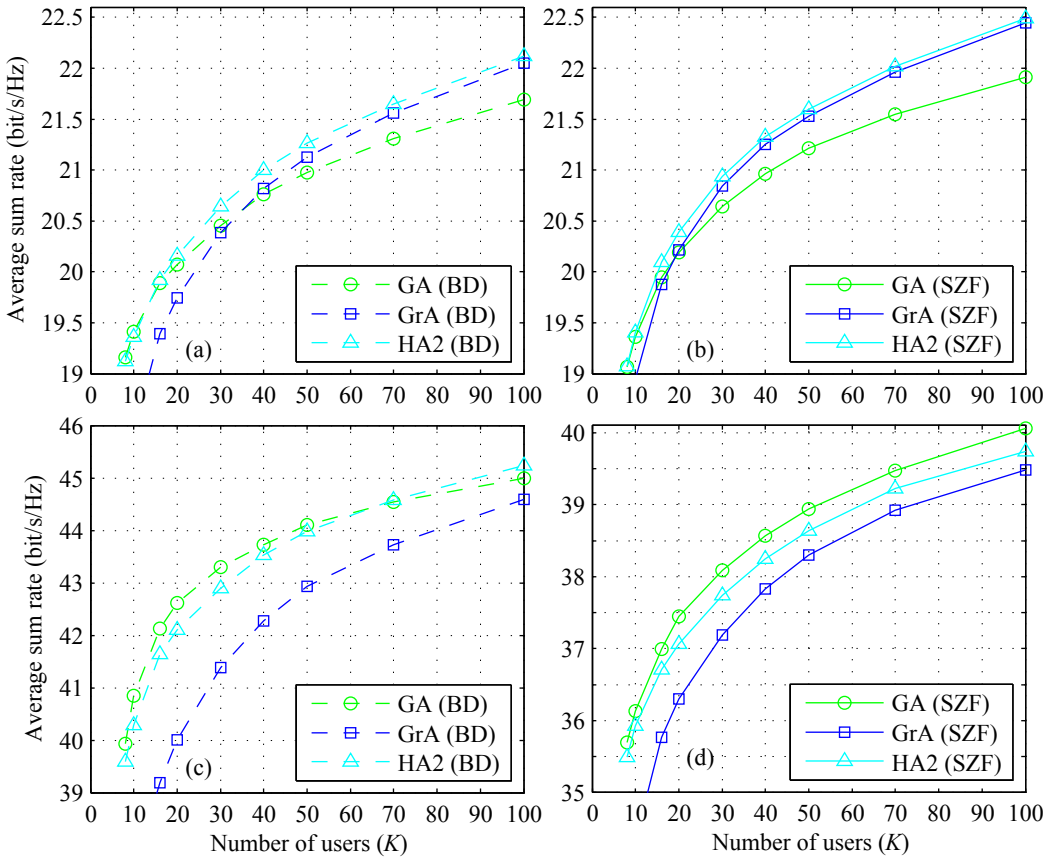

Figure 5.10: Performance of hybrid algorithm 2 vs. K for BD and SZF; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$. (a) BD, SNR = 10 dB; (b) SZF, SNR = 10 dB; (c) BD, SNR = 20 dB; and (d) SZF, SNR = 20 dB.

same as that of the GA. At larger K, its performance is slightly better than that of the GrA. However, most interesting are the mid-range values of K. It is observed that HA2 makes a smooth transition between the GA and GrA performances. In this mid-range (i.e., where the crossover in performance between the GA and GrA occurs), HA2 performs better than either of the GA or the GrA individually. This is without increasing the order of complexity from that of the GrA (though the total number of flops must obviously increase).

We saw the same asymptotic type of performance for HA2 in our simulations for lower values of SNR; these results are shown in Appendix E. However, at high SNR, the trend changes. Figure 5.10(c) and (d) show the HA2 performance at an SNR of 20 dB. It is observed that the HA2 throughput drops below that of the GA. At high SNR, as already discussed, the GrA selection is not necessarily the best, since all users have good channels. Thus, it does not necessarily make a good seed as a starting point for the GA part of HA2, so the net effect is not much better than if the initial population was completely random as in the baseline GA. The throughput then is lower at higher SNRs

simply because HA2 does not iterate as many generations as the GA does. However, we note that the HA2 performance is still improved compared to the GrA. The throughput increases by about 0.64 bit/s/Hz for BD and 0.25 bit/s/Hz for SZF at $K = 100$, and by about 2.9 bit/s/Hz for BD and about 1.5 bit/s/Hz for SZF at $K = 10$. Thus, the gains are the most significant at low K; the sum rate increases by about 8% and 4.5% at $K = 10$ for BD and SZF, respectively. We would expect that for extremely large K the HA2 sum rate would converge asymptotically to about that of the GrA, as was also seen for 10 dB.

Lastly, we note with interest that at 20 dB, the system performance using BD actually exceeds that when using SZF by several bit/s/Hz. This, however, does not have anything to do in particular with the scheduling algorithms. Rather, the lower throughput is largely due to the generation of the SZF covariance matrices. Recall that the method presented in [50] and described in Appendix D to find covariance matrices for SZF is suboptimal. This is in comparison to the waterfilling power allocation for BD, which is known to be optimal. In fact, the covariance matrices obtained using BD must also be a valid solution for SZF, since they meet the SZF constraints. BD requires $\mathbf{H}_{\pi(k)}\mathbf{W}_{\pi(j)} = \mathbf{0}$ for all $j \neq k$, so the relaxed constraint of $\mathbf{H}_{\pi(k)}\mathbf{W}_{\pi(j)} = \mathbf{0}$ for all $j > k$ required for SZF is automatically met. The transmit power constraint on the covariance matrices is also met. Thus, the performance for SZF with optimal covariance matrices should and must be no worse than that when using BD.

For confirmation of the above, we simulated an exhaustive search with both BD and SZF for $M_T = 8$, $K = 16$, and an SNR ranging from 0 to 20 dB. (A full exhaustive search for all K would be too time-consuming to simulate due to the combinatorial complexity.) The results of these simulations are shown in Figure 5.11. These simulations indicate that the best possible SZF average sum rate with these parameters and the suboptimal covariance method at 20 dB is about 40.1 bit/s/Hz. This maximum SZF throughput is still below that for both the suboptimal GA and HA2 with BD, and thus obviously below the optimal throughput using BD. The simulations indicate that the maximum throughput for BD at 20 dB is about 44.5 bit/s/Hz. In fact, the throughput of SZF starts to drop below that of BD starting at about 11 dB SNR, indicating an increasing deficiency in the existing SZF covariance method as the SNR increases. Thus, the lower performance of our scheduling algorithms at 20 dB under SZF, compared to the equivalent algorithms under BD (e.g. the SZF GA or GrA in Figure 5.10(d) versus the BD GA or GrA in Figure 5.10(c)), is unrelated to the scheduling algorithms themselves.

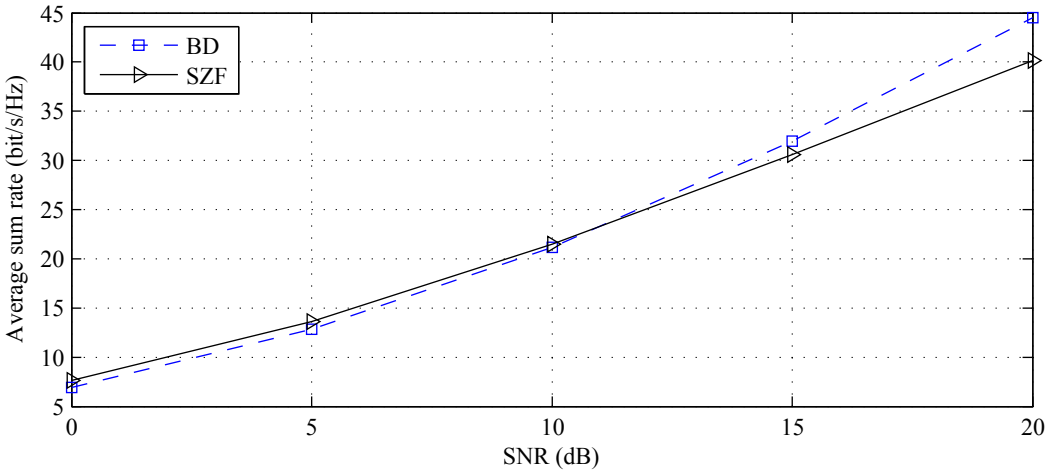

Figure 5.11: Maximum average sum rate vs. SNR for BD and SZF using an exhaustive search; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$, $K = 16$.

5.5.5 Further Discussion

It was seen in the simulation results that the relative performance between the GrA and the GA is dependent on the number of users K and the SNR. The GA performs better than the GrA at higher SNR and lower K; at higher K and lower SNR, the GrA is better. However, in all cases when using BD, the SCAHE capacity-based algorithm performs better than either the GrA or the GA. Additionally, the SCAHE algorithms also have the same order of complexity as the GA.

Furthermore, since the work in this chapter was conducted, we have become aware of another scheduling algorithm proposed in [147]. This algorithm uses a capacity upper bound to schedule users, by assuming the scheduled users can cooperate in decoding their signals. The upper bound is then the single-user MIMO capacity of the aggregate channel matrix $\mathbf{H}$ from Eqn. (5.2). The authors also manage some computational savings by assuming an equal power allocation across the transmit antennas and by the use of the matrix inversion lemma (also known as the Woodbury formula [148]). It is demonstrated in [147] that the proposed algorithm performs about the same as the SCAHE algorithms (most often giving about the same sum rate as the SCAHE F-norm algorithm). However, the order of complexity of the algorithm is only about $\mathcal{O}(KK_0^{-1}M_T^3)$.

In light of the above facts, one may wonder if the use of the GA is justified in practice. If algorithms exist of the same or lesser complexity, but better performance, the use of the GA seems less compelling. However, it should first be recalled that the bulk of the GA complexity lies in the computation of the fitness for each chromosome. Of that,

the most complex part is calculating the null space basis vectors. It should be possible to find some sort of reduced complexity metric, perhaps one that avoids having to calculate the null space vectors every time the GA fitness is calculated. In such a case, the GA complexity would be reduced. Second, BD is a case where encoding order does not apply. When encoding order is also a factor in scheduling, such as in SZF, the problem is more complex, but the GA can handle the extra complexity without much effort. For the specific case of SZF, it was seen that despite the added complexity from the encoding order, the order of complexity of the GA did not increase. We did look briefly at the encoding order for the exhaustive search with SZF, but were unable to see any particular pattern in terms of which users should be encoded where in the order. Third, maximizing throughput is a comparatively simple criterion, so it stands to reason that there exist some lower complexity ways to approximate its performance. However, if several quality of service (QoS) requirements were also added to the mix (e.g. several classes of traffic, delay-sensitive data, etc.), the situation would become much more difficult. The GA is likely more suited to scheduling in such a scenario. Lastly, the GA also used a very naïve initialization method (the random initial population). When some additional information is available to the GA, it can perform better. This was seen to a small degree with HA1, but also with HA2, when the GA used the information from the GrA to improve the performance. In the latter case, the GA was useful as a supplementary search after the initial decision was made. In both cases, again the order of complexity did not increase by using the extra information. In a practical channel with temporal correlation, the GA should be able to use information from past scheduling instances to further improve the performance. [149] describes a related concept, where the authors use an evolutionary algorithm (though not specifically a GA) with an estimation of distribution method [150] for scheduling on a MIMO uplink. The algorithm in general gathers statistical information from the fitness of individuals of previous populations to help guide the initialization of future populations. For all the above reasons, there is still a justifiable reason to use the GA in scheduling applications, either as a full scheduling algorithm itself, or as a supplement to another algorithm.

5.6 Conclusion

In this chapter, we have examined low complexity genetic and greedy user scheduling algorithms for multiuser MIMO downlink systems employing block diagonalization (BD) and successive zero-forcing (SZF) precoding. The proposed

algorithms are much less complex, but perform close to the highly complex exhaustive search. For both BD and SZF, we have demonstrated that at low SNR, the greedy algorithm performs better than the genetic algorithm, but as the SNR increases, the throughput of the genetic algorithm surpasses that of the greedy algorithm. Similarly, at smaller values of K, where K is the number of users requesting service, the GA performs better, but as K increases, the greedy algorithm can outperform the GA. The tradeoff in performance is in large part due to the GrA being biased towards one certain good user, and thus scheduling too few users or not considering certain other users as a result. A detailed complexity analysis showed that for BD, the order of complexity of the GA is higher than that of the GrA by a factor of K_0, where K_0 denotes the maximum number of simultaneously supported multiple-antenna users. For SZF, the GA is more complex than the GrA by a factor of K_0^2. For both BD and SZF, the GrA achieves similar sum-rate growth with K as the exhaustive search, whereas the GA does not for larger M_T.

We have also proposed two hybrid algorithms combining the genetic and greedy algorithms. We demonstrated that the first hybrid, a seeded GA, improved marginally upon the performance of the regular GA at large K by about 0.3 to 0.4 bit/s/Hz, with no increase in the order of complexity. This represents about a 2% increase in throughput. The second hybrid was shown to act asymptotically like the better of the genetic and greedy algorithms at low to mid-range SNRs. At high SNR, the hybrid performance was inferior to that of the GA, but still improved upon the performance of the GrA, while maintaining the same order of complexity as the GrA. At low K, an increase in throughput of about 8% for BD and about 4.5% for SZF was seen relative to the GrA.

The work in this chapter also identified a deficiency in the method used to obtain covariance matrices for SZF. It was seen that even with optimal scheduling, the throughput of SZF can drop below that of BD, when theoretically it should not. There is also a second deficiency with the existing covariance method. The purpose of these examined scheduling algorithms is to potentially improve the performance of fourth generation wireless systems. Such performance measures will also likely include quality of service issues. Unfortunately, the existing SZF covariance method has no way of incorporating those measures. The method is for a straight sum-rate maximization, and thus is not ideal in, for example, a weighted sum rate, wherein the weights may incorporate the QoS measures. We address these issues and propose a solution for both problems in the next chapter.

Chapter 6

Improved Covariance Optimization for Successive Zero-Forcing Weighted and Unweighted Sum-Rate Maximization

6.1 Introduction

In the previous chapter, we examined the average sum rate achievable by using successive zero-forcing (SZF) precoding [50]. In that chapter, we used a suboptimal method from [50] to obtain transmit covariance matrices satisfying the power and null space constraints. That suboptimal method performs reasonably well. The sum rate of SZF exceeds that of block diagonalization (BD) [46] in the simulation results provided in [50] with limited or no scheduling considered. We furthermore found that SZF often exceeds BD when scheduling is also included, based on our own simulation results of the previous chapter. However, we have found two main deficiencies with the existing method. We found in one scenario with 8 transmit antennas, 2 receive antennas per user, and thus up to 4 users capable of being scheduled simultaneously, for any SNR greater than 11 dB, the sum-throughput achieved by SZF dropped below that achieved by BD, even with exhaustive search scheduling. This does not make sense; the BD optimization problem is a more constrained version of the SZF optimization problem. Any solution that satisfies the BD constraints also satisfies the SZF constraints. Therefore, the performance of SZF must in theory always be no worse than that of BD. The problem was ultimately identified to lie with the suboptimal SZF covariance method. The authors in [50] acknowledge that their method is suboptimal, and that better methods can likely be found, but to date, we are unaware of any results in the literature examining exactly how suboptimal the existing method is. We show in the remainder of this chapter that the covariance method in fact becomes worse as both the SNR and the number of supportable users increase.

The second deficiency is that the existing covariance method only accounts for maximization of a pure (unweighted) sum rate. However, the method cannot be directly

applied to a weighted sum-rate (WSR) maximization; i.e., to maximize $\sum_k w_k R_k$, where w_k is a weight for user k, and R_k is the data rate for user k. It may be possible to extend the method of [50] to a WSR by first solving a WSR maximization for the multiple access channel (MAC), then proceeding as normal. However, a WSR maximization for the MAC (and thus for the broadcast channel (BC) due to duality) is found for one specific ordering of users. Namely, it is known that users should be decoded on the MAC in the increasing order of the size of the weights of the users, such that the user with the largest weight is decoded last [82],[151]. Equivalently, the user with the largest weight should be encoded first on the BC. (We have also discussed this issue in Chapter 3 regarding proportionally fair scheduling using DPC.) Because of this, the existing method's transformations and projections may not be the best if a different encoding order is to be considered for the SZF WSR. Neither is it necessarily the case that the same encoding ordering that is optimal to maximize the WSR for SZF is the same ordering required for the MAC / BC.

In this chapter, we propose a new method that accounts for both of these issues. It both significantly improves the throughput for SZF and enables the maximization of a weighted sum rate. Our contributions in this chapter have appeared in [152].

6.2 SZF Covariance Optimization

6.2.1 Problem Discussion

Attempting to solve Eqns. (5.10)–(5.12) from the previous chapter to determine optimal covariance matrices for the SZF sum rate is quite complex. The optimization in (5.11) is non-convex, unlike that of BD, where the complete decoupling of the users' effective channels creates a convex problem. Thus, finding a global optimum can be difficult. In the less-constrained case of DPC, the issue of non-convexity for the broadcast channel can be avoided by operating on the dual MAC instead. This duality would be useful in SZF, since the formula for the SZF sum rate is essentially the same as for DPC, except with the additional null space constraints. Unfortunately, the transformation from MAC to BC or vice versa does not support those additional constraints. Attempting to consider an effective channel such as $\mathbf{H}_{k,e} = \mathbf{H}_k \bar{\mathbf{V}}_{k-1}^0$ on the MAC, similar to that for BD (Eqn. (5.5) in the previous chapter), then transforming to the BC also does not work. The transformations used in [24] have a specific trait that if a set of covariance matrices $\mathbf{P}_k$ with a certain sum-trace (e.g. P) is available for the MAC, the

transformed BC matrices $\mathbf{\Sigma}_k$ will satisfy the same sum-trace constraint. In other words, $\sum_k Tr\left(\mathbf{\Sigma}_k\right) \leq \sum_k Tr\left(\mathbf{P}_k\right)$. It is the *less than* or equal wherein the problem lies for SZF. If $\sum_k Tr\left(\mathbf{P}_k\right) = P$, then $\sum_k Tr\left(\mathbf{\Sigma}_k\right)$ will also equal P only if the matrices $\mathbf{P}_k$ were optimal for the MAC at that sum-trace P. If they were not optimal, then $\sum_k Tr\left(\mathbf{\Sigma}_k\right)$ will be less than P. If one were to then transform these matrices $\mathbf{\Sigma}_k$ with the smaller sum-trace (say, $\tilde{P}$) back to the MAC, one would then obtain a new set of covariance matrices $\tilde{\mathbf{P}}_k$ for the MAC where $\sum_k Tr\left(\tilde{\mathbf{P}}_k\right) = \tilde{P}$, and which provide the same sum rate as the original matrices $\mathbf{P}_k$. This causes the problem for SZF. By forcing the covariance matrices to lie in the null space of other users (as required for SZF) by considering effective channel matrices, the covariance matrices that are obtained will not be optimal for the MAC. Transforming them to the BC in a desire to obtain matrices for SZF will yield a set of matrices with a reduced sum-power. Any set of matrices that does not exactly equal the sum-power constraint cannot be optimal for SZF. This is proven simply by considering the final user encoded in SZF. This user does not cause interference on any users encoded earlier in the encoding order. If the covariance matrices have a total power less than the sum-power constraint, it is then always possible to increase the power allocated to the last user. Doing so will increase its rate, but will have no effect on any other users, leading to a strictly larger new sum rate. This therefore proves the original matrices could not be optimal.

Research has found other capacity duality transformations between the MAC and the BC based on minimax duality [153] and / or Lagrange duality [154]. These transformations allow more general-case linear constraints on the transmit covariance. However, these constraints only apply to the net transmit covariance matrix $\mathbf{\Sigma} = \sum_k \mathbf{\Sigma}_k$, not on the individual covariance matrices $\mathbf{\Sigma}_k$ themselves. These constraints are meant to apply to, for example, power constraints on individual transmit antennas or groups of antennas, instead of (or in addition to) the standard sum-power constraint. There furthermore do exist alternative, more general MAC-BC dualities and transformations, such as mean-squared-error duality [155], SINR duality [53],[156], and rate duality [157], which also account for linear precoding and beamforming[1]. However, the results for these

[1] There are additionally convex transformations for alternative problems other than WSR maximization, such as minimizing power subject to SINR constraints on the users [158]. These, however, are outside the scope of this work.

dualities indicate that even if the null space constraints can be accounted for in the transformations (which is not necessarily guaranteed), the problem on the dual MAC would still be a non-convex problem. Thus, regardless of operating on the MAC or the BC, finding the global optimum would remain difficult. Finding a local optimum solution is somewhat easier, although how far that solution is from the global optimum may be uncertain.

6.2.2 Proposed Conjugate Gradient Projection Method

Since a globally optimal solution is quite difficult to find, a locally optimal method may be useful. Furthermore, since the problem on either the MAC or the BC is non-convex, one may choose to operate on either. Thus, we choose to operate directly on the BC, in order to avoid having to perform MAC-BC transformations.

We propose a conjugate gradient projection (CGP) algorithm to optimize the covariance matrices for SZF. Conjugate gradient methods have provable convergence for convex utility functions, and, when solving systems of linear equations, the convergence is superlinear [159] (that is, if the problem has n real variables, in the worst case the algorithm will take n iterations, but most often much fewer). CGP is also particularly useful in MIMO systems as the solutions can be found using gradients and functions of complex-valued matrix variables. Some methods are only well-defined for functions of real-valued vectors, so in those circumstances the covariance matrices and functions would have to be decoupled and expressed in terms of those vectors. CGP algorithms or gradient projection algorithms have been used for covariance optimization in other similar circumstances. For example, CGP is used in a weighted MAC sum-rate maximization in [108] and [151], and a gradient projection method is used for MIMO interference systems in [160] and the MIMO MAC in [161]. We model our CGP algorithm after the one in [108], which operates on transmit filter matrices $\mathbf{T}_u$ instead of on the covariance matrices $\mathbf{Q}_u$ directly. This method has the advantage of guaranteeing a positive semidefinite covariance matrix $\mathbf{Q}_u = \mathbf{T}_u \mathbf{T}_u^H$ (this is a Cholesky decomposition [43]). Operating on $\mathbf{Q}_u$ directly would require an additional projection during each iteration to ensure the solution is in the set of positive semidefinite matrices (cf. [151]).

Let us rewrite and combine Eqns. (5.10) and (5.11) to account for a weighted sum rate. Without loss of generality, we assume $\pi(k) = k$ for brevity of notation.

$$R_{WSZF} = \max_{\mathbf{B}_k \succeq 0, \sum_k Tr(\mathbf{B}_k) \le P} \sum_{k=1}^{K_0} w_k \log_2 \frac{\left| \mathbf{I} + \mathbf{H}_k \left(\sum_{i=1}^{k} \bar{\mathbf{V}}_{i-1}^0 \mathbf{B}_i \left(\bar{\mathbf{V}}_{i-1}^0 \right)^H \right) \mathbf{H}_k^H \right|}{\left| \mathbf{I} + \mathbf{H}_k \left(\sum_{i=1}^{k-1} \bar{\mathbf{V}}_{i-1}^0 \mathbf{B}_i \left(\bar{\mathbf{V}}_{i-1}^0 \right)^H \right) \mathbf{H}_k^H \right|} \tag{6.1}$$

Note in (6.1) that $Tr(\mathbf{Q}_k) = Tr\left[\bar{\mathbf{V}}_{k-1}^0 \mathbf{B}_k \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \right] = Tr\left[\mathbf{B}_k \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \bar{\mathbf{V}}_{k-1}^0 \right] = Tr(\mathbf{B}_k)$, as the columns of $\bar{\mathbf{V}}_{k-1}^0$ are orthonormal, so $\left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \bar{\mathbf{V}}_{k-1}^0 = \mathbf{I}$ [1]. Thus, there is the same power constraint on $\mathbf{B}_k$ as there is on $\mathbf{Q}_k$.

Let us further define $\mathbf{B}_k = \mathbf{T}_k \mathbf{T}_k^H$, where $\mathbf{T}_k$ is a $\bar{v}_k \times \min\{\bar{v}_k, N_k\}$ matrix, $\bar{v}_k$ is the number of columns in $\bar{\mathbf{V}}_{k-1}^0$, and N_k is the number of receive antennas at user *k*. Thus, the precoding matrix $\mathbf{W}_k$ for SZF will be $\mathbf{W}_k = \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k$. Defining $\mathbf{T}_k$ in such a manner helps reduce the complexity of the optimization by reducing the number of optimization variables [50]. The power constraint can also be re-expressed as $\sum_k \|\mathbf{T}_k\|_F^2 \le P$, since $\sum_k \|\mathbf{T}_k\|_F^2 = Tr(\mathbf{B}_k)$. The CGP algorithm[2] that operates on $\mathbf{T}_k$ is described in Table 6.1.

Because the SZF WSR maximization problem is not convex, the CGP algorithm may not necessarily find the global optimum. Furthermore, the optimal $\mathbf{T}_k$ are not necessarily unique, since $\mathbf{B}_k$ are positive semidefinite. (For example, multiply $\mathbf{T}_k$ by any unitary matrix $\mathbf{U}$, and the new $\tilde{\mathbf{T}}_k = \mathbf{T}_k \mathbf{U}$ will yield the same $\mathbf{B}_k$, since $\tilde{\mathbf{T}}_k \tilde{\mathbf{T}}_k^H = \mathbf{T}_k \mathbf{U} \mathbf{U}^H \mathbf{T}_k^H = \mathbf{T}_k \mathbf{T}_k^H$. Thus, $\tilde{\mathbf{T}}_k$ will provide the same WSR as $\mathbf{T}_k$.)

The local optimum that the algorithm finds is also to some degree dependent on the initial values for $\mathbf{T}_k$. Often, when optimizing covariance matrices, an initial choice of a scaled identity matrix is used, but in general this cannot be done here, as generally $\mathbf{T}_k$ is not a square matrix. Furthermore, even if the algorithm were operating on $\mathbf{B}_k$ instead of $\mathbf{T}_k$, a scaled identity would still not be an appropriate starting point, as the rank would likely be too large; the rank of $\mathbf{B}_k$ would be $\bar{v}_k$ instead of $\min\{\bar{v}_k, N_k\}$. Instead, we

---

[1] However, in general $\bar{\mathbf{V}}_{k-1}^0 \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \neq \mathbf{I}$. $\bar{\mathbf{V}}_{k-1}^0 \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H$ is the projection to the null space used in the covariance method from [50]. If that projection were in fact unitary, the method in [50] would in fact be optimal.

[2] We wish to point out that the "conjugate" in conjugate gradient is unrelated to the concept of the "conjugate" of a complex number, i.e., $\mathbf{x} = \mathbf{a} + i\mathbf{b}$, $\mathbf{x}^* = \mathbf{a} - i\mathbf{b}$. In this context, "conjugate" refers to the property of *conjugacy*. If there exists a set of *n* real-valued, non-zero vectors $\{\mathbf{p}_0, \ldots, \mathbf{p}_n\}$ such that $\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0$ for all $i \neq j$, where $\mathbf{A}$ is a symmetric positive definite matrix, then that set of vectors is said to be conjugate with respect to $\mathbf{A}$ [159].

Table 6.1: CGP algorithm for SZF covariance optimization

Initialize: $\mathbf{T}_k$; $\mathbf{S}_k = \mathbf{0}$, $\forall k$; $\rho = 1$; $\alpha = 1$.
Calculate *WSR* from (6.1).
repeat
Store: $\mathbf{T}_{k\_old} = \mathbf{T}_k$, $\forall k$; $\mathbf{S}_{k\_old} = \mathbf{S}_k$, $\forall k$; $\rho_{old} = \rho$; $WSR_{old} = WSR$.
Calculate gradients: $\mathbf{G}_k, \forall k$ from (6.3).
Normalize gradients: $\bar{\mathbf{G}}_k = \sqrt{\frac{P}{\sum_k \|\mathbf{G}_k\|_F^2}}\mathbf{G}_k$, $\forall k$.
Project gradients: $\hat{\mathbf{G}}_k = \bar{\mathbf{G}}_k - \frac{\sum_k Tr\left(\mathbf{T}_k^H \bar{\mathbf{G}}_k\right)}{\sum_k Tr\left(\mathbf{T}_k^H \mathbf{T}_k\right)}\mathbf{T}_k$, $\forall k$.
Calculate Frobenius norm: $\rho = \sum_k \left\|\hat{\mathbf{G}}_k\right\|_F^2$.
Determine search directions: $\mathbf{S}_k = \hat{\mathbf{G}}_k + \frac{\rho}{\rho_{old}}\mathbf{S}_{k\_old}$, $\forall k$.
Step in search directions: $\hat{\mathbf{T}}_k = \mathbf{T}_{k\_old} + \alpha\mathbf{S}_k$, $\forall k$.
Normalize transmit filter sum-power: $\mathbf{T}_k = \sqrt{\frac{P}{\sum_k \left\|\hat{\mathbf{T}}_k\right\|_F^2}}\hat{\mathbf{T}}_k$, $\forall k$.
Calculate *WSR* from (6.1).
Set *LoopCounter* = 0.
while $WSR < WSR_{old}$ **do**
Decrease step size α.
Set $\mathbf{S}_k = \hat{\mathbf{G}}_k$, $\forall k$.
LoopCounter = *LoopCounter* + 1.
if *LoopCounter* = *LoopThresh* **then**
Set $WSR_{old} = WSR$.
Reset α to 1.
end if
Recalculate $\hat{\mathbf{T}}_k$, $\mathbf{T}_k$, and *WSR*.
end while
until desired accuracy reached

---

initialize $\mathbf{T}_k$ by distributing values of $\sqrt{P/K_0/\bar{v}_k}$ uniformly to the columns of $\mathbf{T}_k$ in a round-robin fashion. This is equivalent to creating a $\min\{\bar{v}_k, N_k\} \times \min\{\bar{v}_k, N_k\}$ identity matrix, vertically concatenating copies of the rows of that identity matrix until there are $\bar{v}_k$ rows, then finally multiplying by $\sqrt{P/K_0/\bar{v}_k}$. For example, if $\mathbf{T}_k$ was 3×2, entries (1,1), (2,2), and (3,1) of $\mathbf{T}_k$ would be initialized to $\sqrt{P/K_0/3}$, while the remaining entries would be 0.

$$\mathbf{T}_{k\_init\_e.g.} = \begin{bmatrix} \sqrt{P/K_0/3} & 0 \\ 0 & \sqrt{P/K_0/3} \\ \sqrt{P/K_0/3} & 0 \end{bmatrix}; \quad \mathbf{B}_{k\_init\_e.g.} = \begin{bmatrix} P/K_0/3 & 0 & P/K_0/3 \\ 0 & P/K_0/3 & 0 \\ P/K_0/3 & 0 & P/K_0/3 \end{bmatrix} \tag{6.2}$$

The gradient can be calculated using matrix calculus from the partial differential of (6.1) with respect to $\mathbf{T}_k^H$ [162]. Specifically, $\nabla_k = 2\partial R_{WSZF} / \partial \mathbf{T}_k^* = 2\left[\partial R_{WSZF} / \partial \mathbf{T}_k^H\right]^T$. Since the gradients will be normalized, leading constants can be left off. We show in Appendix F that the gradient for user k is proportional to:

$$\mathbf{G}_k = \left(\bar{\mathbf{V}}_{k-1}^0\right)^H \left(\sum_{i=k}^{K_0} w_i \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum_{j=1}^{i} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i - \sum_{i=k+1}^{K_0} w_i \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum_{j=1}^{i-1} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i \right) \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k \quad (6.3)$$

The above gradients seem quite complex at first glance. However, some computational savings can be obtained by a successive calculation of part of the gradients. To begin, the sums $\mathbf{\Phi}_i = \sum_{j=1}^{i} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H$ can first be calculated and stored for each $i = 1, \ldots, K_0$ to avoid calculating these sums multiple times. Next, we can define $\mathbf{Z}_k$ as:

$$\mathbf{Z}_k = w_k \mathbf{H}_k^H \left[\mathbf{I} + \mathbf{H}_k \mathbf{\Phi}_k \mathbf{H}_k^H \right]^{-1} \mathbf{H}_k - w_{k+1} \mathbf{H}_{k+1}^H \left[\mathbf{I} + \mathbf{H}_{k+1} \mathbf{\Phi}_k \mathbf{H}_{k+1}^H \right]^{-1} \mathbf{H}_{k+1}. \quad (6.4)$$

Then, each gradient $\mathbf{G}_k$ can be calculated starting from $k = K_0$ downward, using a running sum for $\mathbf{Z}_k$. For example, in the case of $K_0 = 4$, $\mathbf{G}_4 = \left(\bar{\mathbf{V}}_3^0\right)^H \left(\mathbf{Z}_4\right) \bar{\mathbf{V}}_3^0 \mathbf{T}_4$, $\mathbf{G}_3 = \left(\bar{\mathbf{V}}_2^0\right)^H \left(\mathbf{Z}_3 + \mathbf{Z}_4\right) \bar{\mathbf{V}}_2^0 \mathbf{T}_3$, and so on.

In [108], the authors define aggregate matrices $\mathbf{G}$, $\mathbf{S}$, and $\mathbf{T}$, which are the horizontal concatenation of the matrices $\mathbf{G}_u$, $\mathbf{S}_u$, and $\mathbf{T}_u$, respectively. This primarily allows them to avoid the summation of squared F-norms and traces in the notation for their algorithm. For example, in the gradient normalization step, $\sum_u \left\| \mathbf{G}_u \right\|_F^2$ can be represented more compactly as $\left\| \mathbf{G} \right\|_F^2$. This notation, strictly speaking, is not possible with our adaptation for SZF, as the gradients $\mathbf{G}_k$ and matrices $\mathbf{T}_k$ are generally of different dimensions for each k. An equivalent notation could still be used by instead defining aggregate matrices as a block-diagonal formation of the component matrices instead of a horizontal concatenation, i.e., $\mathbf{G} = blkdiag\left(\mathbf{G}_1, \cdots, \mathbf{G}_{K_0}\right)$. However, this could potentially require additional memory and computational complexity unless the algorithm can account for the sparseness of the aggregate matrices (i.e., the many matrix entries after block-diagonalization that equal zero), and is not strictly necessary in the first place.

In the "step in search directions" portion of the algorithm, it is possible to find an approximately best step size for α, for example via an inexact line search like Armijo's Rule [159]. However, we find just as in [108] that it is generally sufficient to simply reduce the step size by a factor if there is no increase in the WSR. For example, we had good results when using equal weights of $w_k = 1$, $\forall k$, by simply multiplying α by about 0.8. We did, however, notice on rare occasions when the algorithm did not converge properly. This is likely due to the non-convexity of the problem; the algorithm is likely stalling near a saddle point in these cases. Repeated decreases in α did not result in an increase in the WSR, and often lead to a small decrease in the WSR. This may also be due to the fact that when a non-linear function is being optimized, an inexact line search (or lack of one, in our case) can lead to the search not being in the correct direction [159]. For example, if a function is being maximized, although the search should be in a direction of ascent, the search direction may actually be in one of descent. Thus, we implement the addition of a loop counter to compensate for these rare cases[1]. If the loop counter reaches a certain threshold (we use a threshold of 100), the previous best WSR is set to the currently found value for the WSR, and α is reset[2] to 1. Since this updated value is often slightly smaller than the previous value, there is a guaranteed larger value that the algorithm can head towards. This slight decrease in WSR and resetting of α is generally enough for the algorithm to get sufficiently far enough away from wherever it has stalled to continue finding a better solution (i.e., even better than where it stalled). If the algorithm's WSR still does not increase notably at this point, then it means the algorithm has found a local solution to the problem, as the change in WSR should be less than the desired accuracy. Thus, the algorithm can stop and return the current solution.

6.3 Simulation Results

In this section, we present simulation results comparing the performance of our proposed SZF CGP covariance optimization method to the existing method. The simulation setup is identical to that in the previous chapters, i.e., a base station with M_T transmit antennas, a pool of K users each with N receive antennas, etc. For reference, we also present the performance when using BD.

[1] During our simulations, these rare cases seemed to primarily occur at low SNR.
[2] This is similar and related in concept to the notion of "restarting" the CGP search during non-linear optimizations, as discussed in [159].

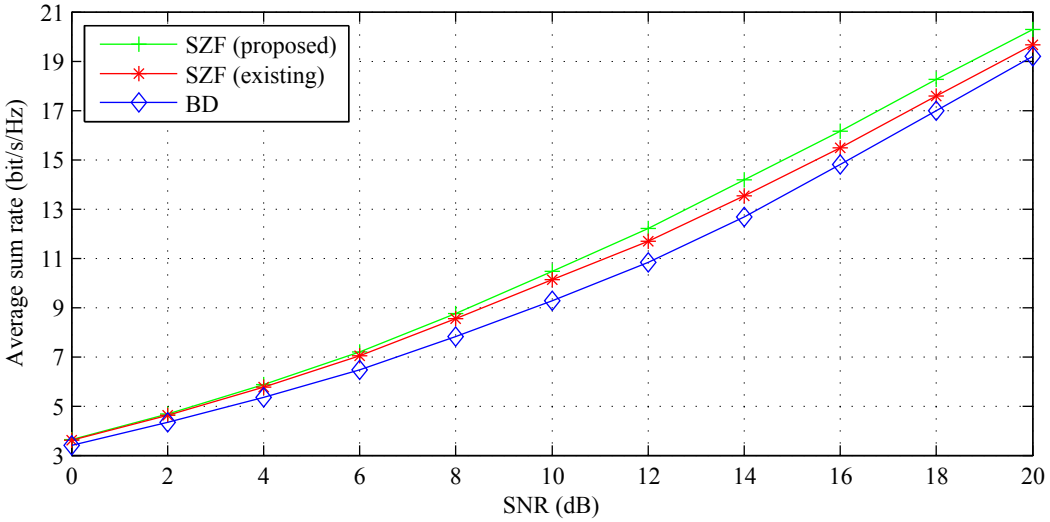

Figure 6.1: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD; $M_T = 4$, $K = K_0 = 2$, $N = 2$.

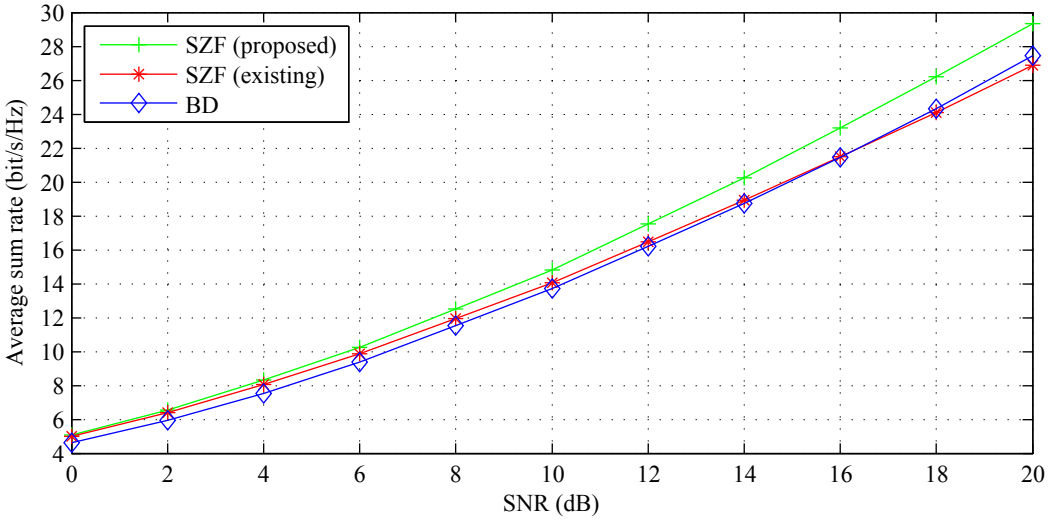

Figure 6.2: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD; $M_T = 6$, $K = K_0 = 3$, $N = 2$.

6.3.1 Unweighted Sum-Rate Performance of Proposed CGP Algorithm

We begin by comparing two relatively simple cases also examined in [50]. Figure 6.1 and Figure 6.2 show cases for an unweighted sum rate (i.e., all users have a weight of 1) for BD and the existing and proposed SZF covariance optimization methods. In the first case, $M_T = 4$, $K = K_0 = 2$, and $N = 2$, while in the second, $M_T = 6$, $K = K_0 = 3$, and $N = 2$. In both of these cases, strictly speaking scheduling is not necessary, as the number of available users K equals the number of simultaneously supportable users K_0. However,

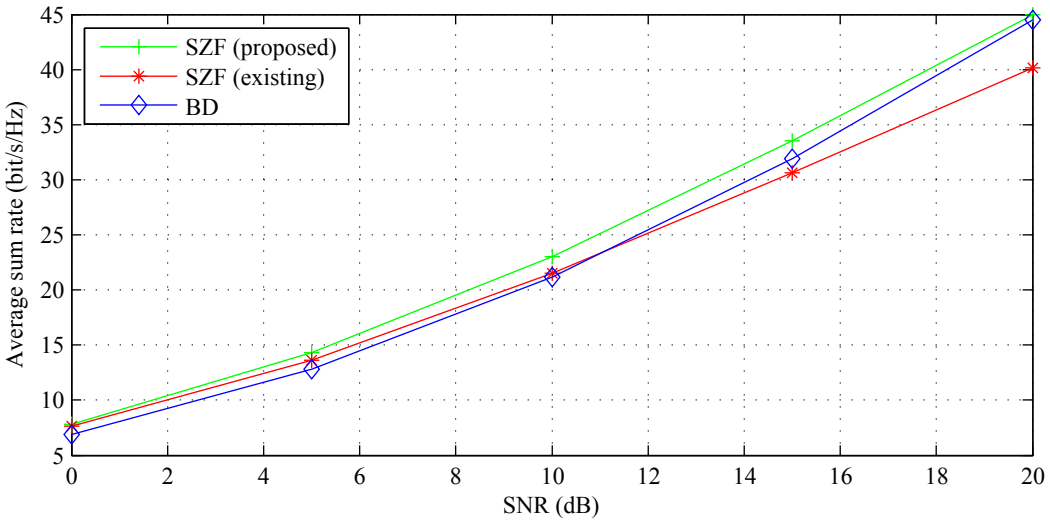

Figure 6.3: Average sum rate vs. SNR with proposed and existing SZF covariance optimization methods and BD, using exhaustive search scheduling; $M_T = 8$, $K_0 = 4$, $N = 2$, $K = 16$.

we do still consider all possible subsets of those users, and all possible encoding orders of those users for SZF, to find the ordered selection that gives the maximum sum rate.

It can be seen that at low SNR, there is essentially no difference between our proposed CGP algorithm and the existing SZF covariance method from [50]. However, as the SNR increases, there is an increasing gain in the throughput of our proposed algorithm relative to the existing algorithm. In Figure 6.1, the gains are rather modest; the sum rate is about 0.35 bit/s/Hz larger at 10 dB, about 0.65 bit/s/Hz larger at 14 dB, and about 0.6 bit/s/Hz larger at 20 dB. These represent percentage gains of about 3.5%, 5%, and 3%, respectively. However, the gains are much more significant in Figure 6.2. The throughput increase is about 0.75 bit/s/Hz at 10 dB, and about 2.45 bit/s/Hz at 20 dB. This is a percentage gain of over 5% and 9%, respectively. More importantly, we note that the performance of the original method is worse than that of BD above an SNR of 16 dB. This result was not visible in [50], as the graph for $M_T = 6$, $K = 3$ in that paper only went up to 16 dB. In comparison, our proposed CGP algorithm performance is consistently above that of BD. We can thus see that the performance gains generally increase both with the number of supported users and with the SNR, though the gain with SNR may eventually start to drop off at lower K_0.

In Figure 6.3, we present a somewhat more complicated scenario more related to our scheduling work in the previous chapter. In this case, we consider a larger user pool size of $K = 16$ with $M_T = 8$. Each user in the pool has $N = 2$ receive antennas, so at most $K_0 = 4$ users can be served simultaneously. We use an exhaustive search for scheduling that

considers all possible subsets of users and encoding orders. This decouples the effect of the specific scheduling algorithm and allows us to focus on the performance of the SZF covariance optimization methods. This is also the scenario from the end of the last chapter where we identified the deficiencies with the existing method.

For the existing numerical method to find the covariance, as also seen in the last chapter, the average sum rate for SZF quickly becomes less than that of BD, at around 11 dB. However, the average sum rate for SZF using our CGP algorithm remains higher than that of BD at least up to an SNR of 20 dB. The improvement in performance over the existing algorithm is about 0.68 bit/s/Hz (about 5%) at 5 dB, about 1.5 bit/s/Hz (about 7%) at 10 dB, and about 4.85 bit/s/Hz (about 12%) at 20 dB.

We note, though, that the gain in throughput relative to BD starts to decrease at higher SNR. The throughput likely becomes less than that of BD at an SNR somewhere larger than 20 dB. This serves to demonstrate that our CGP algorithm, though improved, is still globally suboptimal. However, a worse performance than BD can be avoided with our algorithm. Rather than the "round-robin" initialization described in Section 6.2.2, instead the matrices $\mathbf{T}_k$ can be initialized based on the BD-optimal covariance matrices. Initial values for $\mathbf{B}_k$ can be obtained from the BD matrices $\mathbf{Q}_{k,BD}$ by $\mathbf{B}_k = \left(\bar{\mathbf{V}}_{k-1}^0\right)^H \mathbf{Q}_{k,BD} \bar{\mathbf{V}}_{k-1}^0$. Since the BD matrices meet the SZF null space constraints, they can be built from the SZF null space basis vectors, meaning $\mathbf{Q}_{k,BD}$ can be expressed as $\bar{\mathbf{V}}_{k-1}^0 \mathbf{B}_{k,BD} \left(\bar{\mathbf{V}}_{k-1}^0\right)^H$. Thus, $\mathbf{B}_k = \left(\bar{\mathbf{V}}_{k-1}^0\right)^H \bar{\mathbf{V}}_{k-1}^0 \mathbf{B}_{k,BD} \left(\bar{\mathbf{V}}_{k-1}^0\right)^H \bar{\mathbf{V}}_{k-1}^0 = \mathbf{I}\mathbf{B}_{k,BD}\mathbf{I} = \mathbf{B}_{k,BD}$. Initial values for $\mathbf{T}_k$ can then be obtained through Cholesky decomposition. However, this for the most part should be unnecessary, as that extremely high of an SNR is unlikely to be seen in practice.

We have also noticed that, while the existing SZF covariance method is worse than our proposed CGP method, the covariance matrices $\mathbf{Q}_{k,o}$ provided by that method sometimes provide a better starting point for our CGP algorithm than the round-robin initialization. This is particularly the case at high SNR. For example, consider the scenario from Figure 6.2. For that same scenario, Figure 6.4 shows a distribution of the gain in sum rate at 20 dB obtained using $\mathbf{Q}_{k,o}$ as the starting point, relative to the sum rate using our original initialization. The large spike near zero shows that for a large proportion of the time, initializing with $\mathbf{Q}_{k,o}$ has no effect on the resulting sum rate. About 4% of the time, $\mathbf{Q}_{k,o}$ actually yields a smaller sum rate. However, around 35% of the time, initializing with $\mathbf{Q}_{k,o}$ results in a higher sum rate, with the throughput in certain instances

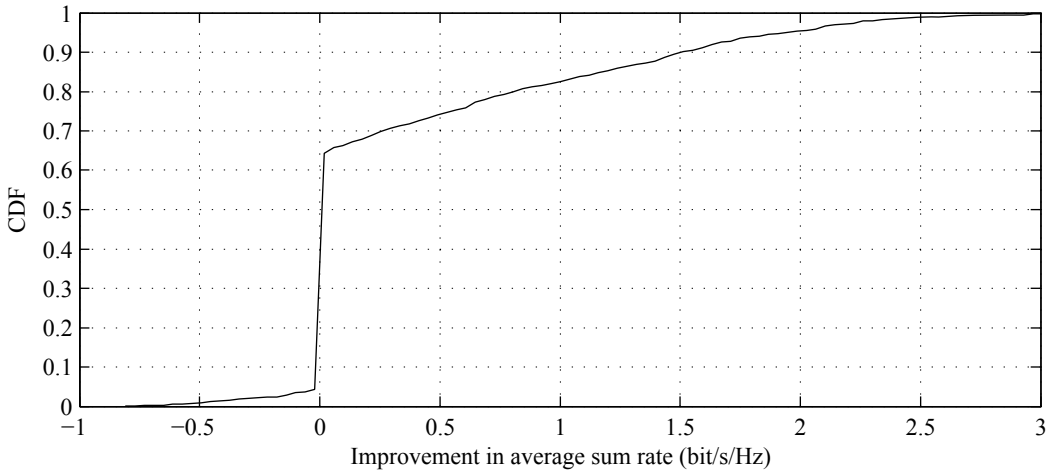

Figure 6.4: Distribution of improvement in average sum rate at 20 dB when initializing CGP algorithm with $\mathbf{Q}_{k,o}$; $M_T = 6$, $K = K_0 = 3$, $N = 2$.

increasing by up to 3 bit/s/Hz. However, averaged over all instances, the mean increase in sum rate is not that significant. When using $\mathbf{Q}_{k,o}$, the average sum rate for our CGP at 20 dB increases from about 29.3 bit/s/Hz to about 29.7 bit/s/Hz. This extra throughput represents on average about an additional 1.4% increase in throughput. Thus, on average, this small additional throughput is likely not worth the added computational complexity. To get that extra percent, in effect two optimizations must be run. The first is on the MAC (followed by transforms and projections) to find $\mathbf{Q}_{k,o}$, then a second with our CGP algorithm using $\mathbf{Q}_{k,o}$ for initialization.

Furthermore, as K increases, this effect seems to essentially disappear. If we consider now the scenario from Figure 6.3, there is virtually no difference in the average sum rate between the two initialization methods. Our simulations only showed an improvement of about 0.03 bit/s/Hz at 20 dB, which is certainly negligible and likely within the error margin of the simulation. It appears that the larger user pool and scheduling has the effect of removing any initialization-based gains. In part, this is because the larger user pool means that the scheduled users' channels are closer to orthogonal. The larger pool also means that the scheduling algorithm has more options to choose a different set of users or encoding order that may negate any effect from the different initialization point.

6.3.2 Weighted Sum-Rate Performance of Proposed CGP Algorithm

We now consider a simple scenario for a weighted sum rate. We examine the case where $M_T = 8$, $K = K_0 = 4$, and $N = 2$. Recall from Chapter 3 that the proportionally fair scheduling algorithm takes a fair amount of effort to simulate, due to the need to build up

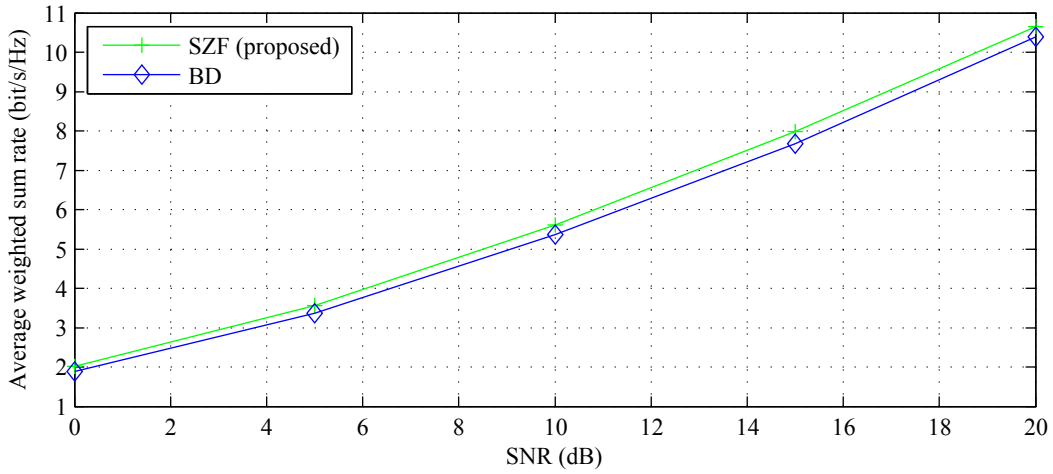

Figure 6.5: Average weighted sum rate vs. SNR with proposed SZF covariance optimization method and with BD; $M_T = 8$, $K = K_0 = 4$, $N = 2$, $w_k = k/10$.

average rate statistics for the users. To avoid this, in this scenario we simply set the weight for each user proportional to that user's index, i.e., $w_k = k / \sum_k w_k$. (The sum in the denominator is just for normalization and does not affect the rates the scheduled users receive.) Such a scenario might arise in practice if each user belongs to a different class of service, such as if they are carrying different types of traffic, or they have paid for higher average data rates. Figure 6.5 shows the WSR performance of our proposed CGP method relative to a WSR using BD. All possible user subsets and orderings are considered. Recall that there exists no prior method for weighted SZF covariance optimization, so we cannot compare our performance to any such algorithm. We observe that the WSR of SZF is larger than that when using BD. The SZF algorithm performs better than BD in this scenario by about 0.5 dB in SNR.

Our simulations for this scenario also indicated only a minor correlation between the best user encoding order and the relative sizes of the weights. Figure 6.6 shows a histogram of how often each user index is encoded in a given position for the best obtained WSR. A user index of 0 indicates that no user was encoded in that position (i.e., transmitting to less than the maximum supportable number of users maximized the WSR). It can be seen that there is somewhat of a tendency to encode the users in the decreasing order of their weights. This trend is strongest at lower SNRs. However, as the SNR increases, this trend diminishes. At 20 dB, for example, it is approximately equally likely that users 3 and 4 (with weights 0.3 and 0.4, respectively) will be encoded first. User 2 is encoded first about half as often as 3 or 4, but also encoded second about half as often as 3 or 4. Thus, there is no hard rule to determine the optimal encoding order for a WSR for

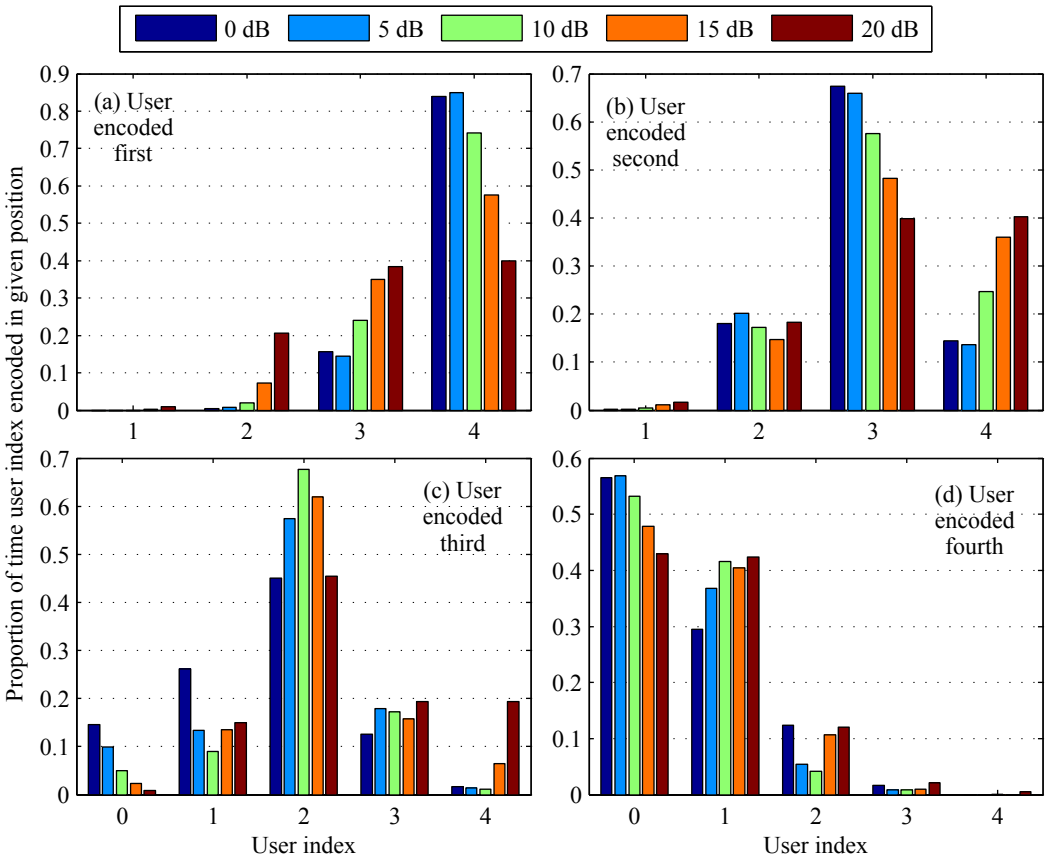

Figure 6.6: Proportion of time that user index is encoded in given position to maximize SZF weighted sum rate, where user weights equal 1/10 of user indices ($w_k = k/10$); $M_T = 8$, $K = K_0 = 4$, $N = 2$. Index "0" indicates no user encoded in that position. (a) User index is encoded first. (b) User index encoded second. (c) User index encoded third. (d) User index encoded fourth.

SZF. This is in stark contrast to the MAC or when using DPC on the BC, as discussed in the introduction to this chapter.

We also note that even at high SNR, it is often best in terms of maximizing the WSR to not transmit to the maximum possible number of users. In this scenario, with the limited user pool to choose from, it is better to transmit to less than the maximum about 43–57% of the time. We made a similar observation in the previous chapter for unweighted sum rates when scheduling to small user pools, for both BD and SZF. The likelihood of scheduling the maximum possible number of users increased in the previous chapter with the size of the user pool K, due to multiuser diversity and the increased chance of finding users with orthogonal channels. This fact is unlikely to change by using our proposed CGP optimization algorithm here.

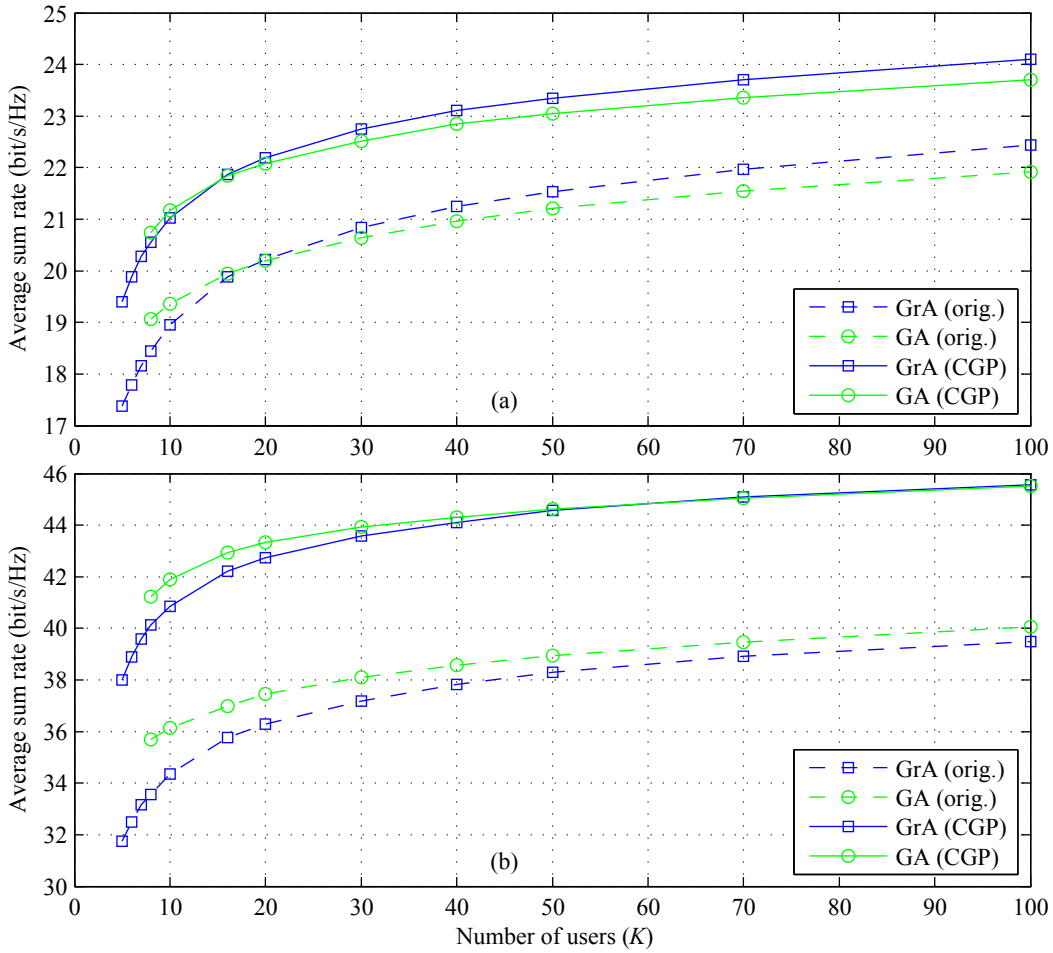

Figure 6.7: Average sum rate vs. K comparing original and proposed CGP covariance optimization methods; $M_T = 8$, $N = 2$, $K_0 = 4$. (a) SNR = 10 dB, and (b) 20 dB.

6.3.3 Updated Scheduling Performance of GA and GrA

Now that we have an SZF covariance algorithm that performs better than the existing algorithm, we proceed to run the genetic and greedy scheduling algorithms from the previous chapter to find the improvement in their performance. We concentrate primarily on the $M_T = 8$ case. Based on the results from Figure 6.1, there is not much change to be expected with the CGP method versus the original method with $M_T = 4$. Simulations verify this; the change between the two methods is generally less than 0.1 bit/s/Hz in sum rate, which is less than a 1% change in throughput.

Figure 6.7 shows a comparison of the performance vs. K of the GA and GrA when using the original method and our proposed CGP covariance optimization method with $M_T = 8$ at an SNR of 10 and 20 dB; more results are in Appendix G. There is a significant increase in throughput seen for our proposed CGP algorithm. At 10 dB, the sum rate of

the GA increases by around 1.7–1.9 bit/s/Hz, or about 8–9.5%, while the GrA increases by about 1.7–2.1 bit/s/Hz, or about 7.5–11.5%. At 20 dB, the GA sum rate increases by about 5.5–6 bit/s/Hz, or about 13.5–16%, while the GrA increases by about 6–6.6 bit/s/Hz, or about 15.5–19.5%.

It is clear that the GrA seems to benefit the most from using our proposed CGP algorithm. This can be seen in the larger increases in throughput compared to those for the GA. It can also be seen when comparing the GA and GrA performance at 20 dB. With the original method, the GA outperforms the GrA, whereas with our CGP algorithm, the throughput of the two algorithms is nearly identical. This is explained in part by the threshold in the GrA. We have seen in our simulations that the optimal threshold for the GrA is somewhat larger when using the CGP method compared to the original method. This means that the GrA rejects fewer users from consideration, leading to an increase in throughput, particularly at low K. However, this must also mean that the GrA is more computationally complex, since it must now calculate metrics for those users when scheduling.

6.3.4 Original Covariance Method for User Selection with CGP for Sum-Rate Maximization

As we have seen from the simulation results earlier in this chapter, the original covariance method from [50] is significantly inferior to our proposed CGP method in terms of the average sum rate it provides at higher SNR and larger K_0. However, the original method does have the advantage of being faster. Recall from the complexity analysis of the GA for SZF in the last chapter that the order of complexity of finding the SZF sum rate (i.e., the fitness of one chromosome) is $\mathcal{O}(M_T^3 K_0)$. The step of finding the null space basis vectors is also $\mathcal{O}(M_T^3 K_0)$ in itself. Since our proposed CGP algorithm also requires the same null space basis vectors, it too must be at least of complexity order $\mathcal{O}(M_T^3 K_0)$. However, the MAC waterfilling algorithm only requires around 5 iterations to converge, as discussed in the previous chapter. (We ignore the additional complexity of the transformations and projections for the moment.) In comparison, our CGP algorithm requires a significantly longer period of time to converge.

Figure 6.8 shows distributions of the number of iterations required for the CGP algorithm to converge such that the change in SZF sum rate is less than 10^{-3} bit/s/Hz and less than 5×10^{-8} bit/s/Hz. We show the case for $M_T = 8$, $N = 2$, $K_0 = 4$, using a random selection and ordering of K_0 users at an SNR of 10 and 20 dB. The CGP algorithm

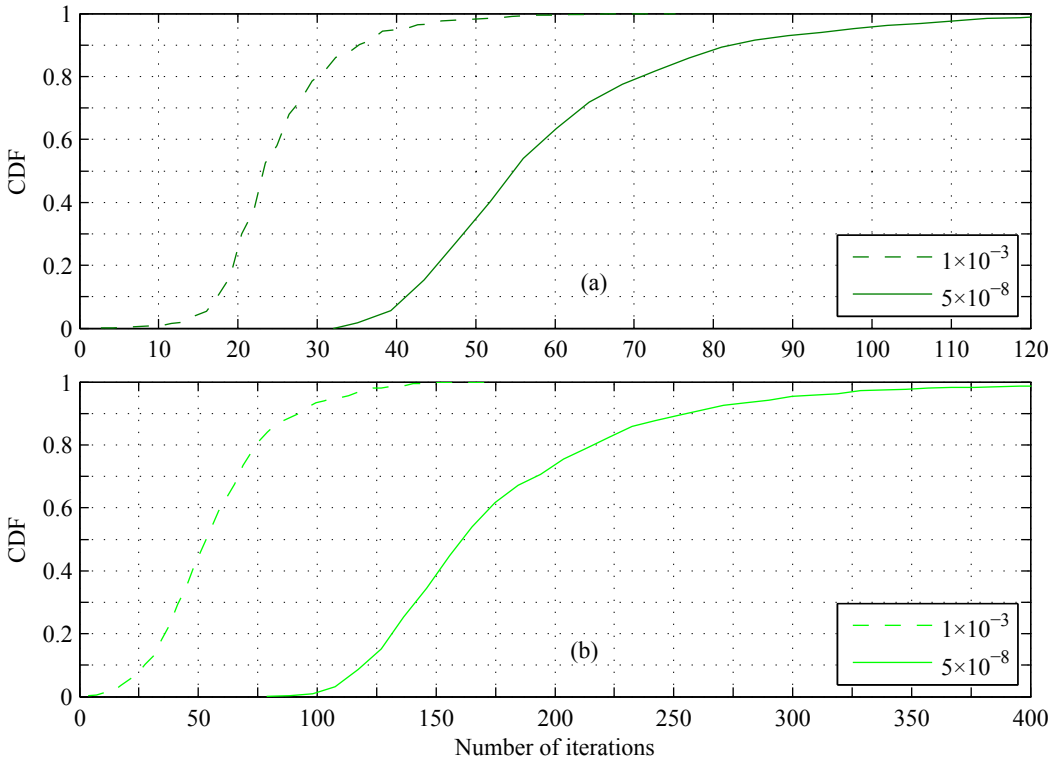

Figure 6.8: Distributions of convergence of CGP algorithm such that SZF sum rate changes by less than 1×10^{-3} / 5×10^{-8} bit/s/Hz; $M_T = 8$, $N = 2$, $K_0 = 4$. (a) SNR = 10 dB, and (b) 20 dB.

appears to have a significant dependence of its convergence on SNR; the higher the SNR, the slower the convergence. At 10 dB, it takes an average of 26 iterations to converge to 10^{-3} bit/s/Hz and 62 iterations to converge to 5×10^{-8} bit/s/Hz. At 20 dB, this increases to 59 and 183 generations, respectively.

This slower convergence is motivation to use the faster original covariance method for user scheduling. Thus, we investigate the following: the fitness for the GA chromosomes is given by the SZF sum rate as determined by the original covariance method. (We shall refer to this as "DL fitness" in the following, based on the initials of the authors' names in [50].) Once a group of users and encoding order is selected based on the DL fitness, our proposed CGP method is used to find better covariance matrices for that scheduling decision. Note that this alternative only applies to the GA, since the SZF covariance optimization is not involved in the GrA scheduling itself, but rather only after the scheduling decision has been made, and to a lesser extent in the selection of the threshold.

Figure 6.9 shows the performance of the GA when using the DL fitness for scheduling. As part of that scheduling, the original method's covariance matrices are

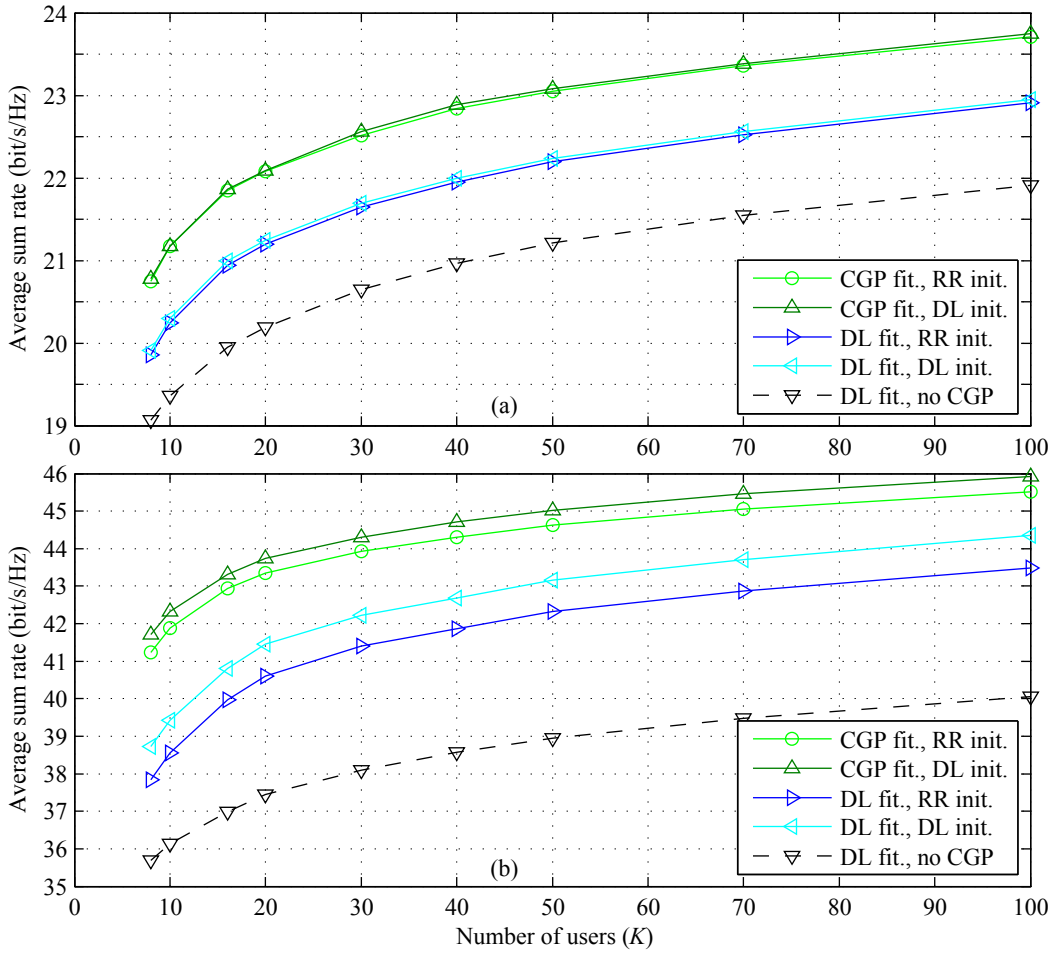

Figure 6.9: Average GA sum rate vs. K comparing CGP and DL fitness for scheduling, followed by CGP sum rate maximization with RR and DL initialization; $M_T = 8$, $N = 2$, $K_0 = 4$. Performance without CGP algorithm also shown. (a) SNR = 10 dB, and (b) 20 dB.

automatically available as part of calculating the DL fitness. Thus, it is trivial to initialize the CGP with those matrices instead of the "round-robin" initialization we have proposed, in order to obtain the small increase in performance they provide. We also compare the sum rate using these two initialization points, dubbed "DL init" and "RR init", respectively. We also include the sum rate from the previous chapter (without using the CGP algorithm) for reference.

It can be seen that the DL fitness provides a worse sum rate than using the CGP algorithm for the GA fitness. At 10 dB, the DL fitness is about 0.8–0.9 bit/s/Hz worse, providing about 96% of the throughput of the CGP fitness. At 20 dB, the loss is about 2.0–3.4 bit/s/Hz when using the RR initialization, or about 4.5–8%. Interestingly, at this SNR, and when using the DL fitness, this is an instance where using DL initialization actually has a notable effect on the sum rate. The loss relative to the CGP fitness with RR

initialization is only 1.1–2.5 bit/s/Hz (2.5–6%), a difference of 0.9 bit/s/Hz compared to the loss using the DL fitness with RR initialization. Hence, there is an overall loss in performance compared to the CGP fitness. However, performing the CGP sum-rate maximization after the selection still provides a better sum rate than not doing so. The improvement over just using the original method by itself is about 0.8–1.0 bit/s/Hz, or 4–5%, at 10 dB, and 2.1–3.4 bit/s/Hz, or 6–8.5%, at 20 dB when using the RR initialization. With the DL initialization, the gain increases to 3.0–4.3 bit/s/Hz (8.5–10.5%) at 20 dB. Thus, there is some justification to using the DL fitness for scheduling, if its reduced complexity is an important factor in the system design. However, the usage is somewhat limited; it is only beneficial for medium SNRs. At high SNR, the throughput when using BD can be larger than using SZF with the DL fitness for scheduling; see also the BD results in Appendix G.

6.4 Conclusion

We have proposed and examined an improved method based on conjugate gradient projection for optimizing the covariance matrices for SZF precoding. This proposed method outperforms the existing method from [50] by up to an additional 12% in sum rate for the simpler cases analyzed with exhaustive search scheduling and smaller numbers of active users K. It was also seen that there is an increasing gain in the performance of our method over the prior method both with increasing SNR and with higher numbers of simultaneously supportable users K_0. Our proposed method also consistently had a throughput larger than that when using block diagonalization (BD); the throughput of the existing scheme was seen to drop below that of BD at higher SNR and K_0.

Our CGP method also supports the maximization of a weighted sum rate (WSR) using SZF. Such a weighted sum rate is important for quality of service issues. To our knowledge, there is no prior method for WSR maximization using SZF in the literature. We demonstrated with a simple case that even when considering a WSR, our proposed method still provided a higher weighted throughput than when using BD.

Even larger gains were obtained when using the CGP method in conjunction with the genetic and greedy algorithms examined in the previous chapter. The GA sum rate increased by about 8–9.5% at 10 dB and about 13.5–16% at 20 dB, while the GrA improved by about 7.5–11.5% at 10 dB and about 15.5–19.5% at 20 dB. It was further seen that using the prior covariance method for scheduling, then optimizing the

covariance matrices with the CGP algorithm for that user selection, provides about half the gain in throughput as using the CGP sum rate for scheduling.

Although our proposed method improves on the performance of the existing method, our method is still not globally optimal. Since the SZF optimization problem is non-convex, finding the global optimum is very difficult. It is thus hard to say how far away our scheme is from the global optimum for SZF. There are a few global optimization techniques which could find the best overall solution. Since the performance of our CGP algorithm is to some degree dependent on the starting point used to initialize the algorithm, a stochastic method could be used to try different starting points. One could run a large-scale simulation trying multiple random starting points for the CGP algorithm. Even better, a genetic algorithm could be used in order to drive those random starting points towards a location that would lead to the global optimum. Given enough time, such a method would eventually cause the CGP algorithm to converge to near the global optimum. Alternatively, a branch-and-bound with reformulation linearization technique such as that described in [163],[164],[165] may assist in finding the global optimum. However, such a technique would be extremely complex and not meant for real-time implementation in practical systems. It would be meant solely to provide an optimal benchmark for comparative purposes. Nonetheless, the global optimization problem remains as future work.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

With research for future wireless communication systems concentrating on MIMO designs, there is a need for MIMO scheduling algorithms. Such scheduling algorithms must choose users to balance numerous factors, such as increasing the system throughput, lowering multiuser interference, ensuring fairness, meeting quality of service constraints, and so forth. In this thesis, we investigated low complexity scheduling algorithms for multiuser MIMO systems.

In particular, we examined the use of genetic algorithms (GAs) for MIMO scheduling. In Chapter 3, we investigated the use of GAs for scheduling in systems employing dirty paper coding (DPC). DPC is optimal in achieving the broadcast channel capacity of multiuser MIMO systems, so this case provided a benchmark for determining the best that the GA can perform. The GA chromosome structure we proposed can both account for an encoding order in the precoding and allow the scheduling of users on several carriers in a multi-carrier system. The maximum throughput and proportional fairness (PF) scheduling criteria were examined, although the general structure used for the GA allows for the maximization of an arbitrary utility function. It was seen that the GA performed within about 0.5 dB of an exhaustive search for the cases examined. A further increase in sum-throughput and a decrease in delays were seen when a 4-carrier OFDM system was examined. The GA performance was obtained at a large runtime decrease compared to an exhaustive search. The GA complexity was linear in both the number of active users and in the maximum number of users that can be scheduled, compared to the combinatorial complexity of the exhaustive search.

In Chapter 4, it was seen that tuning the parameters of the GA is very important in reducing the time it takes for the GA to converge. We found that by tuning the parameters in the adaptive mutation rate, the convergence time dropped to less than 30% of the time for untuned values in one case. We were also able to derive a simple equation

that was linear in the parameters to adjust their values for changing numbers of active users and numbers of supportable users, to allow the values to be in the range required for near-minimum convergence times. We also examined the effect of replacing the one-point crossover operator of the GA with a uniform crossover operator, but found that this had little effect on the GA convergence.

The GA was then extended to systems with linear beamforming in Chapter 5. We examined the performance of a GA and a "greedy" algorithm (GrA) for each of block diagonalization (BD) and successive zero-forcing (SZF). An in-depth complexity analysis found that the GrA was of lower complexity than the GA by a factor of K_0 for BD and K_0^2 for SZF, where K_0 is the number of simultaneously supportable users. It was found that the GA tended to perform better than the GrA at higher signal-to-noise ratios (SNRs) and lower numbers of active users K, while the GrA was better at lower SNRs and higher K. Both algorithms performed fairly closely to an exhaustive search, providing no worse than 90% of the optimal throughput. We also proposed two hybrid algorithms combining the traits of the GA and the GrA. The first hybrid improved upon the performance of the GA by about 2% in throughput, while the second improved upon the GrA, increasing throughput at low K by about 8% for BD and 4.5% for SZF. Both hybrid algorithms yielded these improvements without causing an increase in the order of complexity of the algorithms.

Our work in Chapter 5 identified two deficiencies with the existing method in the literature for SZF covariance optimization. The existing method provides lower throughput than it should at higher SNRs, and does not support the maximization of a weighted sum rate. In Chapter 6, we proposed a new method based on conjugate gradient projection that addresses both deficiencies. The proposed method improved the throughput of an exhaustive search by about 7% at an SNR of 10 dB, and by about 12% at 20 dB. Using this new method with the GA and GrA from Chapter 5 resulted in even larger gains. The throughput of the GA improved by about 8–9.5% at 10 dB and about 13.5–16% at 20 dB, while the GrA improved by about 7.5–11.5% at 10 dB and 15.5–19.5% at 20 dB.

In summary, this work demonstrated that genetic algorithms can provide a viable method of scheduling for multiuser MIMO systems. However, in some circumstances, a GA may not be the best overall choice. The results of Chapter 5 demonstrated that it is possible to design greedy scheduling algorithms with the same or lower complexity than the GA, and that provide higher throughput than the GA. That being said, the

circumstances of those results were somewhat simplistic. The algorithms were examined assuming perfect channel knowledge, uncorrelated channels, and a comparatively simple maximization of the system sum rate. GAs may be better suited for circumstances where there are several other factors of fairness and quality of service that must also be balanced. The GA is also only as complex as the function used to calculate the fitness of the chromosomes. By using a lower complexity fitness function, the GA could perhaps trade off that lower complexity for a longer run time to further increase its performance. Lastly, the GA was seen to be a good supplement to existing algorithms. By running a greedy algorithm, followed by a GA for a small number of generations, the performance is guaranteed to be no worse than the greedy algorithm alone, and for a comparatively small increase in complexity. That is, the increase in complexity was less than the order of complexity overall of the original algorithm. Thus, there is certainly a place for genetic algorithms in the realm of MIMO multiuser scheduling.

7.2 Future Work

The work performed in this thesis used relatively simple scheduling utility functions. That is, we limited our work to the maximum throughput and proportional fairness criteria. However, the PF criterion should be representative of how well the GA should perform relative to an exhaustive search for any weighted sum rate. Nevertheless, it would be useful to investigate the performance of a genetic scheduling algorithm with actual QoS constraints, and various classes of data traffic being carried. Some possible alternative utility functions are described in Chapter 2 (e.g. [87]–[93]). In general, the only change necessary to the GA would be to use these utility functions to calculate the fitness of the chromosomes. All other aspects of the GA would remain the same.

Another possible area for research is in transmitter-receiver coordination. We mentioned in a few places in this thesis that coordinated transmitter-receiver processing is capable of supporting additional users. It would be of interest to examine this further in how it relates to GA scheduling. On a related note, it is also possible for the users to perform receive antenna switching, i.e., to activate and deactivate their antennas as the situation warrants. The results of [73], for example, seem to indicate this may in fact be required to truly maximize the sum rate for BD. A GA could help in this regard; in such a situation, the GA would become a joint scheduling and receive antenna selection algorithm.

The research herein assumed perfect channel knowledge. However, such knowledge is unfeasible in a practical system. Thus, an important area for research is the effect of imperfect or limited channel state information on the scheduling process. DPC is particularly known for being sensitive to errors in the channel knowledge. [86] and [166] are examples of work in this area, but the field in general and for multiuser MIMO specifically has yet to be well covered. Interestingly, there is a possibility that a GA could benefit from limited channel information, in that the fitness function would likely be less complex. But in any case, the effect of limited feedback on GA scheduling would be a good subject for future work. Such work should also consider more practical channel models, including such effects as temporal and / or spatial correlation.

An emerging topic in wireless research is in the area of network coordination. In these techniques, the base stations for several cells act together to help reduce the interference within the network. In one example, this may include coordinating the transmissions of base stations so signals sent to users in one cell do not interfere with the reception of users in other cells from their base station [167]. More recently, though, coordinated multipoint (CoMP) transmission has been of interest, where the base stations in several cells can act as a distributed MIMO array. A user in one cell can receive useful information from several base stations simultaneously [168]. A genetic scheduling algorithm may be quite useful for CoMP. A GA has a structure well suited to parallelization in the calculation of the fitness of the chromosomes. These calculations could be distributed among the base stations in order to help speed up the process and reduce the load compared to a single location running a scheduling algorithm. Thus, the use of GAs for scheduling with CoMP is a promising direction for research.

Lastly, as we noted in Chapter 6, the improved covariance optimization algorithm we proposed is still globally suboptimal. It would be worthwhile to determine what the globally optimal performance of SZF is. It is unlikely that this could be determined on a timescale reasonable enough for practical implementation, but it would still serve as a useful benchmark.

Bibliography

[1] "Requirements related to technical performance for IMT-Advanced radio interface(s)," Int. Telecommun. Union, Tech. Rep. ITU-R M.2134, Nov. 2008.

[2] "Guidelines for evaluation of radio interface technologies for IMT-Advanced," Int. Telecommun. Union, Tech. Rep. ITU-R M.2135-1, Dec. 2009.

[3] Y. Xiao, "IEEE 802.11n: enhancements for higher throughput in wireless LANs," *IEEE Wireless Commun*., vol. 12, no. 6, pp. 82–91, Dec. 2005.

[4] *High Speed Downlink Packet Access* (*HSDPA*); *Overall Description; Stage 2* (*Release 7*), 3rd Generation Partnership Project (3GPP) Standard TS 25.308 V7.11.0 (2010-09), Sept. 2010.

[5] *Evolved Universal Terrestrial Radio Access* (*E-UTRA*) *and Evolved Universal Terrestrial Radio Access Network* (*E-UTRAN*); *Overall Description; Stage 2* (*Release 8*), 3rd Generation Partnership Project (3GPP) Standard TS 36.300 V8.12.0 (2010-03), Mar. 2010.

[6] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, "LTE-advanced: next-generation wireless broadband technology," *IEEE Wireless Commun*., vol. 17, no. 3, pp. 10–22, Jun. 2010.

[7] K. Etemad, "Overview of mobile WiMAX technology and evolution," *IEEE Commun. Mag*., vol. 46, no. 10, pp. 31–40, Oct. 2008.

[8] S. Ahmadi, "An overview of next-generation mobile WiMAX technology," *IEEE Commun. Mag*., vol. 47, no. 6, pp. 84–98, Jun. 2009.

[9] Q. Li, G. Li, W. Lee, M.-il Lee, D. Mazzarese, B. Clerckx, and Z. Li, "MIMO techniques in WiMAX and LTE: a feature overview," *IEEE Commun. Mag*., vol. 48, no. 5, pp. 86–92, May 2010.

[10] İ. E. Telatar, "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecommun*., vol. 10, no. 6, pp. 585–595, Nov./Dec. 1999.

[11] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Pers. Commun*., vol. 6, no. 3, pp. 311–335, Mar. 1998.

[12] L. Zheng and D. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[13] *Orthogonal Frequency Division Multiplexing for Wireless Communications*, Y. Li and G. L. Stüber, Eds. New York, NY: Springer-Verlag, 2006.

[14] S. Weinstein and P. Ebert, "Data transmission by frequency-division multiplexing using the discrete Fourier transform," *IEEE Trans. Commun. Technol*., vol. 19, no. 5, pp. 628–634, Oct. 1971.

[15] M. H. M. Costa, "Writing on dirty paper," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 3, pp. 439–441, May 1983.

[16] G. Caire and S. Shamai, "On achievable rates in a multi-antenna broadcast downlink," in *Proc. 38th Annu. Allerton Conf. Commun., Control and Comput.*, Monticello, IL, Oct. 2000.

[17] G. Caire and S. Shamai (Shitz), "On the achievable throughput of a multiantenna Gaussian broadcast channel," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1691–1706, Jul. 2003.

[18] R. Knopp and P. A. Humblet, "Information capacity and power control in single-cell multiuser communications," in *Proc. IEEE Int. Conf. Commun*. (*ICC'95*), Seattle, WA, Jun. 1995, vol. 1, pp. 331–335.

[19] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness, and stability," *J. Oper. Res. Soc*., vol. 49, no. 3, pp. 237–252, Mar. 1998.

[20] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, NY: McGraw-Hill, 2008.

[21] S. Verdú, "Multiple-access channels with memory with and without frame synchronism," *IEEE Trans. Inf. Theory*, vol. 35, no. 3, pp. 605–619, May 1989.

[22] W. Yu, W. Rhee, S. Boyd, and J. M. Cioffi, "Iterative water-filling for Gaussian vector multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 1, pp. 145–152, Jan. 2004.

[23] A. Cohen and A. Lapidoth, "The Gaussian watermarking game," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1639–1667, Jun. 2002.

[24] S. Vishwanath, N. Jindal, and A. Goldsmith, "Duality, achievable rates, and sum-rate capacity of Gaussian MIMO broadcast channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2658–2668, Oct. 2003.

[25] N. Jindal, S. Vishwanath, and A. J. Goldsmith, "On the duality of Gaussian multiple-access and broadcast channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 768–783, May 2004.

[26] H. Weingarten, Y. Steinberg, and S. Shamai (Shitz), "The capacity region of the Gaussian multiple-input multiple-output broadcast channel," *IEEE Trans. Inf. Theory*, vol. 52, no. 9, pp. 3936–3964, Sept. 2006.

[27] M. Tomlinson, "New automatic equaliser employing modulo arithmetic," *Electron. Lett*., vol. 7, no. 5, pp. 138–139, Mar. 1971.

[28] H. Harashima and H. Miyakawa, "Matched-transmission technique for channels with intersymbol interference," *IEEE Trans. Commun*., vol. 20, no. 4, pp. 774–780, Aug. 1972.

[29] R. F. H. Fischer, *Precoding and Signal Shaping for Digital Transmission*, New York, NY: John Wiley & Sons, 2002.

[30] C. Windpassinger, R. F. H. Fischer, T. Vencel, and J. B. Huber, "Precoding in multiantenna and multiuser communications," *IEEE Trans. Wireless Commun*., vol. 3, no. 4, pp. 1305–1316, Jul. 2004.

[31] V. Stankovic and M. Haardt, "Successive optimization Tomlinson-Harashima precoding (SO THP) for multi-user MIMO systems," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process.* (*ICASSP'05*), Philadelphia, PA, Mar. 2005, vol. 3, pp. III/1117–III/1120.

[32] W. Yu, D. P. Varodayan, and J. M. Cioffi, "Trellis and convolutional precoding for transmitter-based interference presubtraction," *IEEE Trans. Commun*., vol. 53, no. 7, pp. 1220–1230, Jul. 2005.

[33] A. Bennatan, D. Burshtein, G. Caire, and S. Shamai (Shitz), "Superposition coding for side-information channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 5, pp. 1872–1889, May 2006.

[34] S.-C. Lin and H.-J. Su, "Practical vector dirty paper coding for MIMO Gaussian broadcast channels," *IEEE J. Sel. Areas Commun*., vol. 25, no. 7, pp. 1345–1357, Sept. 2007.

[35] C. B. Peel, B. M. Hochwald, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication – part I: channel inversion and regularization," *IEEE Trans. Commun*., vol. 53, no. 1, pp. 195–202, Jan. 2005.

[36] B. M. Hochwald, C. B. Peel, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication – part II: perturbation," *IEEE Trans. Commun*., vol. 53, no. 31, pp. 537–544, Mar. 2005.

[37] U. Erez, S. Shamai (Shitz), and R. Zamir, "Capacity and lattice strategies for cancelling known interference," in *Proc. Int. Symp. Inf. Theory and Its Applications* (*ISITA'00*), Honolulu, HI, Nov. 2000, pp. 681–684.

[38] C. Windpassinger, R. F. H. Fischer, and J. B. Huber, "Lattice-reduction-aided broadcast precoding," *IEEE Trans. Commun*., vol. 52, no. 12, pp. 2057–2060, Dec. 2004.

[39] U. Erez and S. ten Brink, "A close-to-capacity dirty paper coding scheme," *IEEE Trans. Inf. Theory*, vol. 51, no. 10, pp. 3417–3432, Oct. 2005.

[40] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," *IEEE J. Sel. Areas Commun*., vol. 24, no. 3, pp. 528–541, Mar. 2006.

[41] M. Sharif and B. Hassibi, "A comparison of time-sharing, DPC, and beamforming for MIMO broadcast channels with many users," *IEEE Trans. Commun*., vol. 55, no. 1, pp. 11–15, Jan. 2007.

[42] A. Wiesel, Y. C. Eldar, and S. Shamai (Shitz), "Zero-forcing precoding and generalized inverses," *IEEE Trans. Signal Process*., vol. 56, no. 9, pp. 4409–4418, Sept. 2008.

[43] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York, NY: Cambridge Univ. Press, 1985.

[44] M. Joham, W. Utschick, and J. A. Nossek, "Linear transmit processing in MIMO communications systems," *IEEE Trans. Signal Process*., vol. 53, no. 8, pp. 2700–2712, Aug. 2005.

[45] Q. H. Spencer, C. B. Peel, A. L. Swindlehurst, and M. Haardt, “An introduction to the multi-user MIMO downlink,” *IEEE Commun. Mag*., vol. 42, no. 10, pp. 60–67, Oct. 2004.

[46] Q. H. Spencer, A. L. Swindlehurst, and M. Haardt, “Zero forcing methods for downlink spatial multiplexing in multiuser MIMO channels,” *IEEE Trans. Signal Process*., vol. 52, no. 2, pp. 461–471, Feb. 2004.

[47] L.-U Choi and R. D. Murch, “A transmit preprocessing technique for multiuser MIMO systems using a decomposition approach,” *IEEE Trans. Wireless Commun*., vol. 3, no. 1, pp. 20–24, Jan. 2004.

[48] Z. Pan, K.-K. Wong, and T.-S. Ng, “Generalized multiuser orthogonal space-division multiplexing,” *IEEE Trans. Wireless Commun*., vol. 3, no. 6, pp. 1969–1973, Nov. 2004.

[49] Z. Tu and R. Blum, “Multiuser diversity for a dirty paper approach,” *IEEE Commun. Lett*., vol. 7, no. 8, pp. 370–372, Aug. 2003.

[50] A. D. Dabbagh and D. J. Love, “Precoding for multiple antenna Gaussian broadcast channels with successive zero-forcing,” *IEEE Trans. Signal Process*., vol. 55, no. 7, pp. 3837–3850, Jul. 2007.

[51] P. Tejera, W. Utschick, G. Bauch, and J. A. Nossek, “Subchannel allocation in multiuser multiple-input-multiple-output systems," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4721–4733, Oct. 2006.

[52] M. Stojnic, H. Vikalo, and B. Hassibi, “Rate maximization in multi-antenna broadcast channels with linear preprocessing,” *IEEE Trans. Wireless Commun*., vol. 5, no. 9, pp. 2338–2342, Sept. 2006.

[53] M. Codreanu, A. Tölli, M. Juntti, and M. Latva-aho, “Joint design of Tx-Rx beamformers in MIMO downlink channel,” *IEEE Trans. Signal Process.*, vol. 55, no. 9, pp. 4639–4655, Sept. 2007.

[54] A. Wiesel, Y. C. Eldar, and S. Shamai (Shitz), “Linear precoding via conic optimization for fixed MIMO receivers,” *IEEE Trans. Signal Process*., vol. 54, no. 1, pp. 161–176, Jan. 2006.

[55] M. Schubert and H. Boche, “Iterative multiuser uplink and downlink beamforming under SINR constraints,” *IEEE Trans. Signal Process*., vol. 53, no. 7, pp. 2324–2334, Jul. 2005.

[56] H. Boche and M. Schubert, “Optimal multi-user interference balancing using transmit beamforming,” *Wireless Pers. Commun*., vol. 26, no. 4, pp. 305–324, Sept. 2003.

[57] S. Shi, M. Schubert, and H. Boche, “Rate optimization for multiuser MIMO systems with linear processing,” *IEEE Trans. Signal Process*., vol. 56, no. 8, pp. 4020–4030, Aug. 2008.

[58] S. Serbetli and A. Yener, “Transceiver optimization for multiuser MIMO systems,” *IEEE Trans. Signal Process*., vol. 52, no. 1, pp. 214–226, Jan. 2004.

[59] S. Shi, M. Schubert, and H. Boche, “Downlink MMSE transceiver optimization for multiuser MIMO systems: duality and sum-MSE minimization,” *IEEE Trans. Signal Process*., vol. 55, no. 11, pp. 5436–5446, Nov. 2007.

[60] A. J. Tenenbaum and R. S. Adve, "Linear processing and sum throughput in the multiuser MIMO downlink," *IEEE Trans. Wireless Commun*., vol. 8, no. 5, pp. 2652–2661, May 2009.

[61] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, Aug. 2002.

[62] B. M. Hochwald, T. L. Marzetta, and V. Tarokh, "Multiple-antenna channel hardening and its implications for rate feedback and scheduling," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 1893–1909, Sept. 2004.

[63] P. Viswanath, D. N. C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1277–1294, Jun. 2002.

[64] P. Schulz-Rittich, A. Senst, U. Krause, and H. Meyr, "Increasing system throughput by time-varying beamforming in multiuser systems with slowly varying fading channels," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'03-Fall*), Orlando, FL, Oct. 2003, vol. 1, pp. 368–372.

[65] N. Jindal and A. Goldsmith, "Dirty-paper coding versus TDMA for MIMO broadcast channels," *IEEE Trans. Inf. Theory*, vol. 51, no. 5, pp. 1783–1794, May 2005.

[66] R. W. Heath, Jr., M. Airy, and A. J. Paulraj, "Multiuser diversity for MIMO wireless systems with linear receivers," in *Proc. 35th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2001, vol. 2, pp. 1194–1199.

[67] D. Aktas and H. El Gamal, "Multiuser scheduling for MIMO wireless systems," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'03-Fall*), Orlando, FL, Oct. 2003, vol. 3, pp. 1743–1747.

[68] D. Bartolomé, D. P. Palomar, and A. I. Pérez-Neira, "Real-time scheduling for wireless multiuser MISO systems under different fairness criteria," in *Proc. Int. Symp. Signal Process. and Its Applications* (*ISSPA'03*), Paris, France, Jul. 2003, vol. 1, pp. 213–216.

[69] Z. Shen, R. Chen, J. G. Andrews, R. W. Heath, Jr., and B. L. Evans, "Low complexity user selection algorithms for multiuser MIMO systems with block diagonalization," *IEEE Trans. Signal Process.*, vol. 54, no. 9, pp. 3658–3663, Sept. 2006.

[70] J. Wang, D. J. Love, and M. D. Zoltowski, "User selection with zero-forcing beamforming achieves the asymptotically optimal sum rate," *IEEE Trans. Signal Process*., vol. 56, no. 8, pp. 3713–3726, Aug. 2008.

[71] C. Guthy, W. Utschick, and G. Dietl, "Low-complexity linear zero-forcing for the MIMO broadcast channel," *IEEE J. Sel. Topics Signal Process*., vol. 3, no. 6, pp. 1106–1117, Dec. 2009.

[72] G. Dimic and N. D. Sidiropoulos, "On downlink beamforming with greedy user selection: performance analysis and a simple new algorithm," *IEEE Trans. Signal Process*., vol. 53, no. 10, pp. 3857–3868, Oct. 2005.

[73] B. C. Lim, W. A. Krzymień, and C. Schlegel, "Efficient sum rate maximization and resource allocation in block-diagonalized space-division multiplexing," *IEEE Trans. Veh. Technol*., vol. 58, no. 1, pp. 478–484, Jan. 2009.

[74] L. Jin, Z. Hu, and X. Gu, "A new scheduling algorithm with low complexity for multiuser multiple-input multiple-output downlink system," in *Proc. Int. Conf. Wireless Commun. Signal Process.* (*WCSP'09*), Nanjing, China, Nov. 2009, pp. 1–5.

[75] D. J. Mazzarese and W. A. Krzymień, "Scheduling algorithms and throughput maximization for the downlink of packet-data cellular systems with multiple antennas at the base station," *Wireless Pers. Commun*., vol. 43, no. 2, pp. 215–260, Oct. 2007.

[76] J. Jiang, R. M. Buehrer, and W. H. Tranter, "Greedy scheduling performance for a zero-forcing dirty-paper coded system," *IEEE Trans. Commun*., vol. 54, no. 5, pp. 789–793, May 2006.

[77] J. Dai, C. Chang, Z. Ye, and Y. S. Hung, "An efficient greedy scheduler for zero-forcing dirty-paper coding," *IEEE Trans. Commun*., vol. 57, no. 7, pp. 1939–1943, Jul. 2009.

[78] Q. Zhou, H. Dai, and H. Zhang, "Joint Tomlinson-Harashima precoding and scheduling for multiuser MIMO with imperfect feedback," in *Proc. IEEE Wireless Commun. Netw. Conf.* (*WCNC'06*), Las Vegas, NV, Apr. 2006, vol. 3, pp. 1233–1238.

[79] B. Ozdemir and O. Gurbuz, "Scheduling approach for MIMO with Tomlinson-Harashima precoding," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'06-Spr*.), Melbourne, Australia, May 2006, vol. 1, pp. 329–333.

[80] L.-N. Tran and E.-K. Hong, "Multiuser diversity for successive zero-forcing dirty paper coding: greedy scheduling algorithms and asymptotic performance analysis," *IEEE Trans. Signal Process*., vol. 58, no. 6, pp. 3411–3416, Jun. 2010.

[81] V. K. N. Lau, "Proportional fair space-time scheduling for wireless communications," *IEEE Trans. Commun*., vol. 53, no. 8, pp. 1353–1360, Aug. 2005.

[82] H. Viswanathan, S. Venkatesan, and H. Huang, "Downlink capacity evaluation of cellular networks with known-interference cancellation," *IEEE J. Sel. Areas Commun*., vol. 21, no. 5, pp. 802–811, Jun. 2003.

[83] Y. Rong and Y. Hua, "Space-time power scheduling of MIMO links – fairness and QoS considerations," *IEEE J. Sel. Topics Signal Process*., vol. 2, no. 2, pp. 171–180, Apr. 2008.

[84] M. Torabzadeh and W. Ajib, "Packet scheduling and fairness for multiuser MIMO systems," *IEEE Trans. Veh. Technol*., vol. 59, no. 3, pp. 1330–1340, Mar. 2010.

[85] G. Primolevo, O. Simeone, and U. Spagnolini, "Channel aware scheduling for broadcast MIMO systems with orthogonal linear precoding and fairness constraints," in *Proc. IEEE Int. Conf. Commun*. (*ICC'05*), Seoul, Korea, May 2005, vol. 4, pp. 2749–2753.

[86] M. Kobayashi and G. Caire, "Joint beamforming and scheduling for a multi-antenna downlink with imperfect transmitter channel knowledge," *IEEE J. Sel. Areas Commun*., vol. 25, no. 7, pp. 1468–1477, Sept. 2007.

[87] A. Sang, X. Wang, M. Madihian, and R. D. Gitlin, "A flexible downlink scheduling scheme in cellular packet data systems," *IEEE Trans. Wireless Commun*., vol. 5, no. 3, pp. 568–577, Mar. 2006.

[88] A. Khan, R. Vesilo, and I. B. Collings, "Efficient user selection algorithms for wireless broadcast channels," in *Proc. 2nd Int. Conf. Wireless Broadband and Ultra Wideband Commun*. (*AusWireless 2007*), Sydney, Australia, Aug. 2007, pp. 63–68.

[89] R. C. Elliott and W. A. Krzymień, "Scheduling algorithms for high-throughput packet data service in cellular radio systems," *Canadian J. Electr. Comput. Eng*., vol. 29, no. 1, pp. 117–127, Jan.-Apr. 2004.

[90] R. C. Elliott and W. A. Krzymień, "Scheduling of wireless packet data transmissions," U.S. Patent 7,295,513, Nov. 13, 2007.

[91] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Commun. Mag*., vol. 39, no. 2, pp. 150–154, Feb. 2001.

[92] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: the exponential rule," *American Mathematical Society Translations, Series 2*, vol. 207, pp. 185–202, 2002.

[93] X. Liu, E. K. P. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Comput. Netw.*, vol. 41, no. 4, pp. 451–474, Mar. 2003.

[94] C. Wang and R. D. Murch, "Optimal downlink multi-user MIMO cross-layer scheduling using HOL packet waiting time," *IEEE Trans. Wireless Commun*., vol. 5, no. 10, pp. 2856–2862, Oct. 2006.

[95] Z. Zhang, Y. He, and E. K. P. Chong, "Opportunistic downlink scheduling for multiuser OFDM systems," in *Proc. IEEE Wireless Commun. Netw. Conf.* (*WCNC'05*), New Orleans, LA, Mar. 2005, vol. 2, pp. 1206–1212.

[96] J. Hang, Z. Fan, and F. She, "Performance analysis of power optimization and user scheduling in multi-user MIMO-OFDM systems," in *Proc. IEEE Int. Conf. Intelligent Comput. and Intelligent Systems* (*ICIS'09*), Shanghai, China, Nov. 2009, vol. 3, pp. 238–242.

[97] Z. Rosberg, A. Cantoni, and R. P. Liu, "Resource allocation for QoS multiuser MIMO with zero forcing and MMSE beamforming," in *Proc. 18th Int. Workshop on Quality of Service* (*IWQoS 2010*), Beijing, China, Jun. 2010, pp. 1–6.

[98] M. Kountouris, A. Pandharipande, H. Kim, and D. Gesbert, "QoS-based user scheduling for multiuser MIMO systems," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'05-Spr*.), Stockholm, Sweden, May-Jun. 2005, vol. 1, pp. 211–215.

[99] J. Torres, V. Morillo-Velarde, B. Soret, M. C. Aguayo-Torres, and J. T. Entrambasaguas, "Cross-layer user multiplexing algorithms evaluation in MIMO OFDM wireless systems," in *Proc. 16th IST Mobile and Wireless Commun. Summit*, Budapest, Hungary, Jul. 2007, pp. 1–5.

[100] M. Li, Y. Xu, and Y. Cai, "A cross-layer packet scheduling and antenna selection scheme for multi-service MIMO systems," in *Proc. 4th Int. Conf. Wireless Commun., Netw. and Mobile Comput*. (*WiCOM'08*), Dalian, China, Oct. 2008, pp. 1–4.

[101] O. Souihli and T. Ohtsuki, "Joint feedback and scheduling scheme for service-differentiated multiuser MIMO systems," *IEEE Trans. Wireless Commun*., vol. 9, no. 2, pp. 528–533, Feb. 2010.

[102] S. Lee and J. S. Thompson, "QoS-guaranteed sequential user selection in multiuser MIMO downlink channels," in *Proc. IEEE Veh. Technol. Conf*. (*VTC'07-Spr*.), Dublin, Ireland, Apr. 2007, pp. 1926–1930.

[103] T. Hui, W. Shuang, G. Youjun, S. Qiaoyun, and Z. Ping, "A QoS-guarantee resource allocation scheme in multi-user MIMO-OFDM systems," in *Proc. IEEE Veh. Technol. Conf*. (*VTC'07-Fall*), Baltimore, MD, Sept.-Oct. 2007, pp. 1802–1806.

[104] R. C. Elliott and W. A. Krzymień, "Downlink scheduling for multiple antenna systems with dirty paper coding via genetic algorithms," in *Proc. IEEE Veh. Technol. Conf*. (*VTC'07-Spr*.), Dublin, Ireland, Apr. 2007, pp. 2339–2343.

[105] R. C. Elliott and W. A. Krzymień, "Downlink scheduling for multiple antenna multi-carrier systems with dirty paper coding via genetic algorithms," in *Multi-Carrier Spread Spectrum 2007*, S. Plass et al., Eds. Dordrecht, The Netherlands: Springer, 2007, pp. 47–56.

[106] R. C. Elliott and W. A. Krzymień, "Downlink scheduling via genetic algorithms for multiuser single-carrier and multicarrier MIMO systems with dirty paper coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 7, pp. 3247–3262, Sept. 2009.

[107] N. Jindal, W. Rhee, S. Vishwanath, S. A. Jafar, and A. Goldsmith, "Sum power iterative water-filling for multi-antenna Gaussian broadcast channels," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1570–1580, Apr. 2005.

[108] R. Böhnke and K.-D. Kammeyer, "Weighted sum rate maximization for the MIMO-downlink using a projected conjugate gradient algorithm," in *Proc. Int. Workshop on Cross Layer Design* (*IWCLD'07*), Jinan, China, Sept. 2007, pp. 82–85.

[109] V. K. N. Lau, "Optimal downlink space-time scheduling design with convex utility functions—Multiple-antenna systems with orthogonal spatial multiplexing," *IEEE Trans. Veh. Technol.*, vol. 54, no. 4, pp. 1322–1333, Jul. 2005.

[110] J.-Y. Le Boudec. (2008, Dec. 9). *Rate Adaptation, Congestion Control and Fairness: A Tutorial* [Online]. Available: http://ica1www.epfl.ch/PS_files/LEB3132.pdf

[111] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 1st ed. Ann Arbor, MI: Univ. Michigan Press, 1975.

[112] D. S. Weile and E. Michielssen, "Genetic algorithm optimization applied to electromagnetics: a review," *IEEE Trans. Antennas Propagat*., vol. 45, no. 3, pp. 343–353, Mar. 1997.

[113] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, New Series, vol. 220, no. 4598, pp. 671–680, Mar. 1983.

[114] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput*., vol. 6, no. 1, pp. 58–73, Feb. 2002.

[115] C. Mattiussi and D. Floreano, “Analog genetic encoding for the evolution of circuits and networks,” *IEEE Trans. Evol. Comput*., vol. 11, no. 5, pp. 596–607, Oct. 2007.

[116] Y. Yu and Y. Xinjie, “Cooperative coevolutionary genetic algorithm for digital IIR filter design,” *IEEE Trans. Ind. Electron*., vol. 54, no. 3, pp. 1311–1318, Jun. 2007.

[117] X.-M. Hu, J. Zhang, Y. Yu, H. S.-H. Chung, Y.-L. Li, Y.-H. Shi, and X.-N. Luo, “Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks,” *IEEE Trans. Evol. Comput*., vol. 14, no. 5, pp. 766–781, Oct. 2010.

[118] Z. Zhao, Z. Peng, S. Zheng, and J. Shang, “Cognitive radio spectrum allocation using evolutionary algorithms,” *IEEE Trans. Wireless Commun*., vol. 8, no. 9, pp. 4421–4425, Sept. 2009.

[119] H.-Y. Lu and W.-H. Fang, “Joint transmit/receive antenna selection in MIMO systems based on the priority-based genetic algorithm," *IEEE Antennas Wireless Propagat. Lett*., vol. 6, pp. 588–591, 2007.

[120] B. G. W. Craenen, A. E. Eiben, and E. Marchiori, “How to handle constraints with evolutionary algorithms,” in *The Practical Handbook of Genetic Algorithms: Applications*, 2nd ed., L. Chambers, Ed. Boca Raton, FL: Chapman & Hall / CRC, 2001, pp. 341–361.

[121] S. Lin and D. J. Costello Jr., *Error Control Coding*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2004.

[122] *cdma2000 High Rate Packet Data Air Interface Specification*, 3rd Generation Partnership Project 2 (3GPP2) Standard C.S0024-B v3.0, Sept. 2009.

[123] G. L. Stüber, *Principles of Mobile Communication*, 2nd ed. Boston, MA: Kluwer, 2001, ch. 2.4.

[124] W. Yu and W. Rhee, “Degrees of freedom in wireless multiuser spatial multiplex systems with multiple antennas,” *IEEE Trans. Commun*., vol. 54, no. 10, pp. 1747–1753, Oct. 2006.

[125] R. C. Elliott and W. A. Krzymień, “On the convergence of genetic scheduling algorithms for downlink transmission in multi-user MIMO systems,” in *Proc. Int. Symp. Wireless Pers. Multimedia Commun.* (*WPMC’08*), Lapland, Finland, Sept. 2008.

[126] R. C. Elliott and W. A. Krzymień, “On the convergence of genetic scheduling algorithms for downlink transmission in multi-user MIMO systems,” *Wireless Pers. Commun*., Sept. 2010. [Online]. doi: 10.1007/s11277-010-0131-4

[127] E. K. P. Chong and S. H. Żak, *An Introduction to Optimization*, 2nd ed. New York, NY: Wiley, 2001.

[128] M. Srinivas and L. M. Patnaik, “Genetic algorithms: a survey,” *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[129] W. M. Spears, “Adapting crossover in evolutionary algorithms,” in *Proc. 4th Annu. Evol. Programming Conf.*, San Diego, CA, Mar. 1995, pp. 367–384.

[130] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, 2nd ed. Hoboken, NJ: Wiley, 2004.

[131] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd ed. Hoboken, NJ: Wiley, 2006.

[132] S. Sigdel, R. C. Elliott, W. A. Krzymień, and M. Al-Shalash, "Greedy and genetic user scheduling algorithms for multiuser MIMO systems with block diagonalization," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'09-Fall*), Anchorage, AK, Sept. 2009, pp. 1–6.

[133] R. C. Elliott, S. Sigdel, W. A. Krzymień, M. Al-Shalash, and A. C. K. Soong, "Genetic and greedy user scheduling for multiuser MIMO systems with successive zero-forcing," in *Proc. 5th IEEE Broadband Wireless Access Workshop* (*2009 IEEE GLOBECOM Workshops*), Honolulu, HI, Nov. 2009, pp. 1–6.

[134] R. C. Elliott, S. Sigdel, and W. A. Krzymień, "Low complexity greedy, genetic, and hybrid user scheduling algorithms for multiuser MIMO systems with successive zero-forcing," submitted to *Eur. Trans. Telecommun*., Mar. 2011.

[135] B. Farhang-Boroujeny, Q. Spencer, and A. L. Swindlehurst, "Layering techniques for space–time communications in multi-user networks," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'03-Fall*), Orlando, FL, Oct. 2003, vol. 2, pp. 1339–1342.

[136] C.-B. Chae, D. Mazzarese, N. Jindal, and R. W. Heath, Jr., "Coordinated beamforming with limited feedback in the MIMO broadcast channel," *IEEE J. Sel. Areas Commun*., vol. 26, no. 8, pp. 1505–1515, Oct. 2008.

[137] Z. Shen, R. Chen, J. G. Andrews, R. W. Heath, Jr., and B. L. Evans, "Sum capacity of multiuser MIMO broadcast channels with block diagonalization," *IEEE Trans. Wireless Commun.*, vol. 6, no. 6, pp. 2040–2045, Jun. 2007.

[138] R. Chen, R. W. Heath, Jr., and J. G. Andrews, "Transmit selection diversity for unitary precoded multiuser spatial multiplexing systems with linear receivers," *IEEE Trans. Signal Process*., vol. 55, no. 3, pp. 1159–1171, Mar. 2007.

[139] S. Sigdel and W. A. Krzymień, "Simplified transmit covariance optimization and user ordering algorithm for successive zero-forcing precoding," in *Proc. IEEE 10th Workshop on Signal Process. Advances in Wireless Commun*. (*SPAWC'09*), Perugia, Italy, Jun. 2009, pp. 235–239.

[140] S. Sigdel and W. A. Krzymień, "User scheduling for network MIMO systems with successive zero-forcing precoding," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'10-Fall*), Ottawa, Canada, Sept. 2010, pp. 1–6.

[141] S. Sigdel and W. A. Krzymień, "Simplified fair scheduling and antenna selection algorithms for multiuser MIMO orthogonal space-division multiplexing downlink," *IEEE Trans. Veh. Technol*., vol. 58, no. 3, pp. 1329–1344, Mar. 2009.

[142] S. Sigdel and W. A. Krzymień, "Efficient user selection and ordering algorithms for successive zero-forcing precoding for multiuser MIMO downlink," in *Proc. IEEE Veh. Technol. Conf.* (*VTC'09-Spr.*), Barcelona, Spain, Apr. 2009, pp. 1–6.

[143] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The John Hopkins Univ. Press, 1996.

[144] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," J. Symbolic Computation, vol. 9, no. 3, pp. 251–280, Mar. 1990.

[145] C.-H. Guo and N. J. Higham, "A Schur-Newton method for the matrix *p*th root and its inverse," *SIAM J. Matrix Anal. Appl.*, vol. 28, no. 3, pp. 788–804, 2006.

[146] G. J. Foschini, K. Karakayali, and R. A. Valenzuela, "Coordinating multiple antenna cellular networks to achieve enormous spectral efficiency," *IEE Proc.-Commun.*, vol. 153, no. 4, pp. 548–555, Aug. 2006.

[147] X. Zhang and J. Lee, "Low complexity MIMO scheduling with channel decomposition using capacity upperbound," *IEEE Trans. Commun.*, vol. 56, no. 6, pp. 871–876, Jun. 2008.

[148] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, Jun. 1989.

[149] M. Naeem and D. C. Lee, "Estimation of distribution algorithm for scheduling in uplink multiuser wireless communication system," in *Proc. IEEE Symp. Computational Intell. in Scheduling* (*CI-Sched'09*), Nashville, TN, Mar.-Apr. 2009, pp. 36–41.

[150] *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, P. Larrañaga and J. A. Lozano, Eds. Norwell, MA: Kluwer Academic Publishers, 2002.

[151] J. Liu, Y. T. Hou, and H. D. Sherali, "On the maximum weighted sum-rate of MIMO Gaussian broadcast channels," in *Proc. IEEE Int. Conf. Commun*. (*ICC'08*), Beijing, China, May 2008, pp. 3664–3668.

[152] R. C. Elliott and W. A. Krzymień, "Improved and weighted sum rate maximization for successive zero-forcing in multiuser MIMO systems," submitted to *EURASIP J. Wireless Commun. Netw.*, Nov. 2010.

[153] W. Yu, "Uplink-downlink duality via minimax duality," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 361–374, Feb. 2006.

[154] L. Zhang, R. Zhang, Y.-C. Liang, Y. Xin, and H. V. Poor, "On Gaussian MIMO BC-MAC duality with multiple transmit covariance constraints," in *Proc. IEEE Int. Symp. Inf. Theory* (*ISIT'09*), Seoul, Korea, Jun.-Jul. 2009, pp. 2502–2506.

[155] R. Hunger, M. Joham, and W. Utschick, "On the MSE-duality of the broadcast channel and the multiple access channel," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 698–713, Feb. 2009.

[156] M. Codreanu, A. Tölli, M. Juntti, and M. Latva-aho, "Uplink-downlink SINR duality via Lagrange duality," in *Proc. IEEE Wireless Commun. Netw. Conf.* (*WCNC'08*), Las Vegas, NV, Mar.-Apr. 2008, pp. 1160–1165.

[157] R. Hunger and M. Joham, "A general rate duality of the MIMO multiple access channel and the MIMO broadcast channel," in *Proc. IEEE Global Telecommun. Conf.* (*GLOBECOM'08*), New Orleans, LA, Nov.-Dec. 2008, pp. 1–5.

[158] W. Yu and T. Lan, "Transmitter optimization for the multi-antenna downlink with per-antenna power constraints," *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2646–2660, Jun. 2007.

[159] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY: Springer, 2006.

[160] S. Ye and R. S. Blum, "Optimized signalling for MIMO interference systems with feedback," *IEEE Trans. Signal Process.*, vol. 51, no. 11, pp. 2839–2848, Nov. 2003.

[161] R. Hunger, D. A. Schmidt, M. Joham, and W. Utschick, "A general covariance-based optimization framework using orthogonal projections," in *Proc. IEEE 9th Workshop on Signal Process. Advances in Wireless Commun*. (*SPAWC'08*), Recife, Brazil, pp. 76–80, Jul. 2008.

[162] K. B. Petersen and M. S. Pedersen. (2008, Oct.). The matrix cookbook. *Technical Univ. of Denmark*. Ver.20081110. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274

[163] J. Liu, Y. T. Hou, Y. Shi, H. D. Sherali, and S. Kompella, "On the capacity of multiuser MIMO networks with interference," *IEEE Trans. Wireless Commun*., vol. 7, no. 2, pp. 488–494, Feb. 2008.

[164] J. Liu, Y. T. Hou, and H. D. Sherali, "Optimal power allocation for achieving perfect secrecy capacity in MIMO wire-tap channels," in *Proc. 43rd Conf. Inf. Sciences Systems* (*CISS'09*), Baltimore, MD, Mar. 2009, pp. 606–611.

[165] Y. Shi, Y. T. Hou, and H. D. Sherali, "Cross-layer optimization for data rate utility problem in UWB-based ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 6, pp. 764–777, Jun. 2008.

[166] T. Yoo, N. Jindal, and A. Goldsmith, "Multi-antenna downlink channels with limited feedback and user selection," *IEEE J. Sel. Areas Commun*., vol. 25, no. 7, pp. 1478–1491, Sept. 2007.

[167] M. K. Karakayali, G. J. Foschini, and R. A. Valenzuela, "Network coordination for spectrally efficient communications in cellular systems," *IEEE Wireless Commun*., vol. 13, no. 4, pp. 56–61, Aug. 2006.

[168] M. Sawahashi, Y. Kishiyama, A. Morimoto, D. Nishikawa, and M. Tanno, "Coordinated multipoint transmission/reception techniques for LTE-advanced," *IEEE Wireless Commun*., vol. 17, no. 3, pp. 26–34, Jun. 2010.

[169] R. Gonzalez, *Data Analysis for Experimental Design*. New York, NY: The Guilford Press, 2009.

[170] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, England: Cambridge University Press, 2004.

[171] A. Hjørungnes and D. Gesbert, "Complex-valued matrix differentiation: techniques and key results," *IEEE Trans. Signal Process*., vol. 55, no. 6, pp. 2740–2746, Jun. 2007.

[172] E. W. Weisstein. (2010, Dec.). Least squares fitting--polynomial. *MathWorld--A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html

Appendix A

Validation of the Simulation Model

The simulation model used in this work assumed a Rayleigh fading channel, which represents a system with a rich scattering environment, wherein the received signal has no significant specular / line-of-sight component. When examining the proportional fairness criterion, we also assumed the existence of log-normal shadowing. The simulation results were determined via Monte Carlo methods. All simulations were conducted using Matlab. To verify the simulations overall, we examine three factors in more detail. First, we check whether the channel gains are indeed complex Gaussian-distributed values, as they should be. Second, we examine the error in the Monte Carlo results. Lastly, we compare the simulation results to findings published in the literature.

A.1 Complex Gaussian Verification

Gaussian variables are generated in Matlab using the "randn" function. The function generates values for a Gaussian-distributed variable with zero mean and unit variance. To ensure that the values are different each time Matlab is run, the state of the random number generator can be set to a value based on the system clock.

Figure A.1 shows the approximate PDF and CDF of a set of 100,000 values of a real Gaussian random variable generated by Matlab. The PDF $p_x(x)$ and CDF $F_x(x)$ of a Gaussian random variable x with mean μ_x and variance σ_x^2 respectively are [20]:

$$p_x(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-(x-\mu_x)^2/2\sigma_x^2}; \tag{A.1}$$

$$F_x(x) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{(x-\mu_x)/\sqrt{2}\sigma_x} e^{-t^2} dt\,. \tag{A.2}$$

It can be seen that the random variables from Matlab, which would be used in simulations, closely match the theoretical distributions. The plots for the simulated and analytical results overlap. The mean and variance of the random variables were –0.0039 and 1.0034 respectively, which are very close to the expected values of 0 and 1. Testing

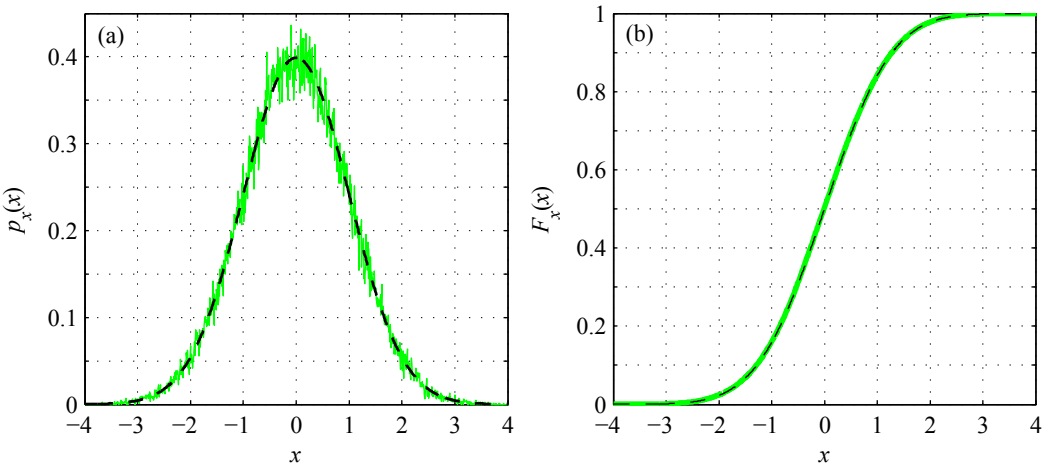

Figure A.1: Comparison of the distribution of values of a random Gaussian variable generated by Matlab (solid lines) with the theoretical Gaussian distribution (dashed lines). (a) PDF, (b) CDF.

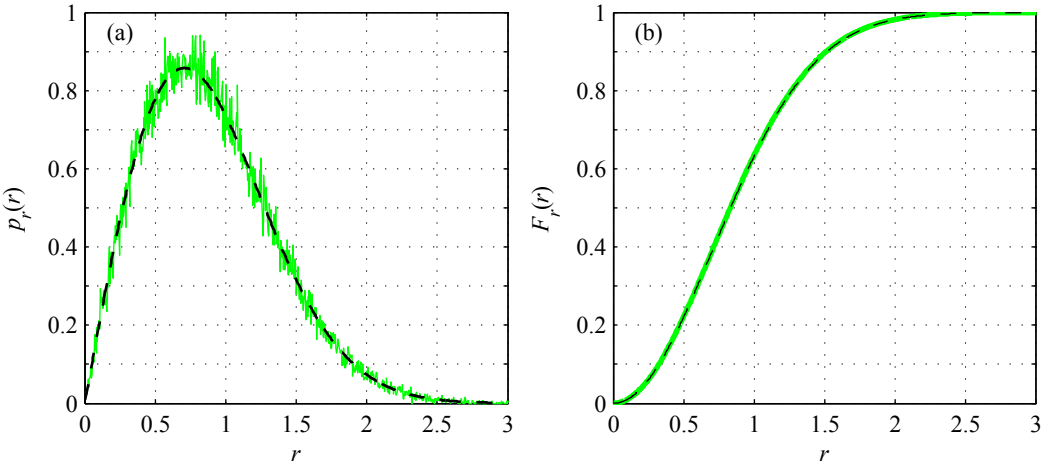

Figure A.2: Comparison of the distribution of values of a random Rayleigh variable generated by Matlab (solid lines) with the theoretical Rayleigh distribution (dashed lines). (a) PDF, (b) CDF.

the covariance of the random variables $x(n)$ and $x(n + t)$, for offsets of $t = 1$ to 1000, the maximum magnitude of the covariance observed was 0.0101 for any offset, which indicates that the Gaussian variables generated by Matlab are also uncorrelated, as desired. These results thus also verify the generation of log-normal shadowing values, which are also Gaussian-distributed. The only difference is that they are in units of dB, and have a standard deviation of 8 dB. The required standard deviation can be obtained easily by multiplying the Matlab-generated Gaussian variables by 8.

The Rayleigh channel requires complex-valued Gaussian variables with zero mean and unit variance. These can be generated from the real-valued Gaussian variables by $r = \left(x_R + ix_I\right)/\sqrt{2}$, where x_R and x_I are i.i.d. Gaussian random variables with zero mean and unit variance. Figure A.2 shows the PDF and CDF of the magnitude of 100,000 values of

such a complex Gaussian variable generated in Matlab. The magnitude of the complex Gaussian variables should be Rayleigh distributed with a PDF and CDF of [20]:

$$p_r(r) = \frac{r}{\sigma_x^2} e^{-r^2/2\sigma_x^2}, \quad r \geq 0, \tag{A.3}$$

$$F_r(r) = 1 - e^{-r^2/2\sigma_x^2}, \quad r \geq 0. \tag{A.4}$$

σ_x^2 is the variance of the Gaussian variables making up the real and imaginary parts of the complex variable; in this case, $\sigma_x^2 = 1/2$.

Once again, it can be seen that the generated and theoretical distributions match very closely. The generated complex Gaussian variables had a mean of 0.0015 – 0.0008*i* and a variance of 0.9979, which again are close to the desired values. The magnitude of the complex Gaussian variables should theoretically have a mean of $\mu_r = \sigma_x\sqrt{\pi/2} = 0.8862$ and a variance of $\sigma_r^2 = (2-\pi/2)\sigma_x^2 = 0.2146$ [20]. The generated variables had a mean of 0.8857 and a variance of 0.2135, which are very near their theoretical values.

Thus, given the overall closeness of the generated random variables to their theoretical distributions, means, and variances, the generation of channel matrices for our Rayleigh channel model is validated.

A.2 Error in Monte Carlo Simulations

The figures for the simulation results throughout the thesis generally are meant to show an average of some sort, most commonly the average throughput or sum rate provided by the scheduling algorithms. What this is supposed to mean is the ergodic average; that is, the expected value obtained by averaging over all possible channel realizations. However, in practice for the simulations, we do not have the true ergodic mean. Instead, we have a sample mean taken over multiple independent realizations of the channel. Thus, there is some error inherent in the reported values. Strictly speaking, the figures should actually have error bars to depict the uncertainty in the mean. However, in practice, the error bars are almost never included in published literature. While there is an error in the mean, if a sufficiently large number of realizations are averaged, the sample mean closely approximates the ergodic mean. If the error is small enough, then leaving off the error bars is justified if their range cannot be viewed clearly in the figures.

The error in the simulation is found from the standard error of the mean, given by:

$$S_E = \hat{\sigma} \Big/ \sqrt{N_{samp}} \ . \tag{A.5}$$

$\hat{\sigma}$ is the standard deviation of the results for the independent runs of the simulation, and N_{samp} is the number of samples used to calculate the mean [169]. The standard error is an estimate of the standard deviation of the sample mean. Clearly, as the number of samples increases, the standard error decreases.

The confidence interval for the mean can then be found from the standard error. In general, a certain percentage confidence interval can be found by a certain number of standard errors around the sample mean. For example, if the samples for the mean follow a normal (Gaussian) distribution[1], then the 95% confidence interval will be $\pm 1.96 S_E$ around the mean [169]. It can then be said that we are 95 percent confident that the true ergodic (or population) mean falls within that interval.

We illustrate this for the case of our genetic scheduling algorithm for block diagonalization from Chapter 5, with $M_T = 8$, $N = 2$, and $K_0 = 4$, at an SNR of 10 dB. Figure A.3 shows how the average sum rate and the 95% confidence interval for the average changes as the number of samples increases. When $N_{samp} = 1000$, the 95% confidence interval is about ±0.06 bit/s/Hz, and the sum-rate curve is fairly smooth. Thus, 1000 runs of the Monte Carlo simulation should in general be enough to obtain a very good estimate of the mean. However, in many cases, we still run simulations for even longer (when feasible) for even tighter bounds on the results. In general, most simulations were run for either 5000 or 10000 independent channel realizations. The main exceptions to this were some of the exhaustive search simulations. With the exhaustive search, the combinatorial complexity greatly increased the run times for the simulations, particularly for order-dependent cases like successive zero-forcing. With the exhaustive search, for larger K, to keep the total simulation time feasible, the total number of runs was limited to 1000. On the other hand, simulations for the proportional fairness criterion were run for a longer time, due to there being two sources of randomness in the channel matrices: the complex Gaussian channel gains for the Rayleigh channel, and the log-normal shadowing components. We note that since the Rayleigh channel gains change for each simulation run and for each transmit-receive antenna pair at each scheduling interval, while the shadowing is fixed for the length of a simulation run and is constant for all

[1] With a large number of samples, the distribution will closely approximate a normal distribution, due to the central limit theorem [20] (also assuming the mean and standard error are finite). If the number of samples is not sufficiently large, a t distribution can be used with $N_{samp} - 1$ degrees of freedom [169].

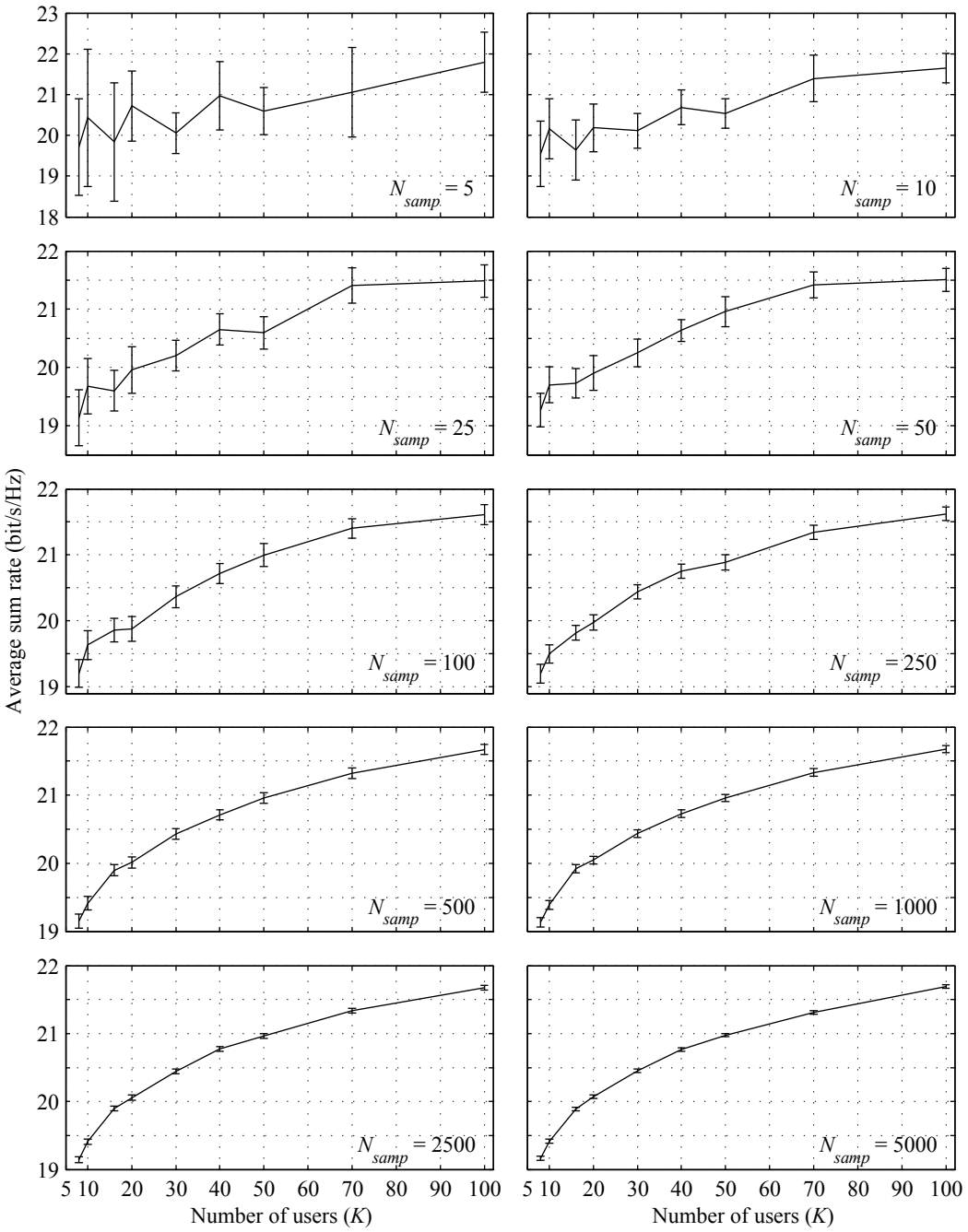

Figure A.3: Comparison of average sum rate and 95% confidence intervals vs. K for genetic scheduling algorithm for BD with varying number of Monte Carlo simulation runs N_{samp}; $M_T = 8$, $N = 2$, $K_0 = 4$, SNR = 10 dB.

receive antennas of a given user, the Rayleigh components account for a much larger proportion of the randomness in the simulations.

Overall, the largest 95% confidence intervals appear to occur for smaller K, and also at lower SNR (not shown above). This is reasonable; we expect there to be a larger variance in the supportable capacity when there is not a larger pool of users to choose

from, and when channel conditions dictate more often that less than the maximum number of users should be scheduled. However, in the worst case throughout our work, we generally see that the 95% confidence interval around the average sum rate / sum-capacity is still at most about ±0.06 bit/s/Hz, similar to what is seen with $N_{samp} = 1000$ in Figure A.3. The $N_{samp} = 5000$ case from Figure A.3 is fairly representative of the error for most of the cases in this thesis, regardless of the specific precoding method, SNR, etc.

A.3 Comparison with Published Results

One final check that can be done is to compare the results from our work with those that exist in prior literature. This is primarily meant to verify that the system simulations have been properly set up. Since almost all prior work focuses on maximizing the system sum rate, the comparisons can be made relative to our optimal results for exhaustive search scheduling. In cases where we simulate scheduling algorithms proposed by others, those results can also be compared for further verification of the setup.

For the results on DPC, we can compare our maximum throughput results[1] with those in [75] and [109]. The results of our exhaustive search from Figure 3.6(a) and Figure 3.7 match those for the sum-capacity in Figure 14 and Figure 11 of [75] for $M_T = 2$ and $N_R = 1$. Our results for $M_T = 4$ in Figure 3.7 are slightly below the sum-capacity in Figure 11 of [75], but we note that the sum-capacity therein does not assume that a maximum of M_T users is scheduled. However, our results are very close to the best "successive projections" scheduling algorithm of [75], which does only schedule M_T users. Thus, our exhaustive search results for $M_T = 4$ (with a maximum of M_T users) are also likely accurate. Lastly, our comparable results for ZFB in Chapter 3 are the same as those reported in [109].

For our results on BD scheduling, we can compare with those in [69] and [147]. Our exhaustive search results in Figure 5.1 match those in Figure 1 of [69] for $M_T = 4$ and $N = 2$ at 10 dB. Furthermore, our simulation results for the SCAHE scheduling algorithms in [69] also match those reported in [69]. Additionally, our exhaustive search and SCAHE results for $M_T = 8$ and $N = 2$ in Figure 5.2 at 10 dB match those reported in Figure 2 of [147].

There exist almost no results in the literature for SZF scheduling, so there is very little with which to compare. Exhaustive search results were obtained from simulations

[1] We unfortunately have been unable to locate other appropriate results in the literature for proportionally fair scheduling with DPC, to which we can compare our results in Chapter 3.

independently in [142]; these match our results in Figure 5.3. There are also SZF results for $K = K_0$ in [50]. We simulated these same cases for our comparison of SZF covariance optimization algorithms in Chapter 6. Our simulation results of Figure 6.1 and Figure 6.2 are the same as those for BD and SZF reported in Figures 5 and 6 of [50].

Thus, overall, we have first seen that the random values we have generated fall into the desired complex Gaussian distribution. Second, the confidence intervals for the mean sum rates of our simulations are quite tight. Last, the results of our simulations match those reported in other published works for the same system setup and certain identical scheduling algorithms. Thus, we can have a high degree of confidence in the validity of our results.

Appendix B

Optimality Conditions for DPC BC Scheduling

From equation (2.9) in Chapter 2, the rate for a given user $\pi(k)$ in a DPC system is given by:

$$R_{\pi(k)} = \log_2 \frac{\left| \mathbf{I} + \mathbf{H}_{\pi(k)} \left(\sum_{u \geq k} \mathbf{\Sigma}_{\pi(u)} \right) \mathbf{H}_{\pi(k)}^H \right|}{\left| \mathbf{I} + \mathbf{H}_{\pi(k)} \left(\sum_{u > k} \mathbf{\Sigma}_{\pi(u)} \right) \mathbf{H}_{\pi(k)}^H \right|}. \tag{B.1}$$

Without loss of generality, let the users be ordered such that $\pi(k) = k$. With non-interfering subcarriers, the rate for user k on subcarrier j is then:

$$R_{jk} = \log_2 \frac{\left| \mathbf{I} + \mathbf{H}_{jk} \left(\sum_{u \geq k} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{jk}^H \right|}{\left| \mathbf{I} + \mathbf{H}_{jk} \left(\sum_{u > k} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{jk}^H \right|}. \tag{B.2}$$

Taking the constraints mentioned in Chapter 3 into account, the optimization problem $G_{\mathbb{S},\pi}^*$ of equation (3.6) can be expressed with Lagrange multipliers [170] as:

$$\begin{aligned} \max_{\left(\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right)} & G\left(\mathbb{S};\pi;\mathbf{\Sigma}_{11},\ldots,\mathbf{\Sigma}_{N_C K}\right) - \sum_{j=1}^{N_C} \lambda_j \left(\sum_{k \in \mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) - P/N_C \right) \\ & - \sum_{j,k} \mu_{jk} Tr\left(\mathbf{\Sigma}_{jk}\right) + \sum_{\forall j,k:\mathbf{\Omega}_{jk} \succeq 0} Tr\left(\mathbf{\Omega}_{jk} \mathbf{\Sigma}_{jk}\right) \end{aligned}. \tag{B.3}$$

The final term in equation (B.3) ensures that the transmit covariance matrices $\mathbf{\Sigma}_{jk}$ are positive semidefinite, given that the variables $\mathbf{\Omega}_{jk}$ are positive semidefinite. Since the trace of a matrix is the sum of its eigenvalues, if $\mathbf{\Sigma}_{jk}$ is not positive semidefinite, it will have a negative eigenvalue, and hence there will exist $\mathbf{\Omega}_{jk}$ such that the trace of $\mathbf{\Omega}_{jk}\mathbf{\Sigma}_{jk}$ and the overall equation will not be maximized compared to a case for which $\mathbf{\Sigma}_{jk}$ were positive semidefinite and had non-negative eigenvalues.

Let s_{jk} be a binary variable indicating if user k is scheduled on subcarrier j. The Karush-Kuhn-Tucker [170] conditions for optimality are then:

$$\lambda_j \left(\sum_{k \in \mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) - P/N_C \right) = 0, \quad j = 1,..,N_C; \tag{B.4}$$

$$\mu_{jk} Tr\left(\mathbf{\Sigma}_{jk}\right) = 0; \tag{B.5}$$

$$\frac{\partial G}{\partial \mathbf{\Sigma}_{jk}} - \lambda_j s_{jk} \mathbf{I}_{M_T} - \mu_{jk} \mathbf{I}_{M_T} + \mathbf{\Omega}_{jk}^T = \mathbf{0}_{M_T \times M_T}, \quad \forall j,k : s_{jk} = 1. \tag{B.6}$$

We can expand $\partial G / \partial \mathbf{\Sigma}_{jk}$ as:

$$\frac{\partial G}{\partial \mathbf{\Sigma}_{jk}} = \sum\nolimits_i \frac{\partial G}{\partial R_i} \underbrace{\frac{\partial R_i}{\partial R_{ji}}}_{=1} \frac{\partial R_{ji}}{\partial \mathbf{\Sigma}_{jk}}. \tag{B.7}$$

Lemma: The partial derivatives $\partial R_{ji} / \partial \mathbf{\Sigma}_{jk}$ are given by:

$$\frac{\partial R_{ji}}{\partial \mathbf{\Sigma}_{jk}} = \begin{cases} \mathbf{H}_{ji}^T \left[\mathbf{I} + \mathbf{H}_{ji}^* \left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T \right) \mathbf{H}_{ji}^T \right]^{-1} \mathbf{H}_{ji}^* - \mathbf{H}_{ji}^T \left[\mathbf{I} + \mathbf{H}_{ji}^* \left(\sum_{u > i} \mathbf{\Sigma}_{ju}^T \right) \mathbf{H}_{ji}^T \right]^{-1} \mathbf{H}_{ji}^*, & k > i \\ \mathbf{H}_{ji}^T \left[\mathbf{I} + \mathbf{H}_{ji}^* \left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T \right) \mathbf{H}_{ji}^T \right]^{-1} \mathbf{H}_{ji}^*, & k = i \\ \mathbf{0}_{M_T \times M_T}, & k < i \end{cases}, \tag{B.8}$$

and in particular,

$$\frac{\partial R_{j1}}{\partial \mathbf{\Sigma}_{j1}} = \mathbf{H}_{j1}^T \left[\mathbf{I} + \mathbf{H}_{j1}^* \left(\sum_{u} \mathbf{\Sigma}_{ju}^T \right) \mathbf{H}_{j1}^T \right]^{-1} \mathbf{H}_{j1}^* \succeq 0, \tag{B.9}$$

where ^{T} and * are the (non-Hermitian) transpose and conjugate operators, respectively.

Proof: We can replace the $\log_2$ in (B.2) with log without affecting the maximization. Then, assuming $k > i$, we can expand (B.2) as:

$$\begin{aligned} R_{ji} &= \log \left| \mathbf{I} + \mathbf{H}_{ji} \left(\sum_{u \geq i} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{ji}^H \right| - \log \left| \mathbf{I} + \mathbf{H}_{ji} \left(\sum_{u > i} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{ji}^H \right| \\ &= \log \left| \mathbf{I} + \mathbf{H}_{ji} \mathbf{\Sigma}_{jk} \mathbf{H}_{ji}^H + \mathbf{H}_{ji} \left(\sum_{u \geq i, u \neq k} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{ji}^H \right| \\ &\quad - \log \left| \mathbf{I} + \mathbf{H}_{ji} \mathbf{\Sigma}_{jk} \mathbf{H}_{ji}^H + \mathbf{H}_{ji} \left(\sum_{u > i, u \neq k} \mathbf{\Sigma}_{ju} \right) \mathbf{H}_{ji}^H \right| \end{aligned}. \tag{B.10}$$

If $k < i$, $\mathbf{\Sigma}_{jk}$ does not appear in R_{ji}, and thus $\partial R_{ji} / \partial \mathbf{\Sigma}_{jk} = \mathbf{0}_{M_T \times M_T}$. First, we consider $k \geq i$. Making some substitutions, let $g(f) = \log(f)$, $f(\mathbf{U}) = |\mathbf{U}|$, $\mathbf{A}_0 = \mathbf{H}_{ji}$, $\mathbf{A}_1 = \mathbf{H}_{ji}^H$, and

$\mathbf{U} = \mathbf{I} + \mathbf{H}_{ji}\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}\right)\mathbf{H}_{ji}^{H}$. Then, based on the tables and techniques in [162],[171], we can calculate the derivative of the first half of R_{ji} as:

$$\begin{aligned}
\partial g &= f^{-1}\partial f = f^{-1}\left[\left|\mathbf{U}\right| \cdot Tr\left\{\mathbf{U}^{-1}\partial\mathbf{U}\right\}\right] \\
&= \left(\left|\mathbf{U}\right|\right)^{-1}\left|\mathbf{U}\right| \cdot Tr\left\{\mathbf{U}^{-1}\partial\mathbf{U}\right\} = Tr\left\{\mathbf{U}^{-1}\partial\mathbf{U}\right\} \\
&= Tr\left\{\mathbf{U}^{-1}\partial\left[\mathbf{I} + \mathbf{A}_0\mathbf{\Sigma}_{jk}\mathbf{A}_1 + \mathbf{A}_0\left(\sum_{u \geq i, u \neq k} \mathbf{\Sigma}_{ju}\right)\mathbf{A}_1\right]\right\} \\
&= Tr\left\{\mathbf{U}^{-1}\mathbf{A}_0\partial\mathbf{\Sigma}_{jk}\mathbf{A}_1\right\} + Tr\left\{\mathbf{U}^{-1}\mathbf{A}_0\left(\sum_{u \geq i, u \neq k} \partial\mathbf{\Sigma}_{ju}\right)\mathbf{A}_1\right\} \\
&= Tr\left\{\mathbf{A}_1\mathbf{U}^{-1}\mathbf{A}_0\partial\mathbf{\Sigma}_{jk}\right\} + Tr\left\{\mathbf{U}^{-1}\mathbf{A}_0\left(\sum_{u \geq i, u \neq k} \partial\mathbf{\Sigma}_{ju}\right)\mathbf{A}_1\right\}
\end{aligned} \quad . \tag{B.11}$$

Note that the second half of the last line above does not contain the term $\partial\mathbf{\Sigma}_{jk}$. Based on [171], we therefore have:

$$\begin{aligned}
\frac{\partial g}{\partial \mathbf{\Sigma}_{jk}} &= \mathbf{A}_0^T\left(\mathbf{U}^{-1}\right)^T\mathbf{A}_1^T + 0 = \mathbf{A}_0^T\left(\mathbf{U}^T\right)^{-1}\mathbf{A}_1^T \\
&= \mathbf{A}_0^T\left[\left\{\mathbf{I} + \mathbf{A}_0\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}\right)\mathbf{A}_1\right\}^T\right]^{-1}\mathbf{A}_1^T = \mathbf{A}_0^T\left[\mathbf{I} + \mathbf{A}_1^T\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T\right)\mathbf{A}_0^T\right]^{-1}\mathbf{A}_1^T \\
&= \mathbf{H}_{ji}^T\left[\mathbf{I} + \left(\mathbf{H}_{ji}^H\right)^T\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\left(\mathbf{H}_{ji}^H\right)^T = \mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^*
\end{aligned} \quad . \tag{B.12}$$

We now note that the second half of (B.10) is basically the same form as the first half, provided now that $k > i$. If so, then we have:

$$\frac{\partial R_{ji}}{\partial \mathbf{\Sigma}_{jk}} = \mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u \geq i} \mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^* - \mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u > i} \mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^* . \tag{B.13}$$

However, if $k = i$, the second half of (B.10) disappears, and $\partial R_{ji} / \partial \mathbf{\Sigma}_{jk} = \partial g / \partial \mathbf{\Sigma}_{jk}$. This proves the first part of the lemma.

In the following, note that any positive semidefinite or positive definite matrix $\mathbf{A}$ can be expressed in the form $\mathbf{A} = \mathbf{M}\mathbf{M}^H$ through a Cholesky decomposition [43]. The transmit covariance matrices $\mathbf{\Sigma}_{jk}$ of the users, being of the form $\mathbf{\Sigma}_{jk} = E\left\{\mathbf{x}_{jk}\mathbf{x}_{jk}^H\right\}$, are positive semidefinite (as are their transposes), and hence a sum of those matrices would be such as well. Let $\mathbf{A} = \sum_{u \geq i} \mathbf{\Sigma}_{ju}^T$. Then,

$$\mathbf{H}_{ji}^{*}\left(\sum_{u\geq i}\boldsymbol{\Sigma}_{ju}^{T}\right)\mathbf{H}_{ji}^{T}=\mathbf{H}_{ji}^{*}\mathbf{A}\mathbf{H}_{ji}^{T}=\mathbf{H}_{ji}^{*}\mathbf{M}\mathbf{M}^{H}\mathbf{H}_{ji}^{T}=\left(\mathbf{H}_{ji}^{*}\mathbf{M}\right)\left(\mathbf{H}_{ji}^{*}\mathbf{M}\right)^{H} \tag{B.14}$$

is positive semidefinite. The addition of the identity matrix, which is positive definite, to the above will make the sum strictly positive definite. Furthermore, note that the inverse of a positive definite matrix is also positive definite [43]. Thus, letting $\mathbf{B}=\left[\mathbf{I}+\mathbf{H}_{ji}^{*}\left(\sum_{u\geq i}\boldsymbol{\Sigma}_{ju}^{T}\right)\mathbf{H}_{ji}^{T}\right]^{-1}$, and setting $k = i$, we have:

$$\begin{aligned}\frac{\partial R_{jk}}{\partial\boldsymbol{\Sigma}_{jk}}&=\mathbf{H}_{jk}^{T}\left[\mathbf{I}+\mathbf{H}_{jk}^{*}\left(\sum_{u\geq k}\boldsymbol{\Sigma}_{ju}^{T}\right)\mathbf{H}_{jk}^{T}\right]^{-1}\mathbf{H}_{jk}^{*}\\&=\mathbf{H}_{jk}^{T}\mathbf{B}\mathbf{H}_{jk}^{*}=\mathbf{H}_{jk}^{T}\mathbf{L}\mathbf{L}^{H}\mathbf{H}_{jk}^{*}=\left(\mathbf{H}_{jk}^{T}\mathbf{L}\right)\left(\mathbf{H}_{jk}^{T}\mathbf{L}\right)^{H}\end{aligned}. \tag{B.15}$$

Thus, $\partial R_{jk}/\partial\boldsymbol{\Sigma}_{jk}$, and in particular $\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}$, must be at least positive semidefinite. In the event $N_R \geq M_T$, and hence $\mathbf{H}_{jk}^{T}$ has more columns than rows, if $\mathbf{H}_{jk}$ is full rank, $\partial R_{jk}/\partial\boldsymbol{\Sigma}_{jk}$ will also have full rank and hence in fact be strictly positive definite. In any event, it is also clear that $\partial R_{jk}/\partial\boldsymbol{\Sigma}_{jk}$ can only be equal to $\mathbf{0}_{M_T\times M_T}$ if $\mathbf{H}_{jk}$ is an all-zero matrix, which occurs with probability zero in practice. This completes the proof of the lemma. ■

From (B.5), we know either μ_{jk} or $Tr\left(\boldsymbol{\Sigma}_{jk}\right)$ is zero, and from the problem constraints we know that $Tr\left(\boldsymbol{\Sigma}_{jk}\right)$ is zero if user k is not scheduled on subcarrier j, and greater than zero if it is. Hence, if $s_{jk} = 1$, $\mu_{jk} = 0$. From (B.4), for each j, either λ_j is zero, or $\lambda_j > 0$ and $\sum_{k\in\mathbb{S}} Tr\left(\boldsymbol{\Sigma}_{jk}\right)=P/N_C$. Let us now focus on $k = 1$. Then, (B.7) reduces to $\partial G/\partial\boldsymbol{\Sigma}_{j1} = \left(\partial G/\partial R_1\right)\left(\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}\right)$. From (B.6), since μ_{jk} is zero when $s_{jk} = 1$, if λ_j is zero, then there are two possibilities: (a) $\boldsymbol{\Omega}_{jk}^{T}=\mathbf{0}_{M_T\times M_T}$, and either $\partial G/\partial R_1 = 0$ or $\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}=\mathbf{0}_{M_T\times M_T}$; or (b) $\left(\partial G/\partial R_1\right)\left(\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}\right)+\boldsymbol{\Omega}_{j1}^{T}=\mathbf{0}_{M_T\times M_T}$. However, we know from the lemma that $\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}$ is non-zero, and $\partial G/\partial R_1 > 0$ by the constraint of equation (3.3) in Chapter 3. Thus, case (a) is impossible. Furthermore, since $\partial G/\partial R_1 > 0$, and since $\partial R_{j1}/\partial\boldsymbol{\Sigma}_{j1}$ and $\boldsymbol{\Omega}_{jk}^{T}$ are both positive semidefinite, case (b) is also impossible; it is impossible for two non-zero positive semidefinite matrices to add to an all-zero matrix. Thus, we must have

$\lambda_j > 0$ for each j. This can also be proven intuitively; if $\sum_{k\in\mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) < P/N_C$, it is always possible to increase the rate of the user encoded first by increasing its power until $\sum_{k\in\mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) = P/N_C$, without affecting the rates of the users encoded later in the order.

We lastly note that due to the structure of (B.8), it is only necessary to sum (B.7) over all $i \le k$. (B.7) can then be expressed more compactly. Hence, from the above facts and equations (B.4) to (B.8), we find the optimal transmit covariance matrices for a given set of users and encoding order satisfy the following set of equations:

$$\mathbf{\Sigma}_{jk} = \mathbf{0}_{M_T \times M_T}, \quad \text{if } s_{jk} = 0 ; \tag{B.16}$$

$$\sum_{k\in\mathbb{S}} Tr\left(\mathbf{\Sigma}_{jk}\right) = P/N_C, \quad j = 1,...,N_C ; \tag{B.17}$$

$$\mu_{jk} Tr\left(\mathbf{\Sigma}_{jk}\right) = 0 ; \tag{B.18}$$

$$\sum_{i\le k}\left(\frac{\partial G}{\partial R_i}\mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u\ge i}\mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^*\right) - \sum_{i<k}\left(\frac{\partial G}{\partial R_i}\mathbf{H}_{ji}^T\left[\mathbf{I} + \mathbf{H}_{ji}^*\left(\sum_{u>i}\mathbf{\Sigma}_{ju}^T\right)\mathbf{H}_{ji}^T\right]^{-1}\mathbf{H}_{ji}^*\right); \tag{B.19}$$
$$-\lambda_j s_{jk}\mathbf{I}_{M_T} - \mu_{jk}\mathbf{I}_{M_T} + \mathbf{\Omega}_{jk}^T = \mathbf{0}_{M_T\times M_T}, \quad \forall k \in \mathbb{S}$$

$$\mathbf{\Sigma}_{jk}, \mathbf{\Omega}_{jk} \succeq 0 . \tag{B.20}$$

Appendix C

Least Squares Polynomial Fit for GA Tuning

In Chapter 4, we find that there is a linear relationship between the log of the product KM_T and the log of a constant c for tuning the adaptive mutation rate parameters β_1 and β_2. This means we can express the relationship as $a \log(KM_T) + b = \log(c)$, for some constants a and b. We now proceed to find the least squares fit for the polynomial coefficients a and b [172]. The linear equations can be written in matrix form as follows:

$$\begin{bmatrix} \log(3.227) \\ \log(7.45) \\ \log(17.2) \end{bmatrix} = \begin{bmatrix} \log(20) & 1 \\ \log(40) & 1 \\ \log(80) & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{X} \begin{bmatrix} a \\ b \end{bmatrix}. \tag{C.1}$$

The least squares fit values for a and b can then be found by left-multiplying both sides of the above equation by the Moore-Penrose pseudoinverse [43] of $\mathbf{X}$:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{X}^{\dagger}\mathbf{X} \begin{bmatrix} a \\ b \end{bmatrix} = \left[\left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \right] \mathbf{X} \begin{bmatrix} a \\ b \end{bmatrix} = \left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \begin{bmatrix} \log(3.227) \\ \log(7.45) \\ \log(17.2) \end{bmatrix} = \begin{bmatrix} 1.2071 \\ -2.4445 \end{bmatrix}. \tag{C.2}$$

Thus, we have $1.2071 \log(KM_T) - 2.4445 = \log(c)$. We can further expand b as follows:

$$b = -2.4445 = -1.2071 \times 2.0252 = -1.2071 \log(7.5773). \tag{C.3}$$

Thus, $1.2071 [\log(KM_T) - \log(7.5773)] = \log(c)$. Then, it is simply a matter of rearranging the equation as $1.2071 \log(KM_T/7.5773) = \log[(KM_T/7.5773)^{1.2071}] = \log(c)$, or $c = (KM_T/7.5773)^{1.2071}$. Thus, we finally have:

$$\beta_1 + 0.15\beta_2 = (KM_T/7.5773)^{1.2071}. \tag{C.4}$$

Appendix D

Existing Method to Find Covariance Matrices for Successive Zero-Forcing

In [50], the authors propose a suboptimal method to find covariance matrices for successive zero-forcing (SZF) precoding that satisfy the required null space and sum-trace constraints. This method involves finding optimal covariance matrices for the users on the multiple access channel (MAC), transforming those matrices to the broadcast channel (BC), and finally projecting the BC covariance matrices to the null space and performing waterfilling for the user encoded last to obtain the SZF matrices. We elaborate on the procedure in the following.

D.1 MAC Waterfilling

To begin, optimal covariance matrices $\mathbf{P}_i$ for the multiple access channel must be found for the users. This could be done using any type of iterative waterfilling procedure to maximize the sum rate for the MAC, or even any procedure to maximize the weighted sum rate, if the weights are set to the same positive value for each user. The authors of [50] recommend the algorithm in [107], and it is this algorithm that we use in our work.

To maximize the MAC sum rate and obtain the optimal $\mathbf{P}_i$ for K_0 users, the algorithm in [107] uses the following steps at each iteration n:

1) For each user $i = 1,\ldots,K_0$, generate an effective channel matrix:

$$\mathbf{G}_i = \mathbf{H}_i \left(\mathbf{I} + \sum\nolimits_{j \neq i} \mathbf{H}_j^H \mathbf{P}_j \mathbf{H}_j \right)^{-1/2}, \tag{D.1}$$

where $\mathbf{H}_i$ is the *broadcast* channel matrix for user i; the channel matrix for the dual MAC is $\mathbf{H}_i^H$.

2) The effective channel matrices are treated as parallel, non-interfering channels. Covariance matrices $\mathbf{S}_i$ are found from the well-known waterfilling algorithm over the block-diagonal matrix formed from $blkdiag\left(\mathbf{G}_1,\cdots,\mathbf{G}_{K_0}\right)$, with power constraint P. First, the following SVD is computed for each i:

$$\mathbf{G}_i\mathbf{G}_i^H = \mathbf{U}_i\mathbf{D}_i\mathbf{V}_i^H . \quad \text{(D.2)}$$

Then, $\mathbf{S}_i = \mathbf{U}_i\mathbf{\Lambda}_i\mathbf{U}_i^H$, where $\mathbf{\Lambda}_i = \left(\mu\mathbf{I} - \left(\mathbf{D}_i\right)^{-1}\right)^+$, $(\mathbf{A})^+$ denotes component-wise the maximum of each component of $\mathbf{A}$ and 0, and μ is found such that[1] $\sum_{i=1}^{K_0} Tr\left(\mathbf{\Lambda}_i\right) = P$.

3) The covariance matrices are updated for the next iteration. If $K_0 = 2$, this can be done simply by $\mathbf{P}_i^{(n+1)} = \mathbf{S}_i$. However, this update may not converge in general; the sum rate may diverge from the optimal. To ensure convergence to the optimal sum rate for any K_0, the covariance matrices should be updated as follows:

$$\mathbf{P}_i^{(n+1)} = \frac{1}{K_0}\mathbf{S}_i + \frac{K_0 - 1}{K_0}\mathbf{P}_i^{(n)} . \quad \text{(D.3)}$$

The authors of [107] also recommend that the first few iterations of the algorithm could be updated by $\mathbf{P}_i^{(n+1)} = \mathbf{S}_i$ before switching to the update method of (D.3), to allow faster initial convergence. In our work, we use a slight modification of this recommendation. Either update method will always produce a set of covariance matrices each iteration that meets the sum-power constraint, provided that the initial starting point $\mathbf{P}_i^{(0)}$ used also meets that constraint. The algorithm will also monotonically approach the optimal sum rate, unless it diverges under the first update method. Since during the algorithm operation we monitor the sum rate at each iteration for convergence to a certain accuracy, we can thus also trivially monitor for a decrease in the sum rate. If one is found, only then do we switch the update method from $\mathbf{P}_i^{(n+1)} = \mathbf{S}_i$ to that in (D.3). This provides faster total convergence, and still ensures convergence to the optimal sum rate and covariance matrices $\mathbf{P}_i$.

D.2 MAC to BC Transformations

Once the optimal $\mathbf{P}_i$ for the MAC are found, they must be converted to optimal $\mathbf{\Sigma}_i$ for the BC. This is accomplished using the transformations proposed in [24], which we describe below.

Assume without loss of generality an encoding / decoding order $\pi(j) = j$, where user 1 in the order is assumed to be encoded *last* on the BC, and decoded *first* on the MAC. In this case, the rate for each user on the MAC will be:

[1] This assumes without loss of generality that the AWGN variance σ_n^2 is 1.

$$R_j^M = \log_2 \frac{\left|\mathbf{I} + \sum_{i=j}^{K_0} \mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i\right|}{\left|\mathbf{I} + \sum_{i=j+1}^{K_0} \mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i\right|}, \tag{D.4}$$

and the rate for each user on the BC will be:

$$R_j^B = \log_2 \frac{\left|\mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j} \mathbf{\Sigma}_i\right) \mathbf{H}_j^H\right|}{\left|\mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j-1} \mathbf{\Sigma}_i\right) \mathbf{H}_j^H\right|}. \tag{D.5}$$

First, define and calculate a set of matrices $\mathbf{B}_j$ as follows:

$$\mathbf{B}_j \triangleq \mathbf{I} + \sum_{i=j+1}^{K_0} \mathbf{H}_i^H \mathbf{P}_i \mathbf{H}_i \,. \tag{D.6}$$

These are the interference matrices for each user j on the MAC (i.e., the denominator of R_j^M). Then, define a set of matrices $\mathbf{A}_j$, which are the interference matrices for each user on the BC, as:

$$\mathbf{A}_j \triangleq \mathbf{I} + \mathbf{H}_j \left(\sum_{i=1}^{j-1} \mathbf{\Sigma}_i\right) \mathbf{H}_j^H \,. \tag{D.7}$$

Lastly, for each j, calculate the product $\mathbf{B}_j^{-1/2} \mathbf{H}_j^H \mathbf{A}_j^{-1/2}$, and denote its SVD as:

$$\mathbf{B}_j^{-1/2} \mathbf{H}_j^H \mathbf{A}_j^{-1/2} = \mathbf{F}_j \mathbf{D}_j \mathbf{G}_j^H \,. \tag{D.8}$$

The BC covariance matrices $\mathbf{\Sigma}_j$ and the matrices $\mathbf{A}_j$ can be calculated successively for increasing j, starting with $\mathbf{A}_1 = \mathbf{I}$. The transformation for $\mathbf{\Sigma}_j$ is given by:

$$\mathbf{\Sigma}_j = \mathbf{B}_j^{-1/2} \mathbf{F}_j \mathbf{G}_j^H \mathbf{A}_j^{1/2} \mathbf{P}_j \mathbf{A}_j^{1/2} \mathbf{G}_j \mathbf{F}_j^H \mathbf{B}_j^{-1/2} \,. \tag{D.9}$$

Using this transformation will yield a set of BC covariance matrices $\mathbf{\Sigma}_j$ with $\sum_j Tr\left(\mathbf{\Sigma}_j\right) \le \sum_j Tr\left(\mathbf{P}_j\right)$. Furthermore, the transformation will result in $R_j^M = R_j^B$ for all j.

The order chosen for the users will obviously affect the covariance matrices and the rates they obtain. The authors of [50] do not explicitly say what order to use for the transformation. However, the equations they use imply that if user $\pi(1)$ is encoded first for SZF, then user $\pi(1)$ should be encoded *last* for DPC on the BC. In this manner, the equation for the DPC user rates will be identical to the equation for the SZF user rates, apart from the additional null space constraints of SZF.

D.3 SZF Null Space Projections

With the optimal covariance matrices $\mathbf{\Sigma}_j$ for DPC obtained, the authors of [50] then project those matrices into the null spaces of users as required for SZF. Let us assume an

SZF encoding order $\pi(j) = j$, with user 1 encoded first. The projection then proceeds as follows:

1) Calculate the null space basis vectors $\bar{\mathbf{V}}_{j-1}^{0}$ for each user j (see Chapter 5).

2) For each user $j = 2$ to $K_0 - 1$, obtain the SZF covariance matrix $\mathbf{Q}_j$ as follows:

$$\mathbf{Q}_j = \bar{\mathbf{V}}_{j-1}^{0}\left(\bar{\mathbf{V}}_{j-1}^{0}\right)^H \boldsymbol{\Sigma}_j \bar{\mathbf{V}}_{j-1}^{0}\left(\bar{\mathbf{V}}_{j-1}^{0}\right)^H. \tag{D.10}$$

   We note that the projection is unnecessary for $j = 1$, since $\bar{\mathbf{V}}_0^0 \triangleq \mathbf{I}$. Thus, $\mathbf{Q}_1 = \boldsymbol{\Sigma}_1$.

3) For the final user $j = K_0$, find the effective channel $\mathbf{H}_{eff}$:

$$\mathbf{H}_{eff} = \left(\mathbf{I} + \mathbf{H}_{K_0}\left(\sum_{j=1}^{K_0-1}\mathbf{Q}_j\right)\mathbf{H}_{K_0}^H\right)^{-1/2} \mathbf{H}_{K_0}\bar{\mathbf{V}}_{K_0-1}^{0}\left(\bar{\mathbf{V}}_{K_0-1}^{0}\right)^H. \tag{D.11}$$

4) Obtain a temporary covariance matrix $\bar{\mathbf{Q}}_{K_0}$ by waterfilling over $\mathbf{H}_{eff}$ with the power constraint $P - \sum_{j=1}^{K_0-1} Tr\left(\mathbf{Q}_j\right)$.

5) Obtain the final covariance matrix $\mathbf{Q}_{K_0}$ by:

$$\mathbf{Q}_{K_0} = \bar{\mathbf{V}}_{K_0-1}^{0}\left(\bar{\mathbf{V}}_{K_0-1}^{0}\right)^H \bar{\mathbf{Q}}_{K_0}\bar{\mathbf{V}}_{K_0-1}^{0}\left(\bar{\mathbf{V}}_{K_0-1}^{0}\right)^H. \tag{D.12}$$

Although the SZF covariance method described in this appendix is suboptimal, it does perform quite well for $K_0 = 2$, and for higher K_0 at low to medium SNRs. However, as K_0 or the SNR grows larger, the conjugate gradient projection algorithm we propose in Chapter 6 can provide a significant improvement in the sum rate.

Appendix E

Supplemental Simulation Results for Scheduling Algorithms under Block Diagonalization and Successive Zero-Forcing

This appendix contains additional simulation results for the scheduling algorithms discussed in Chapter 5. The results here cover a wider range of signal-to-noise ratios (SNRs) than those presented in Chapter 5. We first present the results for BD, followed by the results for SZF. The graphs begin on the next page; the remainder of this page is intentionally left blank.

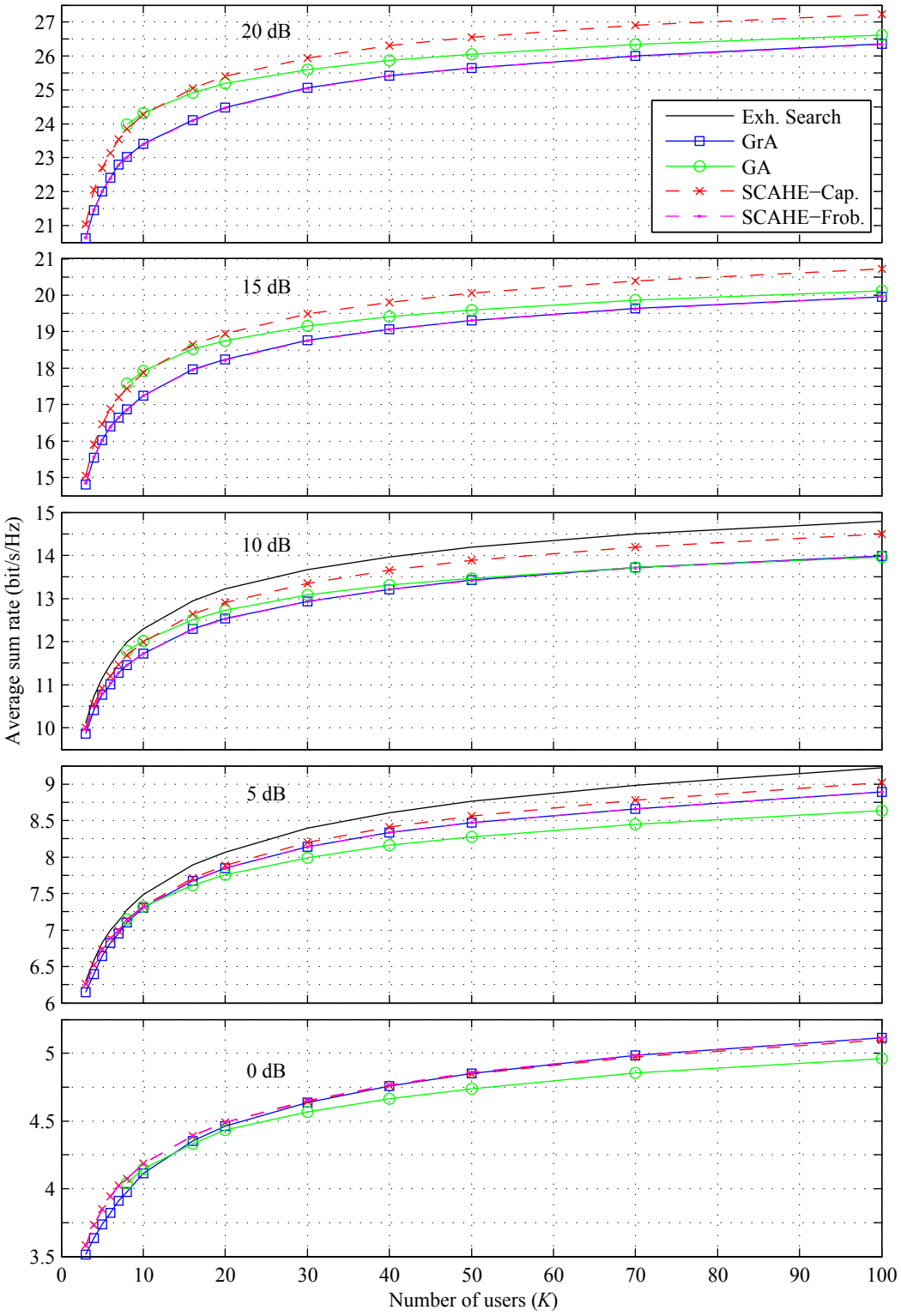

Figure E.1: Performance vs. K of exhaustive search, greedy (GrA), genetic (GA), and SCAHE [69] scheduling algorithms for BD and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$.

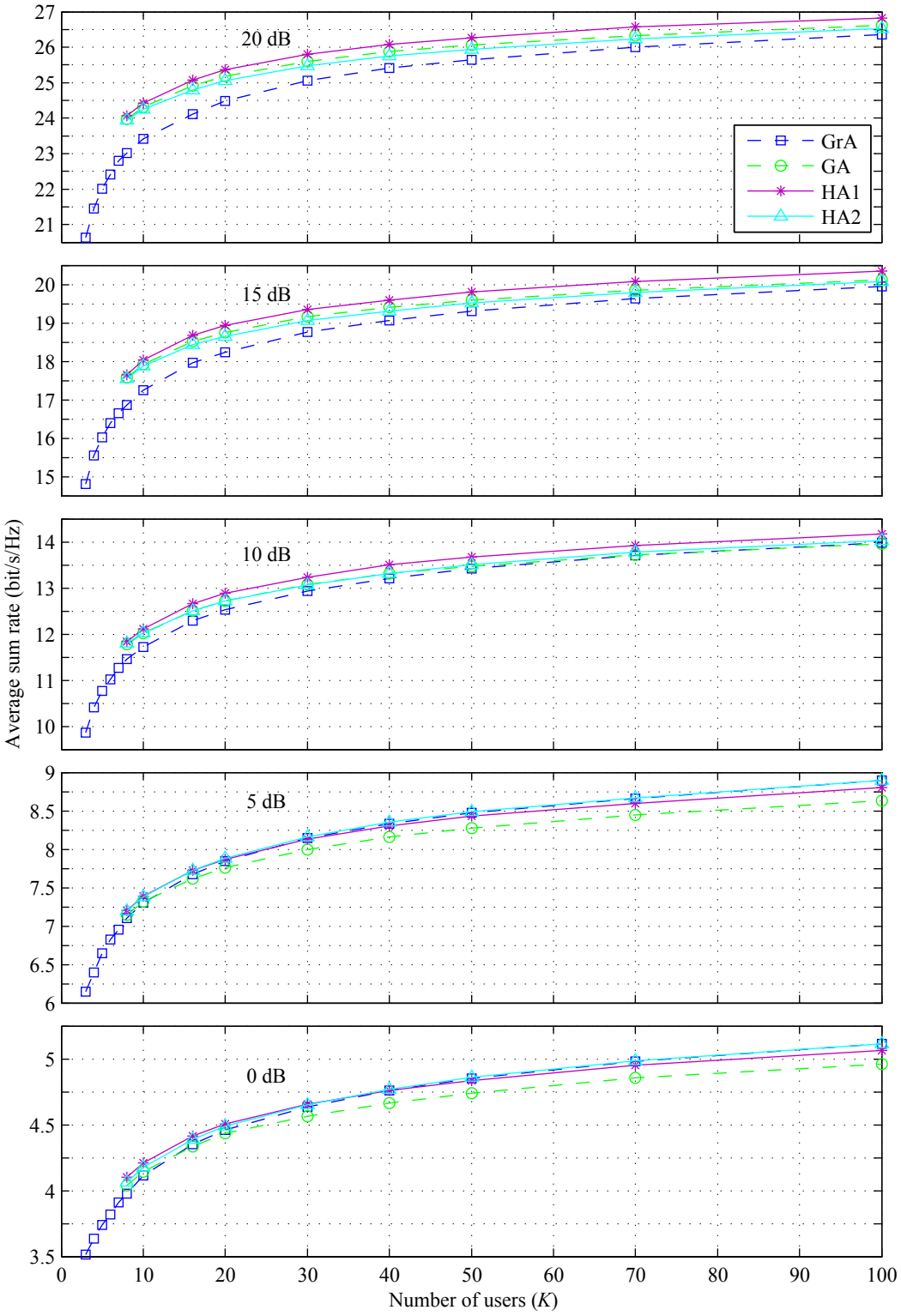

Figure E.2: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for BD and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$.

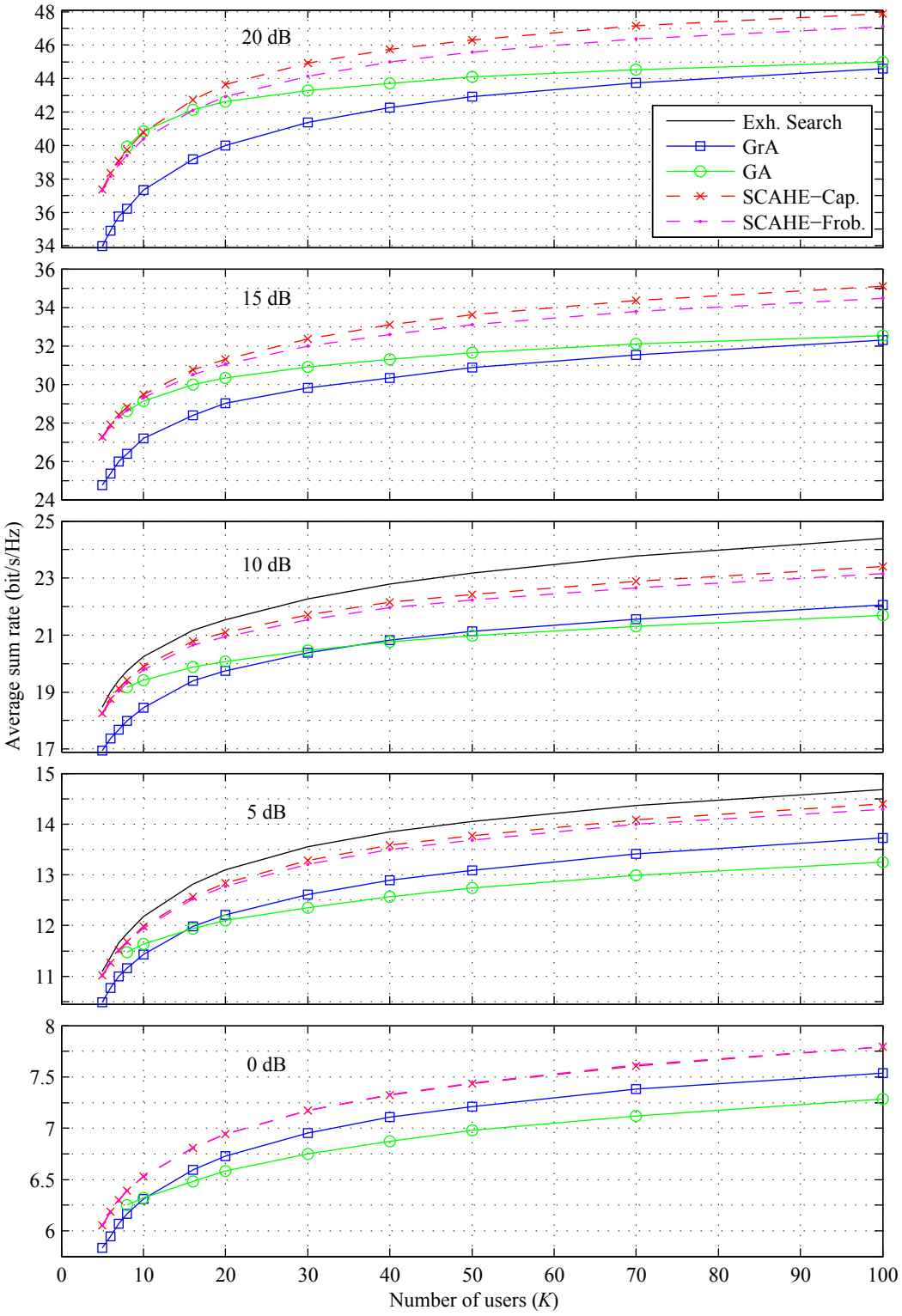

Figure E.3: Performance vs. K of exhaustive search, greedy (GrA), genetic (GA), and SCAHE [69] scheduling algorithms for BD and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$.

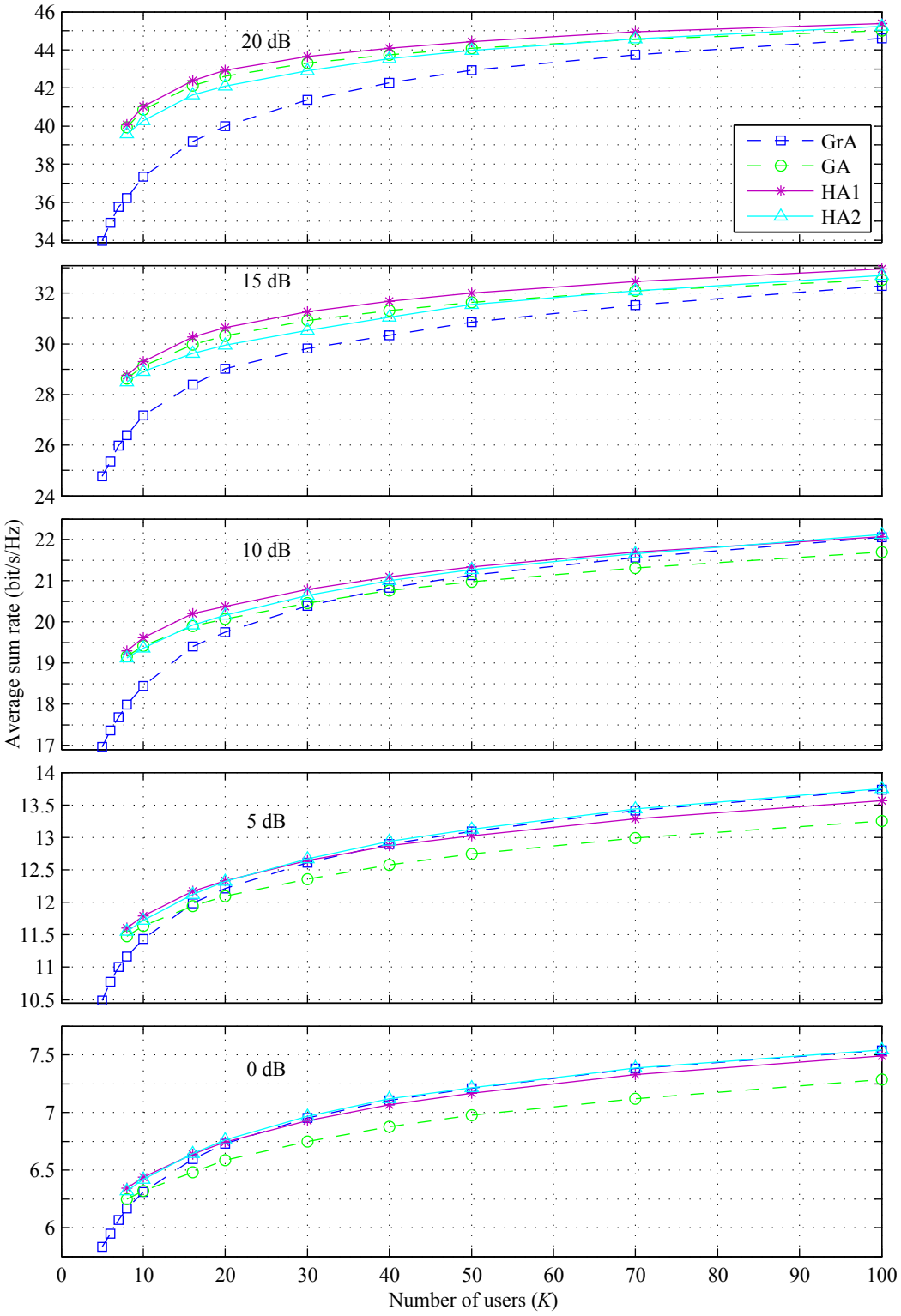

Figure E.4: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for BD and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$.

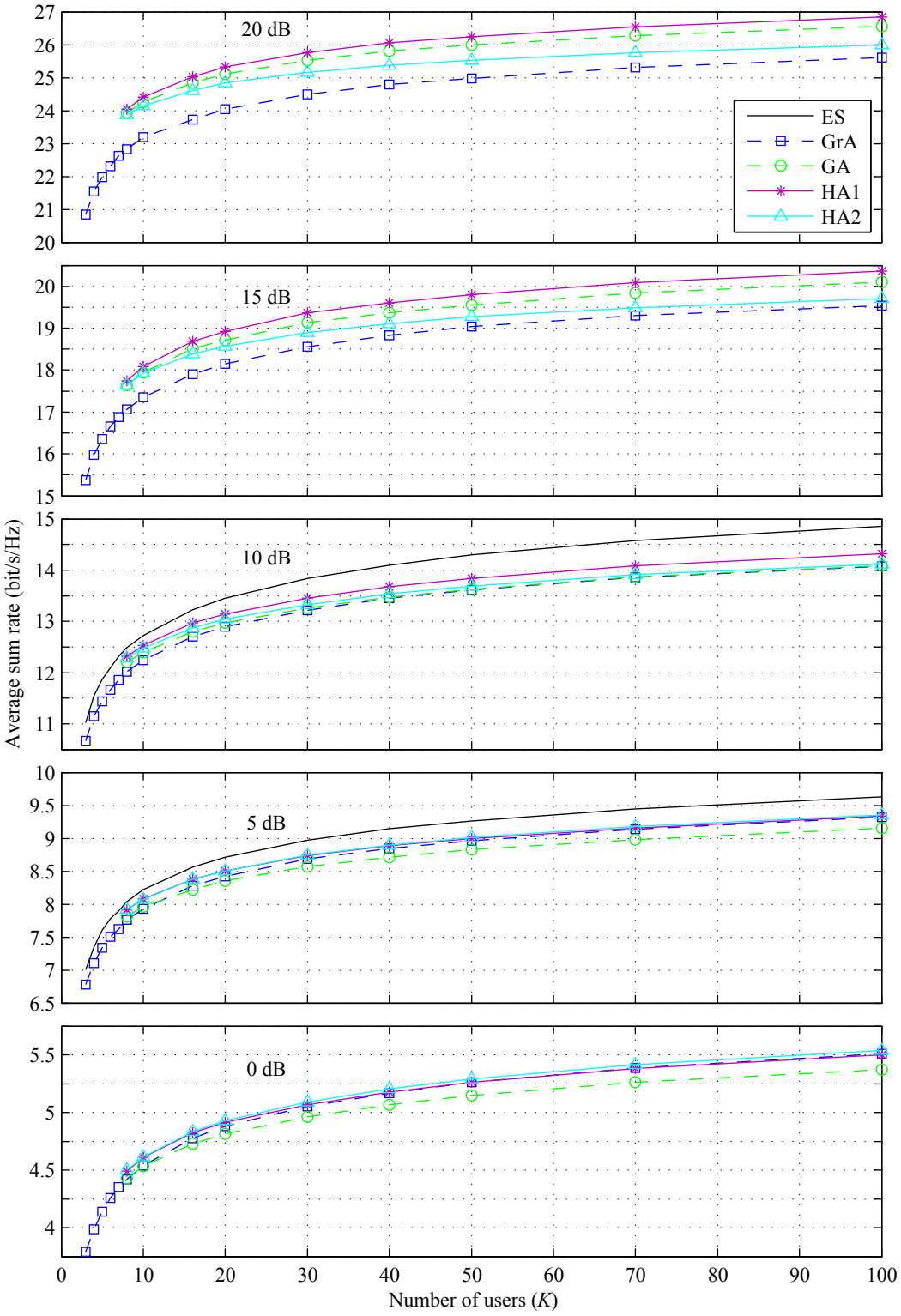

Figure E.5: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for SZF and various SNR; $M_T = 4$, $N_k = N = 2$, $K_0 = 2$.

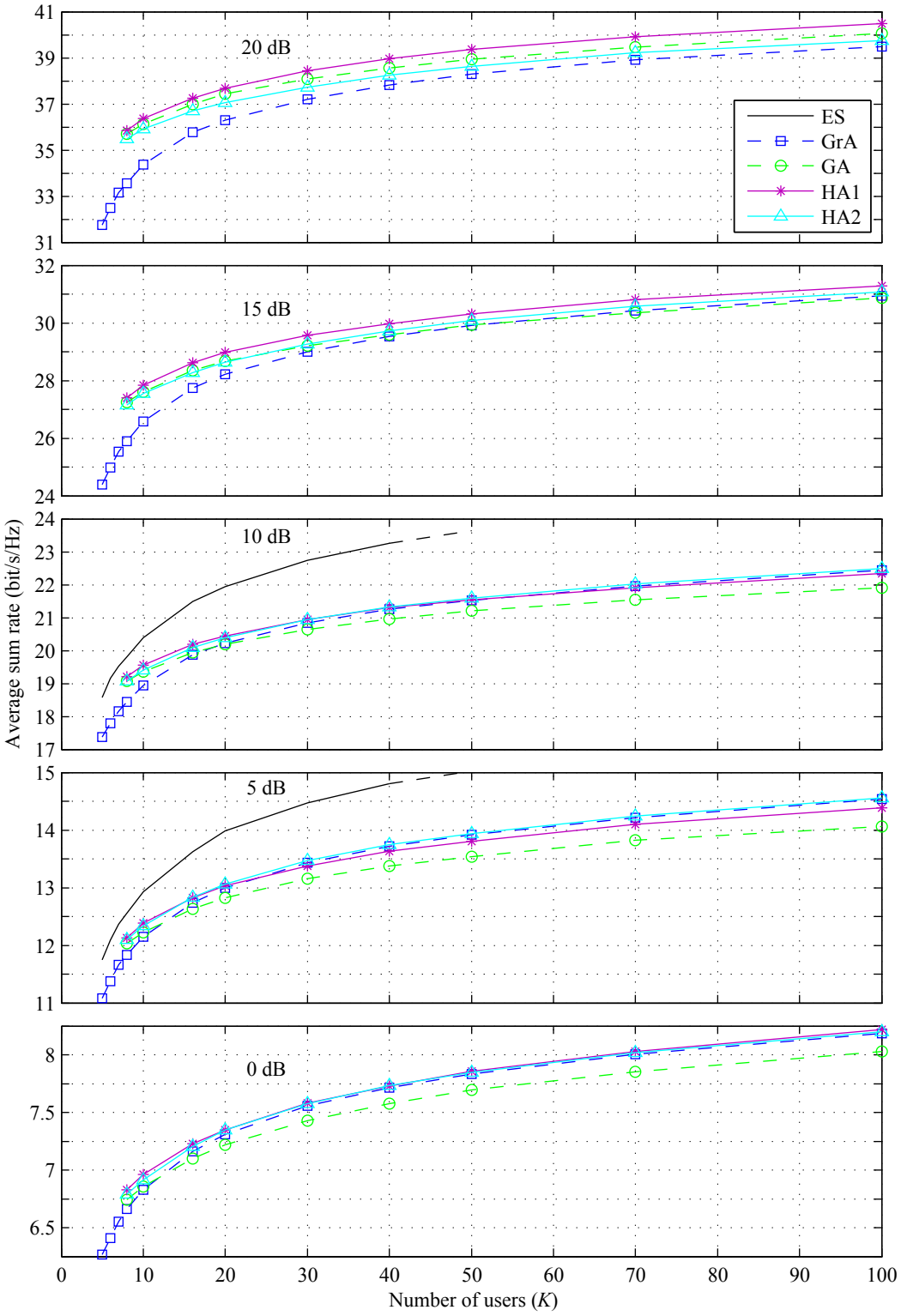

Figure E.6: Performance vs. K of greedy (GrA), genetic (GA), and hybrid scheduling algorithms 1 and 2 (HA1 and HA2) for SZF and various SNR; $M_T = 8$, $N_k = N = 2$, $K_0 = 4$.

Appendix F

Derivation of Gradient of SZF Weighted Sum Rate

Let us assume as in Chapter 6 an encoding order for SZF of $\pi(i) = i$, where user 1 is encoded first. The weighted sum rate of SZF is then $\sum_{i=1}^{K_0} w_i R_i$, where:

$$R_i = \log_2 \frac{\left|\mathbf{I} + \sum_{j=1}^{i} \mathbf{H}_i \mathbf{Q}_j \mathbf{H}_i^H\right|}{\left|\mathbf{I} + \sum_{k=1}^{i-1} \mathbf{H}_i \mathbf{Q}_j \mathbf{H}_i^H\right|} = \log_2 \frac{\left|\mathbf{I} + \sum_{j=1}^{i} \mathbf{H}_i \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{H}_i^H\right|}{\left|\mathbf{I} + \sum_{j=1}^{i-1} \mathbf{H}_i \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{H}_i^H\right|} \quad . \text{(F.1)}$$

$$= \log_2 \left|\mathbf{I} + \sum_{j=1}^{i} \mathbf{H}_i \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{H}_i^H\right| - \log_2 \left|\mathbf{I} + \sum_{j=1}^{i-1} \mathbf{H}_i \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{H}_i^H\right|$$

Note that the expression for R_i is very similar to that for DPC, thus, we can find the partial derivatives of R_i with respect to $\mathbf{Q}_k$ in much the same manner as we used in Appendix B. Let us define $\mathbf{Y}_j = \mathbf{T}_j \mathbf{T}_j^H$ and $\mathbf{X}_j = \mathbf{T}_j^H$. Furthermore, similar to as done in Appendix B, let us define $g(\mathbf{U}) = \log |\mathbf{U}|$, $\mathbf{A}_{0,ij} = \mathbf{H}_i \bar{\mathbf{V}}_{j-1}^0$, $\mathbf{A}_{1,ij} = \left(\bar{\mathbf{V}}_{j-1}^0\right)^H \mathbf{H}_i^H$, and $\mathbf{U} = \mathbf{I} + \sum_{j=1}^{i} \mathbf{H}_i \mathbf{V}_{j-1}^0 \mathbf{Y}_j \left(\mathbf{V}_{j-1}^0\right)^H \mathbf{H}_i^H$. Then, following equation (B.11) in Appendix B, for $k \le i$, we can calculate the derivative of the first half of R_i with respect to $\mathbf{Y}_k$ by:

$$\partial g = Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \partial \mathbf{Y}_k\right\} + Tr\left\{\mathbf{U}^{-1} \left(\sum_{j=1, j\neq k}^{i} \mathbf{A}_{0,ij} \partial \mathbf{Y}_j \mathbf{A}_{1,ij}\right)\right\}. \quad \text{(F.2)}$$

The second half of (F.2) does not contain $\partial \mathbf{Y}_k$, so we may ignore it. Thus, substituting for $\mathbf{Y}_k$, we get [162],[171]:

$$\begin{aligned} \partial g &= Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \partial \mathbf{Y}_k\right\} = Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \partial\left(\mathbf{X}_k^H \mathbf{X}_k\right)\right\} \\ &= Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \left[\left(\partial \mathbf{X}_k^H\right) \mathbf{X}_k + \mathbf{X}_k^H \partial \mathbf{X}_k\right]\right\} \quad . \quad \text{(F.3)} \\ &= Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \left(\partial \mathbf{X}_k^H\right) \mathbf{X}_k\right\} + Tr\left\{\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \mathbf{X}_k^H \partial \mathbf{X}_k\right\} \end{aligned}$$

Now, from [171], if a differential ∂g is a function of $Tr\left(\partial \mathbf{X}^*\right)$ (or of $Tr\left(\partial \mathbf{X}^H\right)$, since the trace of $\mathbf{X}^*$ equals the trace of $\mathbf{X}^H$), then $\partial g / \partial \mathbf{X}$ will evaluate to an all-zero

matrix. (Similarly, if ∂g is a function of $Tr(\partial \mathbf{X})$, then $\partial g / \partial \mathbf{X}^*$ is also all-zero.) Thus, we find:

$$\frac{\partial g}{\partial \mathbf{X}_k} = \mathbf{0} + \left[\mathbf{A}_{1,ik} \mathbf{U}^{-1} \mathbf{A}_{0,ik} \mathbf{X}_k^H \right]^T. \tag{F.4}$$

Substituting in for the temporary variables, for $k \le i$, we thus have:

$$\frac{\partial g}{\partial \mathbf{T}_k^H} = \left\{ \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum\nolimits_{j=1}^{i} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0 \right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k \right\}^T. \tag{F.5}$$

If $k > i$, then $\partial g / \partial \mathbf{T}_k^H$ is an all-zero matrix. The derivation for the second half of R_i is much the same, except that it is only non-zero for $k \le i - 1$.

Thus, we can now find the gradient of the weighted sum rate $R_{WSZF} = \sum_{i=1}^{K_0} w_i R_i$ with respect to user k as follows:

$$\nabla_k = 2 \frac{\partial R_{WSZF}}{\partial \mathbf{T}_k^*} = 2 \left[\frac{\partial R_{WSZF}}{\partial \mathbf{T}_k^H} \right]^T = 2 \sum\nolimits_{i=1}^{K_0} w_i \left[\frac{\partial R_i}{\partial \mathbf{T}_k^H} \right]^T. \tag{F.6}$$

As the first half of $\partial R_i / \partial \mathbf{T}_k^H$ is only non-zero for $i \ge k$, and the second half is only non-zero for $i \ge k + 1$, the bounds of the summation can be reduced. This yields:

$$\nabla_k = \frac{2}{\log 2} \left\{ \begin{array}{l} \sum\limits_{i=k}^{K_0} w_i \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum\nolimits_{j=1}^{i} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0 \right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k - \\ \sum\limits_{i=k+1}^{K_0} w_i \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum\nolimits_{j=1}^{i-1} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0 \right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k \end{array} \right\}. \tag{F.7}$$

The denominator of log 2 comes from the fact that R_i uses a base-2 logarithm, while the above derivations used a base-e (natural) logarithm. However, since our conjugate gradient projection algorithm normalizes the gradients, the leading constants can be left off. Then, pulling terms not involving i out of the summations, we finally find that the gradient for user k is proportional to:

$$\begin{aligned} \mathbf{G}_k = \left(\bar{\mathbf{V}}_{k-1}^0 \right)^H \Bigg(& \sum_{i=k}^{K_0} w_i \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum_{j=1}^{i} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0 \right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i - \\ & \sum_{i=k+1}^{K_0} w_i \mathbf{H}_i^H \left[\mathbf{I} + \mathbf{H}_i \left(\sum_{j=1}^{i-1} \bar{\mathbf{V}}_{j-1}^0 \mathbf{T}_j \mathbf{T}_j^H \left(\bar{\mathbf{V}}_{j-1}^0 \right)^H \right) \mathbf{H}_i^H \right]^{-1} \mathbf{H}_i \Bigg) \bar{\mathbf{V}}_{k-1}^0 \mathbf{T}_k \end{aligned}. \tag{F.8}$$

Appendix G

Supplemental Simulation Results for Scheduling Algorithms under SZF with Conjugate Gradient Projection Covariance Optimization Method

This appendix contains additional simulation results for the scheduling algorithms operating under successive zero-forcing (SZF), using the conjugate gradient projection (CGP) algorithm we propose in Chapter 6 to optimize the covariance matrices. The results here cover a wider range of signal-to-noise ratios (SNRs) than those presented in Chapter 6. We compare the performance of our proposed algorithm with the original method for the calculation of covariance matrices from [50]. For reference, we also show the performance of the scheduling algorithms when using block diagonalization (BD). The graphs are on the next page; the remainder of this page is intentionally left blank.

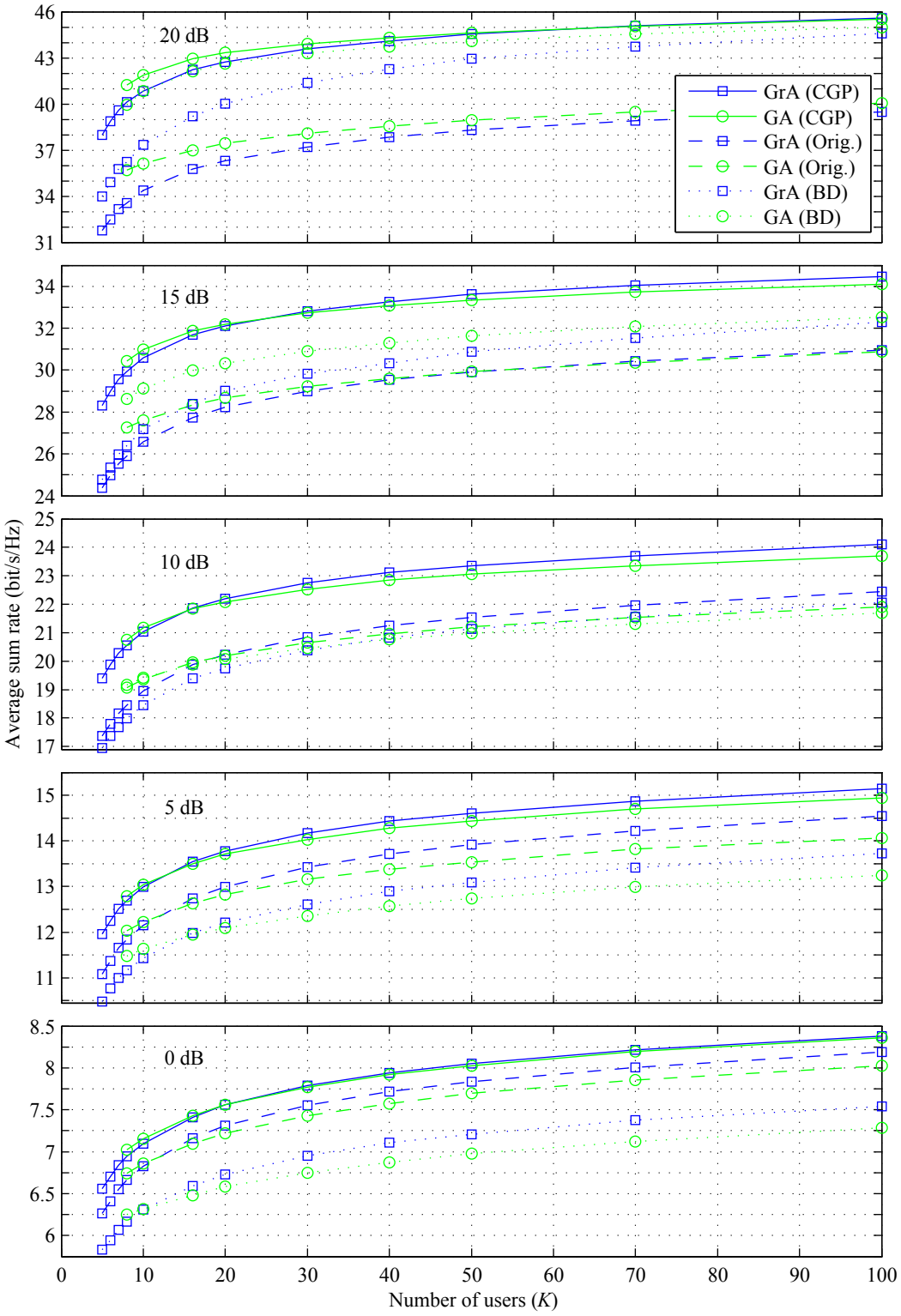

Figure G.1: Performance vs. K of greedy (GrA) and genetic (GA) scheduling algorithms for SZF, using proposed CGP (with "round-robin" initialization) and original methods to obtain covariance matrices. Performance of GrA and GA for BD also shown. $M_T = 8$, $N = 2$, $K_0 = 4$, various SNRs.