Design of Resilient Heterogeneous Wireless Networks

by

Ozgur Kabadurmus

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 3, 2013

Keywords: Survivability, Reliability, Heterogeneous wireless networks, Network design, Optimization

Approved by

Alice E. Smith, Chair, Professor of Industrial and Systems Engineering
Chase C. Murray, Assistant Professor of Industrial and Systems Engineering
Alvin S. Lim, Associate Professor of Computer Science and Software Engineering
Abdullah Konak, Associate Professor of Information Sciences and Technology

Abstract

With the growing use of new telecommunications technologies such as 4G and wireless hotspots, heterogeneous wireless networks (HetNets) are gaining more attention. The source of heterogeneity of a HetNet can either be the differences in nodes (such as transmission ranges, failure rates and energy levels) or the differences in services offered in the network (such as GSM and WiFi). Quality of service (QoS) is an issue for users while the cost of the infrastructure is an issue for the network provider. In telecommunications network design problems, survivability and reliability are well known QoS metrics. Most previous studies considered survivability and reliability as constraints (vertex connectivity or edge-disjoint paths), while other papers used traditional reliability metrics (such as two-terminal reliability or all-terminal reliability). In this dissertation, a new metric that combines network reliability with network resilience in capacitated networks is devised. Exact and approximate methods to evaluate this capacitated resilience metric are formulated and solved. Capacitated resilience is used to solve HetNets network designs ranging from 10 to 150 users. Results are compared to the popular reliability and survivability metrics in the literature. It is shown that networks designed by this new measure are significantly different than other network designs. This metric is the first to consider rerouting under capacity constraints in the instance of failure and thus reflects more realistic practice. It is also computationally tractable for use during network design.

Acknowledgments

I owe my deepest gratitude to my advisor, Dr. Alice E. Smith, for her encouragement, guidance, and support throughout my PhD process. Also, I would like to thank my committee members, Dr. Abdullah Konak, Dr. Chase C. Murray and Dr. Alvin Lim for their support. They gave me great suggestions and ideas throughout the dissertation process. I would also like to thank Dr. Jeffrey S. Smith for providing me research opportunities. I have learned a lot. I would also like to thank Dr. Bülent M. Durmuşoğlu, my former academic adviser from Istanbul Technical University, for encouraging me to start M.S. degree and seek career in academia.

Finally, but perhaps most importantly, this dissertation would have not been possible without the unquestioning support, sacrifice, and patience of my family and friends. I'd like to thank my dad (İsmail), my mom (Mürvet) and my brother (Tolga) for their love and encouragement. Graduate school is a long process, and it would have been even longer and lonelier without the support of Fatmanur Karaman. She was always there to motivate me whenever I needed it the most. I also would like to thank my friends Ramiz Boy, Gökhan Özden, Özgür Özmen, Eren Sakınç and Burcu Özkülhancı for their invaluable friendship.

Table of Contents

Abstract ii

Acknowledgments iii

List of Figures viii

List of Tables xv

List of Abbreviations xvii

List of Notation xix

1 Introduction 1

1.1 Wireless network types 2

1.2 HetNets 3

1.3 Problem definition 6

1.4 Primary contributions 11

2 Literature Review 14

2.1 Heterogeneous wireless networks 14

2.1.1 Nonsensor heterogeneous wireless networks 14

2.1.2 Heterogeneous wireless sensor networks 19

2.2 Survivability/reliability measures 26

2.2.1 Two-terminal reliability 26

2.2.2 Probability that two nodes can communicate 32

2.2.3 Expected loss of traffic (ELT) 33

2.2.4 Traffic efficiency 34

2.2.5 k-connectivity and edge-disjoint paths 36

2.2.6 Summary and discussion 36

2.3 Shortest path problem variations: Constraint shortest path and k shortest path 37

2.3.1 Summary and discussion 42
2.4 Some issues of wireless networks 44
2.4.1 Reliability of a wireless link and network density 44
2.5 Capacity of a wireless network and interference issues 47
3 Proposed metric: Capacitated resilience 51
3.1 The need for a new metric 51
3.2 Proposed metric: Capacitated resilience 52
3.2.1 Finding the most reliable path between the user to an access point . . 53
3.2.2 Identifying alternative paths between the user and any access point . 54
3.2.3 Determining independent subgroups of the alternative paths 57
3.2.4 Reliability calculation of independent subgroups 58
3.2.5 Calculating reliability of the alternative paths (resilience factor) using subgroup reliabilities 60
3.3 An example network 60
3.3.1 Capacitated resilience calculation 60
3.3.2 Connectivity measures 66
3.3.3 Two-terminal and all-terminal reliability 66
3.3.4 Traffic efficiency 66
3.3.5 ELT 66
3.3.6 Summary 67
4 Proposed Method and Solution Approach 68
4.1 Single objective Evolutionary Strategies (ES) model 69
4.1.1 Pseudocode of ES model 70
4.1.2 Encoding 71
4.1.3 Population initialization procedure 73
4.1.4 Edge selection algorithm 73
4.1.5 Calculation of cost, reliability and capacitated resilience 74

4.1.6 Mutation operators . . . 75
4.1.7 Replacement of population with children . . . 77
4.1.8 Stopping criteria . . . 77
4.1.9 Penalty functions . . . 77
4.2 Bi-objective Evolutionary Strategies (ES) model . . . 78
4.2.1 Definitions: "Non-dominated rank", "Crowding distance" and "Pareto optimality" . . . 78
4.2.2 Pseudocode of ES model . . . 81
4.2.3 Global Pareto front operations . . . 82
4.2.4 Multiobjective sorting . . . 84
4.3 Calculation of other survivability and reliability metrics . . . 86
4.3.1 k and all-terminal reliabilities . . . 86
4.3.2 Traffic efficiency (TE) . . . 87
4.3.3 k-connectivity . . . 89
5 Preliminary Results . . . 90
5.1 Single objective vs. bi-objective ES . . . 91
5.1.1 Pareto front diversity . . . 91
5.1.2 Comparison of non-dominated solutions for bi-objective and single objective Pareto fronts . . . 92
5.1.3 Single and bi-objective Pareto Fronts . . . 96
5.2 Equalizing computational effort . . . 97
5.2.1 Pareto front diversity . . . 97
5.2.2 Comparison of non-dominated solutions for bi-objective and single objective Pareto fronts . . . 101
5.3 Variation of single objective ES . . . 105
5.4 Summary . . . 105
6 Results . . . 110

6.1 Comparison of single (capacitated resilience) and bi-objective (cost and capacitated resilience) ES . . . 110
6.2 Correlation among capacitated resilience and other metrics . . . 111
6.3 Network structure: Comparison of capacitated resilience and other metrics . . . 118
6.3.1 A problem instance of the 10 user scenario . . . 119
6.3.2 Another problem instance of the 10 user scenario . . . 132
6.3.3 A problem instance of the 25 user scenario . . . 138
6.3.4 Summary of the differences of the network designs . . . 146
6.4 Bi-objective (TE and capacitated resilience) ES . . . 161
6.5 Solution time comparison of capacitated resilience and other metrics . . . 162
6.5.1 Solution time comparison of capacitated resilience: Effect of number of paths and cut-set size . . . 162
6.5.2 Solution time comparison of capacitated resilience: Problem size . . . 176
6.5.3 Solution time comparison of all metrics . . . 177
7 Conclusions and Further Research . . . 180
7.1 Conclusions . . . 180
7.2 Further research . . . 182
Appendices . . . 185
A Input data of problems . . . 186
A.1 The 10 user scenario . . . 186
A.2 The 25 user scenario . . . 188
A.3 The 50 user scenario . . . 191
A.4 The 75 user scenario . . . 196
A.5 The 100 user scenario . . . 204
A.6 The 150 user scenario . . . 214

List of Figures

1.1 A wireless mesh network design (from Amaldi et al. (2008)) 3

1.2 A HetNet design (from Yang et al. (2007)) 5

1.3 A HetNet design (from Capone et al. (2010)) 6

3.1 An example network 61

3.2 Independent subgroup 1 of the network in Figure 3.1 62

3.3 Independent subgroup 2 of the network in Figure 3.1 62

3.4 Independent subgroup 3 of the network in Figure 3.1 63

3.5 Capacitated version of the example in Figure 3.1 (c and f denote device capacity and user flow requirement, respectively 65

4.1 Crowding distance calculation 81

5.1 Comparison of Pareto front solutions (Problem instance 1) 93

5.2 Comparison of Pareto front solutions (Problem instance 2) 94

5.3 Comparison of Pareto front solutions (Problem instance 3) 95

5.4 Pareto fronts for problem instance 1: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds) 98

5.5 Pareto fronts for problem instance 2: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds) 99

5.6 Pareto fronts for problem instance 3: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds) 100

5.7 Comparison of Pareto front solutions (Problem instance 1) 101

5.8 Comparison of Pareto front solutions (Problem instance 2) 102

5.9 Comparison of Pareto front solutions (Problem instance 3) 103

5.10 Variation of the single objective ES for different budget values 106

5.11 Variation of the single objective ES for different budget values (5000 generations, 10 random number seeds) 107

6.1 Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 10 user scenario 112

6.2 Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 25 user scenario 113

6.3 Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 50 user scenario 114

6.4 Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 75 user scenario 115

6.5 Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 100 user scenario 116

6.6 Inputs of 10 users scenario (problem instance 1): User locations (x and y are the coordinates, f is the traffic flow requirement) 120

6.7 Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for Capacitated Resilience with unconstrained capacities 122

6.8 Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for Capacitated Resilience with unconstrained capacities 123

6.9 Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for Capacitated Resilience with constrained capacities . . 124

6.10 Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for Capacitated Resilience with constrained capacities . . 125

6.11 Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for TE 127

6.12 Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for TE 128

6.13 Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for two-terminal reliability 130

6.14 Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for two-terminal reliability 131

6.15 Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for all-terminal reliability 133

6.16 Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for all-terminal reliability 134

6.17 Summary of the designs of the problem instance 1 of the 10 user scenario, budget=500 135

6.18 Summary of the designs of the problem instance 1 of the 10 user scenario, budget=600 136

6.19 Inputs of the 10 user scenario (problem instance 2): User locations (x and y are the coordinates, f is the traffic flow requirement) 137

6.20 Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for Capacitated Resilience with unconstrained capacities 139

6.21 Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for Capacitated Resilience with constrained capacities . . 140

6.22 Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for TE 141

6.23 Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for two-terminal reliability 142

6.24 Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for all-terminal reliability 143

6.25 Summary of the designs of the problem instance 2 of the 10 user scenario, budget=600 144

6.26 Inputs of the 25 user scenario (problem instance 1): User locations (f is the traffic flow requirement) 145

6.27 Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for Capacitated Resilience with unconstrained capacities 147

6.28 Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for Capacitated Resilience with unconstrained capacities 148

6.29 Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for Capacitated Resilience with constrained capacities . . 149

6.30 Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for Capacitated Resilience with constrained capacities . . 150

6.31 Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for TE 151

6.32 Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for TE 152

6.33 Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for two-terminal reliability 153

6.34 Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for two-terminal reliability 154

6.35 Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for all-terminal reliability 155

6.36 Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for all-terminal reliability 156

6.37 Summary of the designs of the problem instance 1 of the 25 user scenario, budget=1000 157

6.38 Summary of the designs of the problem instance 1 of the 25 user scenario, budget=1200 158

6.39 Pareto fronts for the 10 user scenario problem instance 1 with budget constraint of 500: Bi-objective optimization of TE and capacitated resilience 163

6.40 Pareto fronts for the 10 user scenario problem instance 1 with budget constraint of 600: Bi-objective optimization of TE and capacitated resilience 164

6.41 Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (high TE and low capacitated resilience case) 165

6.42 Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (low TE and high capacitated resilience case) 166

6.43 Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (medium TE and medium capacitated resilience case) 167

6.44 Comparison of the designs presented in Figures 6.41 through 6.43 168

6.45 Solution time per iteration (one ES generation) comparison of single objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 170

6.46 Solution time per iteration (one ES generation) comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 171

6.47 Capacitated resilience comparison of single objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 172

6.48 Capacitated resilience comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 173

6.49 Number of evaluated alternative paths per iteration (one ES generation) comparison of single-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 174

6.50 Number of evaluated alternative paths per iteration (one ES generation) comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each) 175

List of Tables

2.1 Summary of objective functions used in the nonsensor heterogeneous wireless network literature . . . 15

2.2 Summary of decision variables used in the nonsensor heterogeneous wireless network literature . . . 16

2.3 Summary of parameters used in the nonsensor heterogeneous wireless network literature . . . 17

2.4 Summary of optimization methods used in the nonsensor heterogeneous wireless network literature . . . 17

2.5 Summary of survivability/reliability measures used in the nonsensor heterogeneous wireless network literature . . . 18

2.6 Summary of objective functions used in the heterogeneous wireless sensor network literature . . . 20

2.7 Summary of decision variables used in the heterogeneous wireless sensor network literature . . . 21

2.8 Summary of parameters used in the heterogeneous wireless sensor network literature 22

2.9 Summary of constraints used in the heterogeneous wireless sensor network literature 23

2.10 Summary of optimization methods used in the heterogeneous wireless sensor network literature . . . 23

2.11 Summary of survivability/reliability measures used in the heterogeneous wireless sensor network literature . . . 24

2.12 Comparison of reliability/survivability metrics . . . 37

4.1 Input parameters used in the ES model . . . 69

4.2 Decision variables used in the ES model . . . 70

4.3 Model parameters used in the ES model . . . 71

5.1 Performance of single and bi-objective ES for three problem instances 96

5.2 Comparison of Pareto solutions of single (1000 generations) and bi-objective (16,000 generations) ES for three problem instances 104

5.3 Mean and standard deviation of capacitated resilience for single and bi-objective runs (10 problem instances of 10 user scenario with 10 random number seeds for each problem instance. Capacitated resilience is calculated for number of alternative paths=10 and cut set size=4.) 109

6.1 Correlations calculated using the best solutions of 10 problem instances of the 10 user scenario 117

6.2 Summary of all metrics for the 10 user scenario, problem instance 1 132

6.3 Summary of all metrics for the 10 user scenario, problem instance 2 138

6.4 Summary of all metrics for the 25 user scenario, problem instance 1 159

6.5 Solution time comparison of single (capacitated resilience) and bi-objective (cost and capacitated resilience) ES for different problem sizes 176

6.6 Solution time comparison of all metrics for the 10 and the 25 user scenarios . . 178

6.7 The largest solvable scenarios in 24 hours, given as number of users for a medium budget 179

List of Abbreviations

AP	Access Point
BS	Base station
CPU	Central processing unit
CR	Capacitated resilience
ELT	Expected loss of traffic
ES	Evolutionary strategies
GA	Genetic algorithms
GPU	Graphics processing unit
HetNet	Heterogeneous wireless network
LAN	Local area network
MANET	Mobile ad hoc network
MAP	Mesh access point
MC	Mesh client
MG	Mesh gateway
MH	Mobile host, mobile handset
MR	Mesh relays
QoS	Quality of service

RN	Relay node
RP	Relay point
SN	Sensor node
TCP	Transport control protocol
TE	Traffic efficiency
WLAN	Wireless local area network
WMAN	Wireless metropolitan area network
WMN	Wireless mesh network
WSN	Wireless sensor network

List of Notation

G	Graph
V	Set of vertices (nodes)
E	Set of edges
n	Number of nodes
m	Number of edges
U	Set of users
r_j	Range of device j
c_j	Capacity of device j
f_i	Traffic requirement of user i
gridX and gridY	The limits of the area in which nodes are deployed
budgetLimit	Budget limit for installing devices
maxDevice	Maximum number of devices
N_p	Population size
N_c	Number of children
P_m	Probability to mutate
maxGen	Maximum number of generations
maxNonImprovingGen	Maximum number of non-improving generations
σ	The mutation rate for coordinates
successRate	A parameter to adjust σ in ES
sigmaAdjuster	Adjustment rate of σ in ES
penaltyUnassignedUser	Penalty value if a user is not assigned to a device
penaltyInfeasibleRouting	Penalty value if a device cannot connect to the wired backbone

Chapter 1

Introduction

Reliability and survivability of telecommunication networks is a popular area of optimization. With the growing use of wireless services, e.g. 3G/4G and wireless hotspots, the topic has assumed more importance. The needs of users are increasing as the Internet is becoming the new source of entertainment. Bandwidth requirements of users have increased dramatically with the available multimedia services and even more bandwidth will be demanded in the future due to the increasing number of communication devices (such as cell phones, laptops and tablets). Connection speed, coverage, session continuity and reliability are the main concerns of the users. The competitive structure of the telecommunication industry forces service providers to invest more in their infrastructure to satisfy user demands. With the emergence of new technologies, service providers try to combine different solutions to serve customers better. This creates the network design problem.

In the literature, most of the efforts were to evaluate the reliability or the survivability of a network. However, design of the network is very important for service quality. In general, a telecommunication network design problem is to minimize the cost of a network while ensuring some quality of service constraints. The network can be any type, such as wireless or fiber optic. In this dissertation, wireless telecommunication networks are considered. In a typical wireless communication network, users connect to an intermediate node; then intermediate nodes connect to an end node (e.g., a base station - usually connected to a wired backbone). While "session continuity" is a concern for users, the cost of the infrastructure is an issue for the network provider. Although networks vary, the requirement is always the same: coverage and reliability/survivability at a low cost. Network type does not affect this requirement but the related properties, constraints and assumptions vary.

This dissertation defines and investigates "Capacitated Resilience" (CR) in heterogeneous wireless networks (HetNets). The structure is mesh type networks, but the methodology presented in this dissertation can be easily extended to other wireless networks (such as sensor networks). Capacitated resilience, a new metric proposed in this research, is related to reliability and it will be explained in detail in Section 3. Before defining capacitated resilience, the basic wireless network types (mesh, sensor and ad hoc networks) and HetNets are summarized.

1.1 Wireless network types

A wireless mesh network (WMN) consists of "interconnected mesh access points (MAPs), relays (MRs) and gateways (MGs) in which mesh clients (MCs) connect to MAPs to access the Internet and MGs act as bridges between the wireless infrastructure and the Internet while MRs relay the traffic" (Benyamina et al., 2011). According to Amaldi et al. (2008), MRs and MAPs are often fixed and electrically powered. Also, MAPs are connected to a wired link, such as local area network (LAN), DSL or fiber. Wireless network devices in a mesh network have an organizational hierarchy (Amaldi et al., 2008). An illustration of the structure of a mesh network is given in Figure 1.1. In this structure, users connect to a MR and MRs connect to a MAP which is connected to the wired backbone.

A wireless sensor network (WSN) consists of "low-cost, low-power and energy-constrained sensors to monitor a physical phenomenon" (Guidoni et al., 2010). Sensors are aimed to monitor an area and gather information, however, they have limited ranges and they are usually battery operated with limited energy levels. Generally, relay nodes (RNs) are used to transmit information from sensor nodes (SNs) to a server node (or an end node).

Another type of wireless networks is "ad hoc networks". According to Moraes et al. (2009), an ad hoc network has transceivers that can receive and transmit packets (information) by using multiple consecutive wireless links. Mobile ad hoc networks (MANETs) are wireless networks in which the nodes (transceivers) are mobile. Unlike mesh type networks,

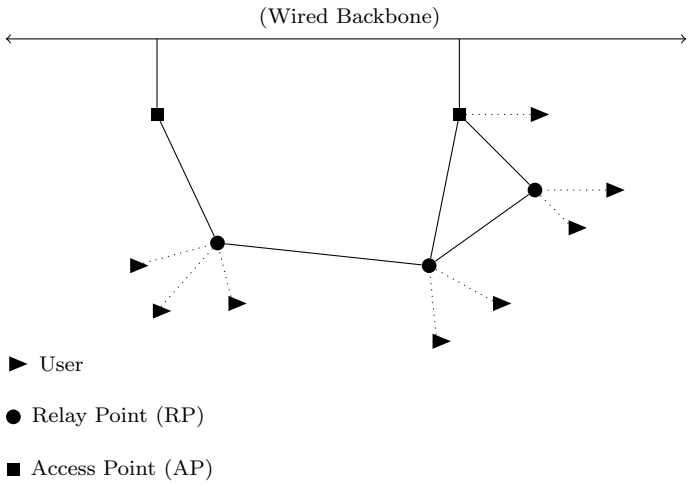


Figure 1.1: A wireless mesh network design (from Amaldi et al. (2008))

where the network has a hierarchy, MANETs are self configuring networks without a fixed infrastructure.

1.2 HetNets

HetNets are an integration of different kinds of wireless networks. Chen et al. (2007) state that telecommunication service providers, such as Verizon, Sprint and T-Mobile, are integrating or planning to integrate multiple wireless technologies with partially overlapped coverage areas. One example is to offer wireless LAN access for their 3G/4G customers. However, to benefit from these services, the users (mobile hosts) must be equipped with one or more wireless access technologies. In this case, a mobile host can choose WiFi in one location and 3G/4G in another location because of different cost rates, bandwidth or coverage properties. Because customer satisfaction is closely related to the quality of service (QoS) level and better service usually means more investment, a service provider should optimize cost by employing the best combination of available heterogeneous wireless technologies (Chen et al., 2007). Niyato and Hossain (2009) claim that the future of wireless networks is to provide seamless mobility to users with the integration of different wireless access technologies. For example, they foresee the integration of 802.16 based metropolitan area

networks (WMANs) and 802.11 based wireless local area networks (WLANs) in the near future. This integration is an example of a HetNet. They cite load balancing and preventing network congestion as important issues of this integration.

Pei et al. (2010) state that research on integration of third-generation (3G) mobile systems (and, currently, 4G service) and WLAN systems are gaining popularity. According to the authors, the main reason for this popularity is to satisfy more diversified QoS needs of users. Although 3G systems can provide more bandwidth and economic revenues, WLAN technologies are widely used and demanded by customers. The important issue is to select an appropriate wireless access technology to have service continuity (Pei et al., 2010).

Yang et al. (2007) give an example of HetNets in Figure 1.2. In this example, the authors introduce ad hoc network technology on top of a cellular system to increase system performance. If a request from mobile handset 2 (MH2) is blocked due to limited bandwidth on base station 1 (BS1), traffic diversion station 2 (TDS2) reroutes the flow to BS2. The new path becomes MH2-TDS2-TDS1-BS2. In this type of system, the traffic diversion station can connect either base stations or mobile handset devices. In other words, the traffic diversion stations and mobile handset devices can use both ad hoc technologies and cellular network technologies. Yang et al. (2007) refer to the IEEE 802.11 protocol for their ad hoc interface. They also mention that all base stations in their example (Figure 1.2) operate at a cellular network frequency.

Yang et al. (2007) assert that destination nodes are unknown and this is the main difference between HetNets and wireless mobile ad hoc networks in which the destination nodes are known. In HetNets the destination can be any node, as long as the bandwidth requirements (or other QoS requirements) are met. Hence, this involves a unique issue for HetNets - the destination selection problem. In this problem, the best suitable base station must be selected to satisfy QoS requirements. However, many important issues must be addressed to solve it. In HetNets, users can choose from variety of wireless access technologies (such as, a cellular network or a WLAN) with different channel characteristics (such as bandwidth,

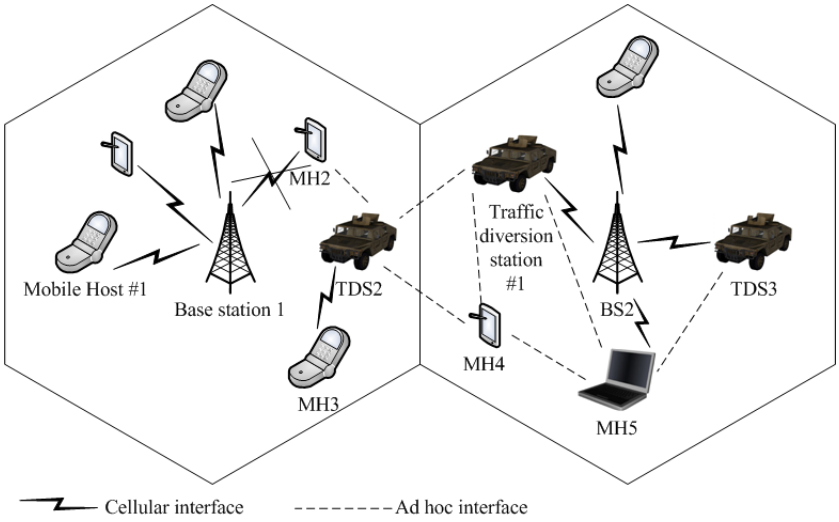


Figure 1.2: A HetNet design (from Yang et al. (2007))

loss or delay). Another issue is that requirements differ due to the different service providers that users interact with, because each service provider offers different services. Yang et al. (2007) state that this routing problem is NP-hard due to the heterogeneity of the system.

Another example of HetNets is given in Figure 1.3. In this design, a sink node gathers all of the information sent from sensor nodes. It is basically a heterogeneous wireless sensor network. Gateways are added to the system to balance load and increase network life time (because sensors are battery powered). Since gateway devices are more expensive than sensors, the objective is to minimize the number of gateway devices without decreasing the QoS level (different QoS metrics are covered 2.2). In this example, the sink node is connected to a direct geographical link, such as a wired backbone or a satellite connection. However, instead of connecting the gateways with these expensive technologies, they are connected to each other with wireless links such as ZigBee, WiFi or WiMax. The authors state that the connection of gateways constitutes a wireless mesh network and sensors are connected to this mesh backbone. Gateways can be fixed or mobile and finding the best location of gateways is an issue. In this example, sensor to gateway and gateway to gateway connections may use different wireless technologies. The structure of this example by Capone et al. (2010) is very similar to the one of Yang et al. (2007).

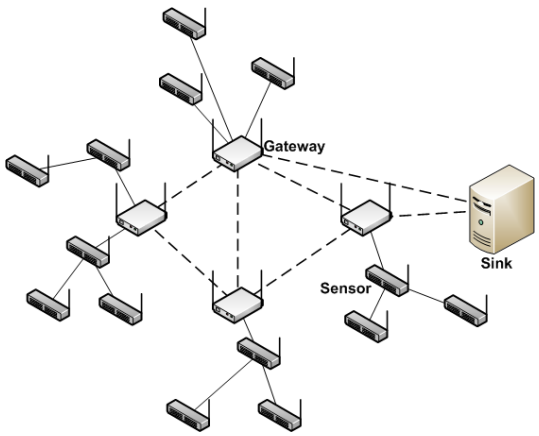


Figure 1.3: A HetNet design (from Capone et al. (2010))

1.3 Problem definition

A telecommunications network can be defined as a graph G consisting of vertices (nodes) of users and network devices. Edges are the connections between users (or devices) and network devices. To be connected with a network device v, a user or network device must be within the communication range of v. The communication range is determined by the technical specification of the network device. Throughout this dissertation, the sets of vertices (devices or users) and edges (wireless or wired links) of G will be denoted as V and E, respectively.

Design of survivable/reliable heterogeneous wireless networks is a new area of optimization which has applications in mesh and sensor networks. With the growing use of new telecommunications technologies such as 4G and wireless hotspots, this subject is gaining more attention. The source of heterogeneity of a HetNet is the difference in services offered in the network (such as 3G/4G and WiFi). Some authors in the literature also use the term heterogeneity as the differences in nodes (such as transmission ranges, failure rates and energy levels). In this dissertation, the research problem includes both different nodes and different services in the wireless network. The network structure will be similar to the ones in Figures 1.2 and 1.3. In this type of network, users connect to a gateway (relay node) and gateways connect to a sink node (e.g., base station) using potentially different wireless

technologies. In a study for homogeneous wireless networks (Amaldi et al., 2008), a similar structure for mesh networks was presented and the network design problem was solved to minimize the installation cost of the network under full coverage constraints. However, the heterogeneous wireless network presented by Capone et al. (2010) integrates different wireless technologies and different nodes (with different properties), and it will be used in this dissertation as a base structure of the proposed model.

The main focus of this dissertation is the resilience of a telecommunications network. Capacitated resilience is related to reliability and survivability; however, it has not yet been considered in survivable networks. Survivability/reliability of heterogeneous wireless networks has gained attention recently and the two important network designs are mesh and sensor type networks. Most studies consider survivability/reliability as a constraint (vertex connectivity or edge-disjoint paths). For example, Kashyap et al. (2010) minimized the number of relay nodes in a sensor network under k-connectivity constraints. Benyamina et al. (2011) worked on a reliable mesh network design problem with k-connectivity constraints. The remaining papers on optimization of heterogeneous wireless networks do not consider survivability/reliability. Among them, Amaldi et al. (2008) proposed a mathematical model to minimize the cost of a mesh network without considering survivability/reliability. Benyamina et al. (2009b) worked on the same problem proposed by Amaldi et al. (2008); however, they used bi-objective optimization (minimizing cost and maximizing network throughput) without survivability/reliability constraints. In another similar work, Benyamina et al. (2009a) proposed a bi-objective algorithm for the minimum gateway placement problem in mesh networks (without survivability/reliability) to minimize cost and congestion of gateways. Benyamina et al. (2009b) and Benyamina et al. (2009a) only considered different nodes in the networks as the source of heterogeneity. In this dissertation, the integration of different wireless technologies are also considered. More importantly, link capacity for rerouting is included in the context of survivability/reliability.

In studies using traditional reliability calculations, cost is minimized under a minimum reliability constraint. For example, Dengiz et al. (2002) used Formulation 1.1 where c_{ij} is the cost to use edge x_{ij}. In this formulation, "all-terminal reliability" of a solution is required to exceed a predefined threshold.

$$\begin{aligned} &min\ z = \sum_{i=1}^{N-1} \sum_{i=1+1}^{N} c_{ij} x_{ij} \\ &s.t. \\ &R(x) \geq R_0 \end{aligned} \tag{1.1}$$

The calculation of all terminal reliability is simply the multiplication of the reliable components. To demonstrate, a simple formulation from Grover (2004) is shown in Equation 1.2. As two-terminal reliability includes only two nodes in the network, all terminal reliability includes all node pairs in the network.

$$\begin{aligned} &R(G, s, t, p) = \sum_{i=0}^{m} N_i(G, s, t) p^i (1-p)^{m-i} \\ &\text{where } \{s, t\} \in E,\ p = \text{fixed failure rate of edges}, \\ &N_i(G, s, t) \text{ is the number of operating states} \end{aligned} \tag{1.2}$$

This reliability calculation is nonlinear due to the multiplication of decision variables x_{ij} (1 if edge (i, j) is selected). However, it can be linearized by using the logarithm of reliability as explained in Equation 3.4 in Section 3.2.1. In this dissertation, "capacitated resilience" metric is used instead of reliability because capacitated resilience includes reliability and also considers rerouting of flows when a failure occurs. Rerouting is very important to ensure survivability of the network. The motivation to use capacitated resilience instead of reliability is explained in detail in Section 1.4.

In this dissertation, a telecommunciation network consists of users and devices; and capacitated resilience is defined at the user level. "User" can represent an individual or

total traffic requirements in an area. Equation 1.3 shows the capacitated resilience of User i (U_i). $R(p_i)$ denotes the reliability of the assigned path of User i where the user is assigned to a device whose path has maximum reliability (from the user to an AP). The resilience factor scales the user reliability. To calculate it, all alternative paths from User i to any available access point are identified first. Then, reliabilities of the alternative paths are calculated. These steps are explained in detail in Section 4.1.4. Capacity of devices and links are considered for finding initially assigned and alternative paths.

$$Capacitated\ resilience\ = R(p_i) * Resilience\ Factor \tag{1.3}$$

After finding the user capacitated resiliences (which is computationally the most expensive part of the capacitated resilience calculation), network level capacitated resilience is calculated easily according to Equation 1.4. Network capacitated resilience calculation is simply the weighted average of user level capacitated resiliences in terms of user traffic requirements. In this equation, w_i is the weight of User i, which is calculated as flow of User i / total flow of users. Both network and user level capacitated resiliences are between 0 and 1 because network level capacitated resilience is the weighted average of user level capacitated resiliences which are found by scaling the reliability of the assigned paths ($R(p_i)$, where $0 \leq R(p_i) \leq 1$).

$$Capacitated\ resilience = w_i * Capacitated\ resilience\ (U_i) \tag{1.4}$$

Similar to a reliability calculation, the capacitated resilience calculation is nonlinear. Nevertheless, this calculation can be handled with metaheuristic search methods, such as Evolutionary Strategies (ES) and genetic algorithms (GA), because of the flexibility of those methods. For heuristic search methods, an important issue is to use penalty functions to accommodate the capacitated resilience constraint in the objective function to eliminate

infeasibility issues. In this dissertation, new aspects of survivability/reliability (with capacitated resilience) are investigated. A practical yet effective solution methodology has been developed to minimize the cost (maximize capacitated resilience) of a network for a required capacitated resilience level (budget). The details of the solution method are covered in Chapter 4.

Typical objective functions used in previous studies are cost (Shahnaz and Erlebach, 2010), energy consumption (Moraes et al., 2009) and lifetime of the network (Yang et al., 2009). In general, the objective functions of survivable/reliable networks are either cost or reliability. In models where the objective function is cost, reliability/survivability is considered as a constraint (either edge-disjoint paths or k-connectivity). If reliability is the objective, then cost is usually constrained to an upper bound.

In addition to single objective optimization, two objective functions are optimized simultaneously in this dissertation: cost and capacitated resilience. Pareto optimality is used as the bi-objective optimization method. Benyamina et al. (2009a) and Benyamina et al. (2009b) used bi-objective optimization for mesh networks, but they did not consider survivability/reliability. Capone et al. (2010) minimized cost and energy consumption, but they used a simple weighted additive objective function and they did not consider survivability/reliability. Therefore, this research fills a gap by using cost and a survivability/reliability metric that considers edge and node capacities in bi-objective optimization of heterogeneous wireless networks.

In this dissertation, the decision variables are the number, types and locations of the devices, user to device assignments, and routing decisions from users to APs (see Section 4.1 for more details). In addition, to distinguish this research from others, realistic features are included in the optimization model. These include variable transmission ranges of devices and the presence of relays in the network. Although the problem becomes more complicated with the addition of different components, the solution becomes more meaningful for real life applications.

1.4 Primary contributions

The primary objectives of this dissertation are to present a new and valuable survivability metric, capacitated resilience, for wireless heterogeneous networks and to develop effective methods for solving realistic heterogeneous wireless network design problems where the objective functions are cost and capacitated resilience. Models with one objective function (cost or capacitated resilience) have been developed to present a solution approach. As there are two metrics to optimize, Pareto optimality is also used in a biobjective model. Using penalty functions for capacitated resilience helps the heuristic optimization method move from the infeasible region to the feasible region easily (see Section 4.1.9) during search. Realistic size problems instances with many (more than 100) users and devices are solved to assess the proposed survivability metric.

The main hypothesis of this research is that a network design with better allocation of redundancies for rerouting options considering node capacities can be obtained by using capacitated resilience instead of using traditional reliability metrics or k-connectivity/edge disjoint paths constraints. Network designs using capacitated resilience, traditional reliability constraints and connectivity constraints are compared to verify this hypothesis in this dissertation (Section 6.3). Capacitated resilience enables more realistic operation because it considers rerouting possibilities in case of a failure. Traditional reliability measures (such as two-terminal/all-terminal reliability, traffic efficiency, probability that two nodes can communicate and expected loss of traffic), do not directly consider rerouting options. Survivability constraints, such as k-connectivity and edge disjoint paths, consider rerouting options; however, they allocate redundant paths to the network without consideration of reliability or capacity. They assume the components (nodes or edges) are perfectly reliable. The redundancy provided by k-connectivity and edge disjoint paths increases the chance of a user remaining connected to the network but it causes more slack capacity and higher costs. On the other hand, capacitated resilience also includes redundancy by considering rerouting

and leads to designs with less, but more effectively distributed, slack capacities. More on this discussion is presented in Section 2.2.6.

This research provides contributions to the survivable/reliable wireless telecommunication networks field in several ways:

- First, capacitated resilience as a survivability/reliability measure is the main contribution of this dissertation. As previously explained, in the survivable/reliable wireless networks literature, survivability/reliability was often considered as a constraint of edge-disjoint paths (or k-connectivity). Some studies used traditional reliability measures, e.g. two-terminal reliability or all-terminal reliability, as objective functions. Survivability/reliability has been similarly studied in the heterogeneous wireless networks literature. However, wireless networks (heterogeneous or homogeneous) require a better design approach considering rerouting of flows (in case of a failure). Therefore, a different metric other than reliability is needed. Capacitated resilience is devised because it includes both reliability and rerouting alternatives. The capacitated resilience metric is applicable not only to telecommunications networks, but also to other network types, such as transportation networks. It can also be used in military applications where instantaneous changes can affect the success of an operation. Both exact calculation and estimation methods are presented for capacitated resilience calculation.

- A second contribution is to present a bi-objective method to optimize HetNets for cost and capacitated resilience. In most real life applications, cost is the main limitation to achieve higher QoS levels that are demanded by users. Bi-objective optimization with Pareto optimality allows decision makers to minimize cost and maximize capacitated resilience by selecting the most appropriate solution from a set of non-dominated solutions.

- A third contribution is the comparison of the network designs obtained by optimization of different reliability/survivability metrics. Specifically, network designs optimized for

capacitated resilience are compared with designs optimized for other metrics. The specific network designs of capacitated resilience and the other metrics are identified and their effectiveness are discussed.

Chapter 2

Literature Review

In this chapter, the previous work in literature is summarized. The main focus is on network types, survivability/reliability measures and wireless communication issues.

2.1 Heterogeneous wireless networks

Throughout this dissertation, the term "heterogeneous wireless network" refers to a heterogeneous wireless telecommunications network. As explained before, the source of heterogeneity can be different services used in a wireless network or differences among the properties of nodes.

In this section, heterogeneous wireless networks, including sensor and nonsensor networks, are summarized. In this dissertation, nonsensor HetNets are termed as heterogeneous wireless networks and do not have sensor nodes. The reason for this classification is the large and specific literature of sensor networks.

2.1.1 Nonsensor heterogeneous wireless networks

In nonsensor HetNets various objective functions have been used to optimize a given network. Among them, cost and throughput are the widely used ones. The total number of nodes is a variation of cost. Overhead is a well known metric that defines the extra data that is sent with the actual communication data to route it over a network. It is an important metric for routing protocols. Availability is defined by Choi et al. (2011) as "the probability that a system or service is available at a specific time". Another objective function, redundancy ratio, is the ratio of number of backup systems to working systems. Connected dominated sets are defined in detail in Tiwari et al. (2007) and it can be briefly

defined as a set of vertices $V' \in V$ in which a vertex $i \in V'$ can connect to another vertex $j \in V'$ by using only vertices of V' and every other vertex of $V - V'$ is adjacent to a vertex $j \in V'$. Therefore, this metric is related to the connectivity of a network. Energy consumption of nodes has also been an objective. Relays help connect users with an end node, for example a base station. They are generally used to increase coverage by providing multi-hop connections with a lower cost. Table 2.1 summarizes the objective functions used in the literature.

Table 2.1: Summary of objective functions used in the nonsensor heterogeneous wireless network literature

	Dilmaghani and Rao (2007)	Choi et al. (2011)	Shahnaz and Erlebach (2010)	Tiwari et al. (2007)	Moraes et al. (2009)	So and Liang (2009)
Overhead	X					
Throughput	X					
Availability		X				
Redundancy ratio		X				
Cost			X			
Size of connected dominated set				X		
Energy consumption					X	
# of relay nodes (RNs)						X

Many different decision variables have been used in nonsensor HetNets literature as shown in Table 2.2. Shahnaz and Erlebach (2010) used Steiner subgraphs, where a Steiner tree spans all the nodes in a graph G. Similarly, a Steiner subgraph spans a given set of nodes. As another decision variable, transmission power of a node affects its range, with higher power yielding a larger range. To find the best routing from users to a base station, "edges to select" was used as a decision variable. Number of backup systems is simply the number of additional systems other than the working ones. The location of nodes has a

direct impact on reliability of a network. Nonoverlapping wireless channel assignment is used to prevent interference (see Section 2.4.1 for more information on interference).

Table 2.2: Summary of decision variables used in the nonsensor heterogeneous wireless network literature

	Dilmaghani and Rao (2007)	Choi et al. (2011)	Shahnaz and Erlebach (2010)	Tiwari et al. (2007)	Moraes et al. (2009)	So and Liang (2009)
Transmission power of each node					X	
Edges to connect nodes	X		X	X		
# of backup systems for control systems		X				
Relay locations						X
Wireless channel assignments						X

In the optimization process, many variables have been fixed. For example, number of nodes, transmission ranges of nodes, network density, and number of users are common parameters. Network density is one such parameter and is defined by Tiwari et al. (2007) as the number of nodes per unit area. Choi et al. (2011) used node weight as a parameter. It determines the priority of a node to be secured against failures. Similar to this dissertation, traffic requirements of nodes are fixed in So and Liang (2009). The parameters used in heterogeneous wireless networks are given in Table 2.3.

Various methods have been used to optimize a nonsensor HetNet, including mixed integer programming, approximation algorithms and custom algorithms (Table 2.4).

As discussed in Chapter 1, instead of using survivability/reliability measures, edge-disjoint paths and k-connectivity constraints are usually used in the nonsensor HetNets literature (Table 2.5).

Some studies focused on the architecture of HetNets. For example, Birkos et al. (2012) proposed ROMEO (Remote colloborative real-time multimedia experience over the future Internet) architecture for live 3D video delivery networks. This architecture aims for seamless

Table 2.3: Summary of parameters used in the nonsensor heterogeneous wireless network literature

	Dilmaghani and Rao (2007)	Choi et al. (2011)	Shahnaz and Erlebach (2010)	Tiwari et al. (2007)	Moraes et al. (2009)	So and Liang (2009)
Number of nodes				X		
Different transmission ranges of nodes				X		
Network density				X		
Number of users					X	
Unidirectional/bidirectional edges					X	
Weight of the nodes		X	X			
Traffic requirements of users						X

Table 2.4: Summary of optimization methods used in the nonsensor heterogeneous wireless network literature

	Dilmaghani and Rao (2007)	Choi et al. (2011)	Shahnaz and Erlebach (2010)	Tiwari et al. (2007)	Moraes et al. (2009)	So and Liang (2009)
Approximation algorithm				X		
Custom algorithm			X			
MIP					X	X
Simulation				X		
Queuing model		X				

video session continuity over heterogeneous networks consisting of WiFi, LTE (Long-term evolution, usually referred to 4G LTE) and DVB (Digital video broadcasting) for mobile and roaming users. This dissertation will not cover the technical aspects of these technologies.

Table 2.5: Summary of survivability/reliability measures used in the nonsensor heterogeneous wireless network literature

	Dilmaghani and Rao (2007)	Choi et al. (2011)	Shahnaz and Erlebach (2010)	Tiwari et al. (2007)	Moraes et al. (2009)	So and Liang (2009)
Two edge-disjoint paths	X		X			
k-connectivity		X		X	X	
No survivability/reliability (only connectivity)						X

Summary and discussion

In the above listed studies either k-connectivity or edge-disjoint path constraints are used for survivability/reliability. In this research, "capacitated resilience" is maximized while cost is minimized.

Another main difference of these nonsensor HetNet studies with this dissertation is the level of comprehensiveness of the models. Each of these studies captured a specific aspect of the problem. For example, Dilmaghani and Rao (2007) worked on a possible disaster scenario. In their problem, they deployed a mesh network to increase throughput and reduce overhead. They used Transport Control Protocol (TCP) for their mesh network and observe the bottlenecks in a real application. However, they did not use any optimization approach. The main focus of Choi et al. (2011) was the redundancy of devices. They solved the problem as a queuing system ($M/M/c$), which restricts their model with unrealistic assumptions, for example, exponentially distributed failures and repairs. Also they did not consider the path of communication links from nodes i to j, which is considered in this dissertation as a decision variable. Shahnaz and Erlebach (2010) considered a "node-weighted graph" having different nodes with different costs. However, their problem was to find a Steiner tree and connect all nodes together. Similarly, Tiwari et al. (2007) worked on connected subgraphs. Although

their idea of a node-weighted graph is similar to the one studied in this dissertation, their connectivity assumption does not apply here because connectivity of a user to a base station (or an end node) is required in this dissertation. In other words, users are not required to be connected to each other in this dissertation, but they must be connected to an AP. Ad hoc networks were studied by Moraes et al. (2009) with a focus to find optimum power assignments of nodes. Instead of locating the nodes, they had device positions as an input. It is a different problem because positions of nodes are optimized in this dissertation.

2.1.2 Heterogeneous wireless sensor networks

Although this dissertation does not include sensor networks, some of the studies are related to the HetNets and therefore this section summarizes the relevant wireless sensor network literature. There have been many studies aimed at optimizing heterogeneous wireless sensor networks. In the literature, an important objective function is cost while others are total energy consumption or lifetime of the network. The number of additional nodes (either additional relay or sensor nodes) is minimized in some studies. In addition, transmitted communication packets and latency are two important performance metrics for sensor networks in which realtime information is crucial. Latency is the time delay in data communications (Guidoni et al., 2010). In one study (Wang et al., 2008), the number of high transmission power state nodes was minimized to reduce total power usage, which is an important issue for battery powered sensors. Also, Machado et al. (2010) minimized vacancy which is defined as the area outside of the sensing region. Al-Turjman et al. (2013) used k-connectivity as the performance metric. The objective functions of the selected literature are summarized in Table 2.6.

The decision variables used in heterogeneous wireless sensor networks have been various. The number (and/or location) of relay nodes has been used in many studies. Also, number of additional sensor nodes is another decision variable that is used in some studies. Similarly, the number (and/or location) of base stations has been used as a decision variable. The

Table 2.6: Summary of objective functions used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
Number of relay nodes	X	X	X	X	X							
Lifetime of network						X	X					
Energy consumption								X				
Transmitted communication packets								X				
Probability of a node can securely deliver information									X			
Latency								X				
Cost									X			
Number of high transmission power state nodes										X		
Lifetime of a node (related to energy consumption)		X										
Avg. and Coef. Var. of capacity utilization of RNs		X										
# of linear programming models have been applied in the algorithm						X						
Vacancy							X					
# of additional sensor nodes (SNs)											X	
CPU time	X		X									
k-connectivity												X

decision variable "edges to select" was used to route information from sensors to a base station with flow on each edge (and edges to select) as a decision variable. The decision variables of the selected literature are given in Table 2.7.

Many parameters have been used in the heterogeneous wireless sensor networks literature. Number of nodes, especially sensor nodes, is one of the most frequently used parameters in optimization models. Transmission ranges of sensor and relay nodes are also used in many studies. Grid (or region) size is used as a parameter for initial deployment of sensors or other nodes. As an indicator of network activity of sensor nodes, number of broadcast messages was used by Machado et al. (2010). The network activity parameter is important because it

Table 2.7: Summary of decision variables used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
# and (or) locations of RNs	X	X	X	X	X		X		X			X
# and (or) locations of base station (BS) nodes						X						
# and locations of (additional) SNs							X		X		X	
Edges to use								X				
Binary if SN connected to BS and RN	X											
Flow from SNs to BS and RN	X											
Transmission power of each node										X		
Flow of a commodity from node i to j						X						

directly affects the power usage of the sensor nodes. Table 2.8 summarizes the parameters used in the heterogeneous wireless sensor networks literature.

Many of the constraints in heterogeneous wireless sensor networks are related to either link or device capacity. Qian et al. (2007) used a different connectivity constraint, key connectivity, which affects only less powerful sensor nodes. They defined key connectivity as the probability that less powerful (limited ranged) sensors can connect to network. Lifetime of a network was defined in Al-Turjman et al. (2013) as the time period that the sensor network operates. The constraints previously used are summarized in Table 2.9.

To optimize heterogeneous wireless sensor networks different methods have been used. Among them, traditional optimization methods (e.g., mixed integer programming), custom algorithms and approximation methods are the common ones. Also, a metaheuristic search method, genetic algorithms, has been used. One study used Markov chains to solve the heterogeneous wireless sensor network design problem (Machado et al., 2010). Simulation is another common tool to verify solutions. Custom mixed integer programming methods

Table 2.8: Summary of parameters used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
Number of SNs	X		X				X	X	X	X		X
Number of other nodes	X		X				X					X
Number of edges	X		X									
Transmission ranges of RNs		X	X	X	X							
Transmission ranges of SNs				X	X							X
Traffic load		X										X
Initial energy levels of SNs						X						
Grid/region size	X		X	X				X				
Average distance between nodes								X				
Direction of wireless communication (one-way, two-way)				X								
Network activity level							X					
Pr. of a SN to be active							X					
Bidirectional/unidirectional links											X	

(such as Lagrangian relaxation) have also been also used. Table 2.10 summarizes methods that have been used to optimize heterogeneous wireless sensor networks.

Two/k-edge disjoint paths and k-connectivity are the most commonly used constraints to ensure survivability/reliability in heterogeneous wireless sensor networks. As a variation of connectivity constraints, Guidoni et al. (2010) worked on small world concept, in which there are some shortcuts that reach from a node i to a farther node j. These shortcuts are obtained by using more powerful devices (yielding a larger range). As a variation of all-terminal reliability, Qian et al. (2007) used probability that a node can securely deliver information to ensure survivability in wireless sensor networks. To ensure secure communication between nodes, they proposed assigning different communication keys to sensor nodes (for sending/receiving encrypted information). In their model, a set of edges $E' \in E$ fails after a key is compromised by a successful attack on a node which uses the same key as E'. The survivability/reliability measures used in sensor networks are given in Table 2.11.

Table 2.9: Summary of constraints used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
Restriction on RN placement (only certain areas)	X											
Survivable path from nodes to RNs			X	X			X			X	X	
Survivable path from RN to BS	X		X				X			X	X	
Key connectivity									X			
Capacity of relay/BS nodes		X										
Link capacity (bandwidth)						X						
Max transmission range						X						
Candidate/restricted locations for RNs			X		X							
Cost												X
Lifetime of network												X
No constraints								X				

Table 2.10: Summary of optimization methods used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
Custom algorithm					X			X		X	X	
Approximation method (algorithm)	X	X	X	X		X					X	
Mixed integer programming	X			X		X			X			
Simulation							X	X		X		X
Markov chain							X					
GA									X			
Lagrangian relaxation							X					
New routing scheme						X						

Summary and discussion

Similar to this dissertation, the number and locations of RNs were optimized by Misra et al. (2010). However, instead of considering cost, they minimized the number of deployed

Table 2.11: Summary of survivability/reliability measures used in the heterogeneous wireless sensor network literature

	Misra et al. (2010)	Wang et al. (2007)	Yang et al. (2010)	Han et al. (2010)	Kashyap et al. (2010)	Yang et al. (2009)	Machado et al. (2010)	Guidoni et al. (2010)	Qian et al. (2007)	Wang et al. (2008)	Bredin et al. (2010)	Al-Turjman et al. (2013)
Two/k-edge disjoint paths					X					X		X
k-connectivity (vertex connectivity)	X		X	X	X		X				X	X
Probability of a node can securely deliver information									X			
Probabilistic failures on edges/components							X	X	X			
Small world concept								X				
Routing based survivability						X						
No survivability/reliability (only connectivity)		X										

relay nodes under connectivity constraints. Yang et al. (2010) extended the same problem to have two edge-disjoint paths for both sensor to relay nodes and relay to base station nodes whereas the original problem only considered relay to base station nodes. Han et al. (2010) also worked on the same problem. Differing from these studies, devices with different capabilities (such as range or capacity) and costs are considered in this dissertation. In addition to a mixed integer programming model, these studies proposed approximation methods for special cases of the problems. Yang et al. (2010) reported that their approximation algorithm never exceeds an optimality gap of 100%. Although this gap is large, the method could be useful for large problem instances. In this dissertation, optimum solutions of maximum capacitated resilience problem are not known. Therefore, an optimality gap cannot be calculated for the method in this dissertation. In another similar study, Yang et al. (2009) proposed an approximation algorithm for optimum base station placement. They were the first to present a polynomial time approximation algorithm to solve this problem. Although they did not compare their algorithm with an exact method, their algorithm was reported to be better than other approximations. Major difference compared to this dissertation is

that none of these studies used capacitated edges or considered rerouting of flows in case of a failure. Al-Turjman et al. (2013) considered three dimensional node placement, whereas this dissertation and other studies mentioned in this section place nodes in a two dimensional area.

Wang et al. (2007) minimized cost under connectivity and energy constraints. Their problem is different than the research problem presented in this dissertation, because they did not assign paths from sensors to base stations. Also, they only studied the problem as a minimum set covering problem and flows were not considered. They did not model a mixed integer programming formulation, but they provided an exact algorithm to solve the problem. They also presented approximation methods. Machado et al. (2010) worked on the lifetime of a sensor network. The unique thing about their model is the Lagrangian relaxation, because it is a powerful method to solve large sized problems.

Similar to nonsensor networks, edge-disjoint paths and k-connectivity are the two most common survivability measures in the heterogeneous wireless sensor network literature. Wang et al. (2008) presented an approximation algorithm to minimize the number of high transmission power state nodes under k-connectivity constraints and it outperformed competing approaches in the literature. Kashyap et al. (2010) minimized the additional number of relay nodes in a given network while ensuring connectivity constraints. They gave an exact method and also presented an approximation algorithm with a k-connectivity constraint. Their approximation method works for larger values of k as well as small ones. Similar to Kashyap et al. (2010), Bredin et al. (2010) minimized the additional number of sensor nodes to satisfy k-connectivity constraints for any value of k. It is a different problem than the one in this dissertation, because in their model the network is already deployed. They also provided an approximation algorithm for their problem. Al-Turjman et al. (2013) defined connectivity in a different way by using Laplacian Matrix of nodes. Matrix element (i, j) is -1 if nodes i and j are connected, 0 otherwise The element (i, i) is the number of edges connected to node i. They explained that λ_2, the second smallest Eigen value of the matrix,

defines the minimum number of nodes and links to disconnect the network. Therefore, they maximized λ_2 to maximize connectivity. Guidoni et al. (2010) worked on a small world concept. Similar to this dissertation, they decided on paths in their model ("edges to use" as a decision variable). It is basically a connectivity problem without considering link reliabilities of the links. Qian et al. (2007) considered network attacks and connectivity issues. They proposed a multiobjective GA to minimize cost and maximize the probability that a node can securely deliver information. As explained earlier, the latter metric is a variation of all-terminal reliability. In their multiobjective optimization they used a simple weighted additive objective function, whereas Pareto optimality is used in this dissertation. They did not consider flow assignment or capacity issues, but they chose the edges to build paths. Also, there were no relay nodes in their model. Therefore, this dissertation fills an important gap by proposing capacitated resilience as a survivability/reliability metric and using it in a bi-objective optimization model.

2.2 Survivability/reliability measures

In this section, the most common reliability measures are summarized and compared with capacitated resilience, which is proposed in this dissertation as a new survivability/reliability measure.

2.2.1 Two-terminal reliability

According to Grover (2004), there are two probabilistic failure models used to analyze the survivability of a network: (1) Given occurrence of failure models and (2) Random occurrence of failure models. In the first one the typical question is: "If failure x occurs, how well are network services protected from it?". The second type of models deals with: "How likely is it that a path between nodes has total outage of over x minutes a year?"

Grover (2004) also asked the question "How likely is that at least one path between nodes exists?" These questions are related to "two-terminal reliability" which is defined as

the likelihood that at least one distinct path between (s, t) works. Grover (2004) stated that calculating two-terminal reliability is NP-complete when link failure probabilities are known. The formula of two-terminal reliability is given in Equation 1.2.

A more general form of two-terminal reliability is k-terminal reliability. Ball (1986) defined k-terminal reliability as the probability that there exists an operating path from a source node s to each node in K, where $s \in K$ and $k \equiv |K|$. If $k = 2$, then the reliability is called two-terminal reliability. All-terminal reliability is calculated when $k = n$ (in other words, K is V) where a network can be defined as a graph $G = (V, E)$ having n nodes and m edges. Harms (1995) defined all-terminal reliability as "the probability that there is a path of operating edges from node s to all other nodes". A different explanation would be to define unreliability as the likelihood that every distinct path between (s, t) contains at least one failed (blocked) edge. Two-terminal reliability is a special case of all-terminal reliability where it is calculated for only node pairs (i, j).

According to Dengiz et al. (2002), all-terminal network reliability (also called uniform or overall network reliability) can also be defined as "the probability that every pair of nodes can communicate with each other". The primary design problem is to choose a set of links for a given set of nodes to either maximize reliability given a cost constraint or to minimize cost given a minimum network reliability constraint, as given in Equation 1.1 (in Section 1.3). Dengiz et al. (2002) also used the upper bound estimation technique of Jan (1993) for their all-terminal reliability measure.

These measures consider only edge failures, and nodes are assumed to perfectly reliable. Another assumption is that the failures are independent than each other. This is important for computational ease. Also, in most of the studies, repairs are assumed to take a very short time and are therefore ignored in the models. And, links are either working or failed - degradation is not considered.

These reliability measures can be calculated exactly or approximately. For larger networks, exact calculation of reliability requires extensive computational work, therefore approximation approaches are required.

Ball (1986) stated that a cutset (minimal subset of components whose failure implies the failure of network) and a pathset (minimal subset of components required to operate the network) can be defined by using a "stochastic binary system" (which was first presented by Ball and Nemhauser (1979)) as described below ($S \subseteq T$, where T is the set of all components):

$$\phi(S) = \begin{cases} 1 & \text{if when S operates and T - S fails then the system operates} \\ 0 & \text{if when S operates and T - S fails then the system fails} \end{cases}$$

Pathsets and cutsets were enumerated to evaluate reliability in early studies (Ball, 1986). To use this enumeration method for reliability calculation, the number of pathsets and cutsets must be small.

In an early paper, deMercado et al. (1976) used a partitioning approach to calculate 2-terminal network reliability. Ball (1979) provided a partition based approach to calculate 2, k and all terminal reliability and compared the proposed algorithm with the enumeration methods. The proposed method is more effective than enumeration approaches in terms of CPU time. Provan and Ball (1984) proposed an algorithm to compute two-terminal reliability between nodes s and t in polynomial time based on the number of (s,t)-cuts in the network. The complexity of their algorithm is $\mathcal{O}(|E| + |N|\mu^2)$, where μ is the number of (s,t)-cuts. This algorithm is intractable for large values of μ. Abraham (1979) used an algorithm based on boolean algebra to find the 2-terminal reliability of a network. Disjoint (mutually exclusive) paths between nodes i and j are identified first and then reliability is calculated. Beichelt and Spross (1987) improved the algorithm proposed by Abraham (1979) and calculated 2-terminal reliability more effectively because they reduced the number of disjoint terms and therefore reduced the computational time.

Despite these efforts, exact calculation of reliability becomes intractable for large problems. Therefore, alternative methods are needed to solve the problem.

Van Slyke and Frank (1971) developed an effective approach to calculate the probability of a network being connected and the fraction of communicating node pairs. Their approach was based on simulation. Deeter and Smith (1998) worked on all-terminal reliability (actually they used source-sink reliability, which is a variation of all-terminal reliability) and they used a GA. In their GA model, they used reliability in a penalty function (to ensure a minimum reliability constraint). The calculation method of reliability in their model is taken from Ball and Van Slyke (1977). It is a backtracking algorithm and is a reasonable choice for small sized networks. For larger networks, they suggested Monte Carlo simulation to estimate reliability.

Monte Carlo simulation is a popular tool that has been used in reliability analysis; however, it is an approximation. In eary works, Kumamoto et al. (1977) used Monte Carlo simulation estimate system reliability and Karp and Luby (1985) proposed Monte Carlo simulation to estimate k-terminal network reliability. Lomonosov (1994) estimated reliability using Monte Carlo simulation combining various states of the network. Nel and Colbourn (1990) used Monte Carlo simulation and improved a well-known estimation of reliability bounds (Ball and Provan, 1982) by adding additional constraints to the original estimation. Colbourn and Harms (1988) used linear programming to obtain tighter bounds than (Ball and Provan, 1982) for estimation of all-terminal network reliability.

Dengiz et al. (1997) also worked on all-terminal reliability. They minimized cost under a minimum reliability constraint. They calculated reliability using Equation 2.1:

$$\sum_{\Omega} \left(\prod_{l \in L'} p_l \right) \left(\prod_{l \in L \setminus L'} q_l \right), \tag{2.1}$$

where Ω = all operational states, $L' =$ set of operational links, $L' \in L$

p_l = reliability of link $l, q_l = 1 - p_l$

However, Dengiz et al. (1997) reported that this formula is not tractable for large values of Ω. Therefore, they proposed an estimation method based on Monte Carlo simulation. In their GA methodology, they included reliability as a penalty in the objective function.

Hui et al. (2003) proposed a Monte Carlo approach based on simulation and partitioning edge sets to estimate network reliability. They only considered edge failures (nodes are assumed to be perfectly reliable) and they calculated k-terminal reliability. They used a structure function $\phi(x)$ which is originally proposed by Ball and Nemhauser (1979). They reported that the new method improved the performance dramatically in comparison to the Crude Monte Carlo and Permutation Monte Carlo approaches. This approach could have been adopted in this dissertation to estimate reliability because the problem herein is focused on heterogeneous wireless networks, but an exact method has been developed to calculate capacitated resilience.

Hui et al. (2005) combined a Cross-Entropy method and a Monte Carlo simulation approach to estimate k-terminal network reliability. They used cross-entropy with Crude Monte Carlo, Permutation Monte Carlo and Merge Monte Carlo simulation based approaches. In their estimation approach, they sampled only up time of each edge and then calculated the probability of the network functioning at a specific time. By iteratively estimating the reference parameter of the cross-entropy method and using likelihood formulations, reliability of the network is estimated. Kroese et al. (2007) also approached reliability estimation by using a cross-entropy method. In their problem, they designed the most reliable network under a budget constraint. Hui et al. (2005) reduced the variation in the reliability estimation dramatically and increased the speed of the calculation. Also, they concluded that Permutation Monte Carlo and Merge Monte Carlo yield better solutions than Crude Monte Carlo for estimating reliability. This proposed approach might be used for extensions of this dissertation, because the authors reported promising performance for large sized networks. Hui (2007) proposed a new method (Synchronous Construction Ranking) for Monte Carlo

simulation to rank different network designs according to their k-terminal reliability values. Their main concern was to find the most reliable design among many candidates.

Ramirez-Marquez and Coit (2005) estimated two-terminal reliability in multi-state networks by using Monte Carlo simulation. Multi-state networks are consisted of edges with different available capacities where the current state (capacity) of the edge can take values of $b_i \geq 0$. Thus, multistate two-terminal reliability ($M2TR_d$) is defined as the probability that a flow requirement (d units of flow) between nodes i and j can be served by the network of multi-state edges.

Cancela and El Khadiri (1995) proposed a variance reduction technique for Monte Carlo simulation to estimate k-terminal reliability. They used a recursive method to reduce variation of Monte Carlo simulation by evaluating the unreliability of the network. Their method yielded faster results than existing methods, namely Dagger Sampling, Sequential Construction, Bounds, Failure Set and Merge Process approaches. This method was further improved by Cancela and El Khadiri (1998) by applying series-parallel reductions. A series reduction is to replace two serially connected links (with reliabilities r_{ij} and r_{jl}) by one link where the new reliability is calculated as $r_{ij} * r_{jl}$. A parallel reduction is to replace two parallel connected links by one link where the reliability of the new link is $r_{ij} + r_{kl} - r_{ij} * r_{kl}$. These reduction techniques were proposed by Rushdi (1984) in an earlier study to calculate k-terminal reliability. Similar to Cancela and El Khadiri (1995), Cancela and El Khadiri (2003) proposed a new formulation to reduce variance of Monte Carlo simulation to estimate k-terminal unreliability. They also combined the new formulation with the series-parallel reduction approaches of Cancela and El Khadiri (1998). Their formulation yielded better solutions in terms of CPU time.

In a recent study by Grosan et al. (2009), the objective functions of cost and resilience were used. They defined resilience by assigning a backup path in addition to a primary path between commodity pairs. Similar to k-connectivity and edge-disjoint paths, the backup paths of Grosan et al. (2009) ensure survivability but the paths are not required to be

disjoint (they can have more than one common edge or node). Also, they only consider link failures. Their cost function is simply the edge costs. They minimized the number of common edges with primary and backup paths to ensure resilience. Inversely, they also maximized the number of non-common edges between backup paths. They used Pareto optimality for these three objective functions. In summary, their approach only assigns primary and backup paths and it minimizes common edges between the primary and backup paths. This can be accomplished using edge-disjoint path constraints more effectively. They did not compare their method with other methods, and only discussed the network designs generated by their algorithm.

2.2.2 Probability that two nodes can communicate

Wilkov (1972) presented the "probability of two nodes communicating" measure assuming that all communication-link failures and computer-center breakdowns are statistically independent and that each communication link fails with probability p and each computer center goes down with probability q. This metric is similar to two-terminal reliability; however, it includes both link and node failures. Calculation of $P_c(a,b)$, the probability of successful communication of any operating nodes a and b, is approximated by Equation 2.2.

$$P_c(a,b) = \sum_{i=0}^{b} A^e_{a,b}(i)(1-p)^i p^{b-i},\ p \gg q \tag{2.2}$$

$A^n_{a,b}(i)$ is the number of combinations of i nodes such that if they are operative and the remaining $(n-2-i)$ nodes fail, there is at least one communication path between nodes a and b. Wilkov (1972) also gave an approximation formula for the $q \gg p$ case. However, a comparison with an exact method is not given.

This approximation calculation could be used in our problem, but a downside is the requirement of homogeneous failures (with failure probability p). Also the calculation of $A^n_{a,b}(i)$ is computationally expensive.

2.2.3 Expected loss of traffic (ELT)

Another measure for survivability/reliability is the expected loss of traffic (ELT). According to Grover (2004), it is basically the expected number of lost demand-minutes over a year. In this calculation, the total demand (flow) between i and j is not required to be along a single path – multiple paths are allowed. The demand d_{ij} may be realized by routing over P different routes, but total demand must be satisfied. Grover (2004) gives the formula of ELT in Equation 2.3.

$$ELT_{i,j} = \sum_{p=1...P} \left(d^p_{ij} \sum_{\forall k \in (S,N)} \delta^p_{ij}(k) U_k \right) M_0 \tag{2.3}$$

Thus, ELT is the "sum of the demand-weighted unavailability of each distinct (not disjoint) path employed to satisfy total demand d_{ij}" (Grover, 2004). Disjoint paths can be assured by additional constraints.

Similar to capacitated resilience, ELT can be calculated if flow from i to j is routed over different paths. ELT can also be interpreted as unreliability with split flows, however, it returns the lost demand-minutes over a year. Therefore, unlike capacitated resilience, ELT is not scaled within 0 and 1. In the capacitated resilience calculation, if a flow cannot be rerouted, capacitated resilience is equal to zero. On the other hand, ELT does not capture rerouting possibilities. It only calculates the unreliability for given splits of a flow and use them in the lost-demand calculation. As another difference, ELT calculation assumes that all distinct paths (the given splits) are independent. This assumption may lead to a higher ELT value than it should be if the splits are routed over two paths having a common network

component. In capacitated resilience calculation, independent subgroups of alternative paths (see Section 3.2.3 for details) are used to overcome the path dependency issues.

2.2.4 Traffic efficiency

Konak and Bartolacci (2007) used the traffic efficiency (TE) measure which was originally proposed by Kubat (1989). The main reasons for them to use TE instead of reliability are to include node failures and to consider traffic flows. They also included a 2-node connectivity constraint in their model.

They defined the state of edges and nodes as either 0 or 1. If a component is operational, then its state is 1, otherwise it is 0. They assumed that reliability (and unreliability) of nodes and edges are given as inputs.

They defined TE as the expected value of $\Omega(x)$, which is the fraction of the traffic that the network delivers in state x (Equation 2.4). The definition of $\Omega(x)$ is given in Equation 2.5. They defined τ_{ij} to be 1 if there is a path between nodes i and j. γ is defined as the total traffic demand of the network. This formulation needs to consider 2^{m+n} states for calculation of TE.

$$TE = E[\Omega(x)] = \sum_{x \in S} \Omega(x) * P\{x\} \tag{2.4}$$

$$\Omega(x) = \frac{1}{\gamma} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \tau_{ij}(x) t_{ij} \tag{2.5}$$

There are four main differences between TE and the capacitated resilience metric proposed in this dissertation. First, TE does not consider capacity of edges (or nodes), however, capacitated resilience considers capacity in finding the initial path (the most reliable path)

and in rerouting. Second, TE considers node failures as well as link failures, whereas capacitated resilience considers only link failures. This is an important difference of TE with capacitated resilience and the other connectivity based reliability measures, such as all-terminal reliability. Third, paths from i to j are found for each state of network to calculate TE, whereas all distinct paths are found to calculate capacitated resilience for a given state of network. In other words, capacitated resilience calculates all possible paths from originating node i to all operating APs. And for capacitated resilience, rerouting is only enacted if the initially assigned path fails (any failure in the path). Therefore, distinct paths are calculated without using the failed component and all components other than the failed one(s) are assumed operational. Nevertheless, TE considers all possible states of all components in the network where network components are either operational or failed. Last, split flows are allowed in routing process of capacitated resilience when maximum capacity of an alternative path is reached. On the other hand, as seen from Equation 2.5, TE does not split the traffic flow between nodes i and j. These are the main differences between TE and capacitated resilience.

An additional difference is the calculation method of TE. It is calculated by an efficient simulation technique based on "sequential construction" (summarized in Section 4.3.2) by Konak and Bartolacci (2007). In the original article proposing the TE measure, Kubat (1989) combined a Monte Carlo based simulation and analytical approach to calculate TE. The motivation of the calculation approaches of these studies is the intractability of the calculation of TE for large sized networks. However, in this dissertation, capacitated resilience is calculated by enumeration of paths. This enumeration may be intractable for large networks. Therefore, limits on the number of paths (see Section 3.2.2) and cut sets (see Section 3.2.4) are used to estimate capacitated resilience in larger networks.

Similar to capacitated resilience metric, TE is already normalized because it is the expected fraction of the traffic that the network delivers. Therefore, two different network

designs can be readily compared using TE. Both TE and capacitated resilience prefer larger values, and for both metrics the upper limit is 1.

2.2.5 k-connectivity and edge-disjoint paths

A two-connected (or two-node connected) graph has at least two node-disjoint paths between every pair of vertices (Bondy and Murty, 2008). If two paths are edge-disjoint, they do not have a common edge (but can have a common node). Similarly, if two paths are node-disjoint, they do not have a common node (Kashyap et al., 2010). Therefore, if a graph is two-connected, it is also a two-edge connected graph. However, the reverse of this statement is not necessarily true. These connectivity constraints are widely used to ensure survivability/reliability of a network.

2.2.6 Summary and discussion

To summarize, some survivability/reliability metrics have been used in the HetNets literature although it is most common to add k-connectivity or edge-disjoint path constraints. Among them, two-terminal and all-terminal reliability measures are the most frequently used ones. These metrics consider link failures. Similar to these, "probability that two nodes can communicate" also captures link failures, but it includes node failures as well. However, flow between nodes is not taken into consideration in these metrics. It is important to have flows and capacities of the devices because rerouting of flow is possible in case of a failure. The proposed metric in this dissertation, capacitated resilience, uses reliability and rerouting options at the same time. It assumes capacitated links due to capacity of wireless devices. Capacitated links have been considered in some studies (such as Benyamina et al. (2009a) and Capone et al. (2010)), whereas some studies (for example Shahnaz and Erlebach (2010)) did not consider capacity of links. Using capacitated wireless links and devices for resilient heterogeneous wireless network design problem is more realistic.

Table 2.12 compares capacitated resilience with the other reliability/survivability metrics. The most important difference of capacitated resilience is the consideration of capacity. Another difference is that capacitated resilience prioritizes the availability of rerouting options. Also, it allows split flows in rerouting. On the other hand, capacitated resilience assumes that nodes are perfectly reliable. In conclusion, capacitated resilience is a more comprehensive metric as it includes both reliability and rerouting under capacity constraints.

Table 2.12: Comparison of reliability/survivability metrics

	Two-terminal rel.	All-terminal rel.	Pr. that nodes can com.	ELT	Traffic efficiency	k-connectivity	Edge-disjoint paths	Capacitated resilience
Rerouting						X	X	X
Capacity								X
Node Failures			X	X	X			
Link Failures	X	X	X	X	X			X
Split flows				X				X

In Section 3.3, the reliability/survivability metrics presented in this chapter are calculated for an example network and the differences are discussed.

2.3 Shortest path problem variations: Constraint shortest path and k shortest path

The rerouting calculation of capacitated resilience requires enumeration of all possible alternative paths under capacity constraints. This problem is known as the constrained k shortest path problem. The problem reduces to the constrained shortest path problem if k equals to one. Constrained shortest path and k shortest path problems are well known and they are summarized in this section.

In one of the earliest works on the shortest path problem Yen (1971) proposed an exact method to find k shortest paths between two nodes. However, this algorithm does not include

constraints. Since that time, the shortest path problem has been solved under constraints and the problem was termed the "constrained shortest path problem". According to Dumitrescu and Boland (2001), the weight constrained shortest path problem is to minimize the cost of a route between two nodes while ensuring the total edge weight is less than a predefined value. Another variation of this problem uses node weights instead of edge weights. The basic formulation of this problem is taken from Dumitrescu and Boland (2001) and given in Equation 2.6.

$$min \sum_{e \in E} c_e x_e \tag{2.6a}$$

$s.t.$

$$\sum_{e \in \delta^+(i)} x_e - \sum_{e \in \delta^-(i)} x_e = \begin{cases} 1 & \text{, if } i = s \\ -1 & \text{, if } i = t \\ 0 & \text{, if } i \in V \setminus \{s, t\} \end{cases} \qquad \forall i \in V \tag{2.6b}$$

$$\sum_{e \in E} w_e x_e \leq W \tag{2.6c}$$

$$x_e \in \{0, 1\},\ \forall e \in E,\ \delta^+(i) = \{(i, j) \in E\},\ \delta^-(i) = \{(j, i) \in E\} \tag{2.6d}$$

Dumitrescu and Boland (2001) reported that the weight constrained shortest path problem is closely related to the shortest path problem with time windows and the resource constrained shortest path problem. Also, the bi-objective (cost and weight minimization) shortest path problem is another variation of this problem.

To solve this problem, mixed integer programming formulations were developed in many studies. However, other methods were devised to solve the problem more effectively. Among

them, Lagrangian relaxation, column generation and dynamic programming are the most common ones.

Desrochers et al. (1992) worked on vehicle routing with time windows problem, which is a variation of the weight constrained shortest path problem. In this problem, time windows (allowable delivery times) are added to the vehicle routing problem in which the minimum cost route is sought between two nodes. They used distance between nodes as cost and the weight of edge (i, j) is defined as the time duration (including the service time at customer i). They defined the problem as a set partitioning problem to select a set of minimal cost routes (Equation 2.7):

$$
\begin{aligned}
& min \sum_{r \in R} c_r x_r \\
& s.t. \\
& \sum_{r \in R} \delta_{ir} x_r = 1 \ , \ i \in N \setminus \{d\} \qquad (2.7) \\
& x_e \in \{0, 1\},\ r \in R,\ \delta_{ir} = 1 \text{ if route } r \in R \text{ visits customer } i \in N \setminus \{d\},\ 0 \text{ otherwise} \\
& \text{where } d \text{ is the central depot that routes are originating and terminating at}
\end{aligned}
$$

In this formulation, R is the set of feasible routes for the vehicle routing problem with time windows. x_r is a binary variable (1 if route r is used). Since the number of columns (i.e., feasible routes) are extremely large, Desrochers et al. (1992) presented a column generation approach.

Barnhart et al. (1998) also used column generation to solve an aircraft routing problem which is similar to vehicle routing with time windows. Differing from the problem solved by Desrochers et al. (1992), they simultaneously solved a fleet assignment problem and an aircraft routing problem (The problem reduces to the aircraft routing problem when there is only one fleet to assign.) Holmberg and Yuan (2003) used column generation to find the

shortest path under time-delay and reliability constraints for capacitated multicommodity network flow problems.

In an early influential paper, Beasley and Christofides (1989) solved the resource constrained shortest path problem by a Lagrangian relaxation formulation. In their problem, the resource constraint was the budget of the traveler. Again in an early study, Handler and Zang (1980) solved the constrained shortest problem by a Lagrangian relaxation algorithm. They used a knapsack constraint in their problem. Their algorithm terminates at the kth shortest path which is the first path satisfying the constraints.

Irnich and Desaulniers (2005) summarized solution methodologies as dynamic programming, Lagrangian relaxation, constraint programming and heuristics. Dynamic programming is used to build new paths. A trivial path P is extended for all feasible combinations and new paths are checked whether they are useful or not. Paths are often encoded by labels in dynamic programming algorithms to solve the shortest path with resource constraints problem (Irnich and Desaulniers, 2005). They explained the usage of Lagrangian relaxation in detail. They also defined how to use constraint programming using a "domain reduction algorithm" for each constraint. Heuristic methods, such as "preprocessing" techniques, can be used to eliminate edges and reduce the network. "Dynamic programming heuristics" stop if a predefined number of negative (that is, cannot enter the basis) columns are found.

Carlyle et al. (2008) improved the Lagrangian relaxation approach by closing the optimality gap using an enumeration of near-shortest paths. In this dissertation, this approach would be useful because rerouting of flows (for the capacitated resilience calculation) requires enumeration of all possible paths. However, this approach yields an approximate solution.

Avella et al. (2002) proposed a heuristic to solve large resource constrained shortest path problems. Their heuristic depends on penalty functions and provides approximate solutions. In many cases, their method yielded better solutions with lower number of iterations than Lagrangian relaxation. They eliminate resource constraints by using an exponential penalty

function. This approach could be used in this dissertation to find the most resilient paths under a capacity constraint, but it, too, is an approximate method.

Ribeiro and Minoux (1985) proposed a heuristic method to get good feasible solutions for double constrained shortest path problems. Their algorithm generates partial solutions and combines them to a final solution using enumeration. They showed that the method produces good, feasible solutions even for hard constraints. This method might be useful as a future work of this dissertation by finding the most resilient path for instances with capacity and minimum hop constraints.

Mehlhorn and Ziegelmann (2000) worked on the resource constrained shortest path problem where there are k resource constraints. They proposed an algorithm, "hull approach", to solve this problem and showed that it provides good upper and lower bounds for the problem. It is a polynomial algorithm when k equals 1. This algorithm could be useful in this dissertation if more than one constraint (such as capacity and minimum number of hops) are applied. However, only capacity is considered herein.

van der Zijpp and Catalano (2005) proposed an algorithm to enumerate k-shortest paths with resource constraints. Their problem is equivalent to a shortest path problem with resource constraints if k is equal to 1. Their algorithm finds the constrained shortest paths directly instead of selecting feasible ones from a large set, which reduces CPU times dramatically.

Feillet et al. (2004) presented an exact algorithm to find the optimum path for the constrained shortest path problem. Their algorithm is a label-correcting algorithm and the problem is a restricted case where the paths are constrained to use a node only once. In their problem, capacity is defined for nodes, which is similar to the problem presented in this dissertation. Righini and Salani (2008) worked on the same problem and proposed a dynamic programming model. They also used a "state-space relaxation" based dynamic programming method in which an infeasible region can be projected onto a feasible region without guaranteeing optimality.

As another variant, Villeneuve and Desaulniers (2005) worked on the shortest path problem with forbidden paths. In this problem some paths (edges) are restricted in building shortest paths. They proposed a polynomial time algorithm and stated that their method can be used to eliminate cycles in k different shortest paths between nodes i and j. In this dissertation, this problem may have been useful to find distinct paths (for rerouting) if forbidding some edges was necessary.

Similar to Desrochers et al. (1992), Irnich and Desaulniers (2005) defined the same problem and they stated that this problem is very close to a multiobjective function where time and cost are considered. They also noted that paths can be incomparable because one path may be better than another for one criterion (for example, cost) but can be worse for the other one (e.g., time). They also classified different variants of this problem. They distinguished problems according to formulation of the resource constraints, existence of path-structural constraints, objective and underlying network.

Irnich and Villeneuve (2006) proposed an algorithm to eliminate cycles with length k from resource constrained shortest paths. In this study, they proposed a pseudo-polynomial labeling algorithm. They found that this method solves hard problem instances (with wide time windows) faster than the other methods in literature. They also showed that their algorithm can handle Pareto optimal resource constrained shortest paths. In an early work, Aneja and Nair (1978) solved the constrained shortest path problem with a bicriteria model. Warburton (1987) presented a method to approximate Pareto optimal paths for multiobjective shortest path problems.

2.3.1 Summary and discussion

In relation to constrained shortest path problems, the problem in this dissertation is somewhat different. In this dissertation, the route with maximum reliability is sought under capacity constraints. Edge reliabilities have been transformed with a logarithm to linearize the model and the k shortest path problem has been solved under a capacity constraint.

However, similar to previous studies (Desrochers et al., 1992; Barnhart et al., 1998) there are a large number of paths to enumerate. The k shortest path problem helps to reduce the size of the problem for larger networks by limiting the number of rerouting options (k). These distinct paths are important for the rerouting calculation of capacitated resilience. Reliability of a path is maximized, instead of minimizing the cost. Also, the weight constraint is replaced with capacity of nodes. In this dissertation, the k shortest path problem is solved using Yen's algorithm (Yen, 1971), presented in Section 3.2.2, because of its ease for implementation. Nevertheless, the solution approaches presented in Section 2.3 might be adopted to solve the problem. For example, as a future study, column generation could be applied to calculate capacitated resilience. All possible paths could be generated as columns and the restricted problem solved.

In this dissertation, capacity constraints are based on node (device) constraints. Hence, a transformation from node capacities to edge capacities is required. As demonstrated in Equation 2.8, device capacities can easily be converted to edge capacities. The capacity of edge (i, j) is the minimum of the capacities of devices i and j that the edge connects. Therefore, edge capacity constraints ensure that the capacity of nodes are not exceeded.

$$l_{ij} = \min\{l_i, l_j\}, \text{ where } l \text{ denotes capacity} \tag{2.8}$$

In this dissertation, enumeration of all possible paths is an appropriate solution for small sized networks; however, for realistic sized problems more computationally effective solution approaches must be used. A limit on the number of alternative paths (k) helps to approximate for larger networks (see Section 3.2.2 for more details). Cut set size is another parameter in the capacitated resilience calculation (see Section 3.2.4 for more details).

2.4 Some issues of wireless networks

2.4.1 Reliability of a wireless link and network density

In one of the earliest and the most influential works, Takagi and Kleinrock (1984) worked on multihop packet radio networks and defined "probability of successful transmission" by Equation 2.9a. In this formula, S is one-hop throughput, which is defined as the average number of successful transmissions per slot from the terminal (Equation 2.9b), and N is the expected number of nodes within transmission range. $Pr.(P \rightarrow Q)$ is the probability that the transmission from P to Q is successful. Transmission is only possible if none of the terminals (including Q) within the range (R) of Q transmits when P is transmitting to Q. p is the probability that a terminal transmits a packet and it depends on N (number of nodes within range). p is calculated according to Equation 2.9c. Takagi and Kleinrock (1984) found the optimal number of nodes in a transmission range as 7.72 to maximize throughput. The key of this formulation is that the probability of successful transmission is calculated considering interference in the network due to density. Similarly, several other studies defined probability of successful transmission for wireless networks by taking interference concept into consideration (Hunter et al., 2008; Huang et al., 2008; Hira et al., 2007).

$$\text{Pr. of Successful Transmission} = \frac{S}{p} \quad , \; where \tag{2.9a}$$

$$S = Pr.(\text{there is at least one terminal within } R) \tag{2.9b}$$

$$* \, Pr.(P \text{ transmits}) * Pr.(P \rightarrow Q)$$

$$p = \frac{2}{N + 2 + \sqrt{N^2 + 4}} \tag{2.9c}$$

The reliability of a link defined in this dissertation and "probability of successful transmission" (Takagi and Kleinrock, 1984) are related to each other in terms of their definitions.

Both definitions consider the probability if nodes i and j can communicate with each other. However, they have a major difference in terms of their calculation method. The probability of successful transmission takes interference (due to number of nodes in transmission range) into account, however, the reliability of a link only considers the distance (which affects signal quality) between the nodes. Reliability also depends on signal quality which is affected by distance, and is assumed to be negligible in this dissertation.

In another study, Camp et al. (2006) defined the probability that there is a route between nodes i and j (denoted as R_{ij}) for wireless mesh networks. Similar to the definition of path reliability given in this dissertation, they calculated the product of the probabilities of all links in a path between i and j (Equation 2.10). In this formulation, S_l is the signal strength of link $l \in R_{ij}$ and T_{min} is the minimum required signal strength.

$$Pr(R_{ij}\ exists) = \prod_{\forall l \in R_{ij}} Pr(S_l > T_{min}) \tag{2.10}$$

Xue and Kumar (2004) stated that the number of connected neighbors affect the capacity of an ad hoc network due to interference of nodes. Node i interferes with node j when it broadcasts within the range of j. Transmission range (r) determines the number of nodes in the neighborhood and that interference increases on order of r^2 (Xue and Kumar, 2004; Gupta and Kumar, 2000). In another study, Raniwala and Chiueh (2005) stated that the bandwidth of a wireless network using the IEEE 802.11 protocol can be decreased by interference due to the relays in the same path or neighboring paths.

Hekmat and Van Mieghem (2004) used a path-loss law model for radio propagation to calculate interference of mobile ad hoc networks. In their model, the mean value of received signal power (p_a) decreases as distance (d) between devices increases. The calculation of p_a is given in Equation 2.11 where constant c is affected by transmission power and some other properties of the devices, and η is the path loss exponent between 2 and 6. η is 2 for free

space, and 5 for a building environment with obstacles. Thus, they modeled interference with this assumption. Their assumption of degradation of signal quality has not been used in this dissertation to keep the model more tractable. Also, the model in this dissertation is more general and the specific environment is not known.

$$p_a = cd^{-\eta} \tag{2.11}$$

Hekmat and Van Mieghem (2004) also showed that delays in communication increase and network throughput reduces as the number of nodes increases. They found that when network density increases, the average number of hops to communicate decreases but fewer nodes transmit simultaneously. Therefore, capacity per node reduces. In this dissertation, density of a network is adjusted according to the number of users in the network to address this issue.

Takagi and Kleinrock (1984) stated that if the average number of terminals in a transmission range is N, then the probability of successful transmission is proportional to $1/N$ in multihop packet radio networks. A shorter transmission range includes a smaller number of terminals in a neighborhood and therefore means a lower possibility of collision. However, a long transmission range increases the chance of finding an appropriate receiver in a desired direction. Takagi and Kleinrock (1984) worked on this trade-off and investigated the optimum number of average terminals in a transmission range. In another study, Grossglauser and Tse (2002) assumed that all nodes can communicate with each other in a mobile ad hoc network. Their analysis showed that throughput for each communication pair in the network decreases proportional to $1/\sqrt{n}$ (where n is the number of nodes per unit area) as n increases.

In a real life application, reliability of a link is affected by the number of nodes (links) in the transmission range. Probability of successful transmission is proportional to $1/N$ (where

N is the average number of nodes in transmission range) due to the interference caused by density (Takagi and Kleinrock, 1984).

2.5 Capacity of a wireless network and interference issues

As a simplistic view, the number of APs definitely affects the capacity in this dissertation. All users must connect to an AP; however, possible routes to reach an AP can be extremely difficult to find if a limited number of APs exist in a network. In other words, more APs increase the chances to balance the flow of the network. The main bottleneck of a given network can be investigated by a graph theoretical approach. To determine whether a specific node is a bottleneck, the cut-sets of the network should be checked. If the node is repeatedly included in many of the node cut-sets of a network, that device is a bottleneck. On the other hand, another definition of a bottleneck would be a device that has no or limited capacity to route flows. Therefore, the number of devices affects the capacity of a network. In this dissertation, both device capacities and cut-sets are considered for the capacitated resilience calculation.

Another issue with the number of devices in a network (or density) is the interference in the wireless channels. Nevertheless, the effects of interference caused by neighbors have not been included in the dissertation because this complicates the mathematical model which is already hard to solve. It is assumed that nonoverlapping channels are used to avoid interference issues.

The issues on interference and capacity have been studied in the wireless networks literature and therefore they are summarized in this section, even though they are not included in this dissertation.

Xue and Kumar (2004) showed that an ad hoc wireless network (having n nodes) becomes (asymptotically) connected if each node is connected to more than $5.1774 \log n$ nearest neighbors n goes to ∞. By definition, the number of connected neighbors affect the capacity of the network due to interference of nodes. They also mentioned that device i interferes

with device j when it broadcasts within the range of j. Therefore, more links cause more interference and, eventually, a lower capacity of the network. However, there is a trade-off between the number of links (number of neighbors) and reliability of a network. As the expected number of neighbors (number of links) increases, the chance of having a one hop connection increases ("relaying burden" decreases) but overall interference of network grows. Relaying burden means the requirement of a node to relay the packets from other nodes. Although this is a natural result of multi-hop communication, a higher level of relaying burden is not desired because it increases the loads on RPs by requiring them to transceive a larger number of packets from other nodes (users or RPs). Transmission range (r) affects not only the number of neigbors but also the relaying burden and interference (Xue and Kumar, 2004). Relaying burden grows with order of $1/r$, whereas interference grows with the order of r^2 (also discussed in Gupta and Kumar (2000)). Therefore, less range and fewer neighbors are better in terms of interference, but the network may be disconnected if the range is too small.

In the problem solved in this dissertation, the network has a different structure. Its structure is closer to the one presented by Gastpar and Vetterli (2002) in which there is only one active link between i and j and all remaining nodes are simply relay nodes for conveying information between i and j. This is similar to the user to AP connection of this dissertation. However, in this dissertation instead of the one active link of Gastpar and Vetterli (2002) there are as many connections as the number of users. By using the min-cut maximum flow theorem, Gastpar and Vetterli (2002) showed that the capacity of the network goes to $O(\log n)$ bits per second when n (number of nodes in the network) goes to ∞.

Li et al. (2001) used simulation to analyze the capacity of a wireless ad hoc network. One important result related to this dissertation was the effect of multi-hop communication in an ad hoc network. They showed that throughput significantly reduces as the number of hops increases, but the reduction becomes very small after some number of hops. If n is the total number of nodes in the network, they showed that a wireless network (using

IEEE 802.11) approaches its theoretical maximum capacity per node ($O(1/\sqrt{n})$), which was originally proposed by Gupta and Kumar (2000). Therefore, capacity of a network reduces as the network becomes denser .

Reliability of a wireless mesh network increases when there are more redundant paths for communication pairs to alleviate catastrophic effects due to failure of bottleneck links (Bruno et al., 2005). Raniwala and Chiueh (2005) explained the term "path capacity" which is the minimum residual bandwidth of the path that connects a WMN node to the wired network. Another approach to analyze wireless network capacity is "gateway capacity" which takes into account the capacity of the gateway (AP) link with the assumption that a bottleneck is caused by the gateway (AP) connection. A gateway is simply an AP, which is explained in this dissertation. Raniwala and Chiueh (2005) also stated that a bottleneck is generally located on links around an AP due to heavy collision and interference. Therefore, the number of APs affects the capacity of a network. However, Raniwala and Chiueh (2005) added that bottlenecks can occur even in intermediate wireless links due to other radio sources and therefore path load balancing can be a more effective strategy to improve network capacity instead of AP (gateway) load balancing. Similarly, Jun and Sichitiu (2003) showed that gateways (APs) in a WMN are the main bottlenecks. If n is the number of users served by an AP (gateway), they claimed that the available capacity of each node is affected by order of $1/n$. Adding more APs increases the capacity and reliability of the network (Jun and Sichitiu, 2003). The number of APs directly affects the capacity of both individual nodes and the network. Jun and Sichitiu (2003) computed throughput through identification of bottleneck collision domain which was defined as the area in the network limiting the amount of data that can be transmitted in the network.

Wang and Liu (2006) proposed a linear programming model to find maximum capacity (in terms of throughput) for a given network. They investigated ad hoc networks and did not consider mesh type networks, but their methodology could be adapted to this dissertation as a future work to calculate throughput.

Wu et al. (2006) assumed that mesh routers use different channel assignments to prevent interference in their model. The same assumption has been included in this dissertation. A non-overlapped channel is used for one of the routers when two of the routers are close enough to interfere with each other. As summarized in Wu et al. (2012), IEEE 802.11a has 12 non-overlapping channels, whereas IEEE 802.11b/g has only three. In their proposed queuing model, a bottleneck is defined as the average delivery delay of data requests at the gateway nodes (APs). Also bottleneck throughput is defined as the "maximum feasible data requesting frequency from all users attached to a mesh router". They showed that the bottleneck delay decreases when the number of APs (gateways) increases. They assumed the location of gateways do not affect bottlenecks because they ignored the delay caused by routers. However, this is an issue in real life problems. By using their $M/D/1$ queuing model, they found that the number of APs must be greater than $\lambda N s$ to ensure a steady network where N is the number of mesh routers, λ is the mean arrival rate of data to the mesh routers and s is the server time at an AP node.

Chapter 3

Proposed metric: Capacitated resilience

3.1 The need for a new metric

In network design literature, requiring k-connectivity or edge-disjoint paths are the most common ways to ensure survivability. Other reliability metrics, namely k-terminal reliability (Grover, 2004), all-terminal reliability (Harms, 1995), "traffic efficiency" (Konak and Bartolacci, 2007), "probability that two nodes can communicate" (Wilkov, 1972) and "expected loss of traffic" (Grover, 2004), have also been used. Among them, two-terminal and all-terminal reliability measures are the most frequently used ones. These well known reliability/survivability measures, and the differences between them and capacitated resilience are summarized in Sections 2.2 and 2.2.6, respectively.

The main motivation for developing a new metric is to include rerouting options as well as reliability when considering capacitated links and devices. Capacity is one of the main differences. If a device (or a link) does not have enough capacity then traffic cannot be routed on that device (or link). Connectivity based survivability metrics and most reliability/survivability metrics do not consider capacity.

Rerouting is an important issue for session continuity in telecommunication networks and it is the essential part of the capacitated resilience metric. For example, capacitated resilience becomes zero if there is no alternative path available. Capacitated resilience emphasizes session continuity in network design. Unlike connectivity constraints, it also allows comparison of different network designs. In other words, two different k-connected network designs can have different reliability and capacitated resilience values.

The capacitated resilience metric proposed in this dissertation can be calculated with an exact method. In the network reliability/survivability literature, metrics were mostly

calculated by approximate methods, such as Monte Carlo simulation. The studies using exact methods made many simplifying assumptions to keep problems tractable and do not consider alternative paths. Connectivity based metrics, such as k-connectivity, have only k different alternative paths and they do not consider reliabilities of the paths. This is another difference between capacitated resilience and the others.

3.2 Proposed metric: Capacitated resilience

Network level capacitated resilience is the weighted average of capacitated user resiliences in terms of their traffic requirements. Equation 3.1 presents the calculation of capacitated network resilience. In this equation, w_i is the weight of User i, which is the proportion of traffic flow of User i to the total traffic flow of all users. Note that the traffic flow of a user is assumed to be constant over time. The capacitated resilience calculation requires user level capacitated resilience values to be known. If the network consists of one user, then capacitated user resilience is equal to network resilience.

$$Capacitated\ Resilience = \sum_{i \in\ Users} w_i * Capacitated\ Resilience\ (U_i) \tag{3.1}$$

User level capacitated resilience is defined in Equation 3.2. $R(p_i)$ denotes the reliability of the assigned path of User i where the user (U_i) is assigned to a device whose path has maximum reliability (from the user to an AP). The resilience factor scales the user reliability. To calculate it, all alternative paths (those having available capacity) from User i to any available access point are identified first. Then, reliabilities of the alternative paths are calculated. These steps are explained in Sections 3.2.1 through 3.2.5. Capacity of devices and links are considered for finding assigned and alternative paths, but split flows are not allowed due to complexity of calculations.

$$Capacitated\ Resilience\ (U_i) = R(p_i) * Resilience\ Factor\ (U_i) \quad , i \in Users \tag{3.2}$$

The next sections explain the calculation steps of capacitated resilience (user level) components: (1) Finding the most reliable path between the user to an access point, (2) Identifying alternative paths between the user and any access point, (3) Determining independent subgroups of the alternative paths, (4) The reliability calculation of independent subgroups, and (5) Calculating reliability of the alternative paths (Resilience factor) using subgroup reliabilities.

A subgroup consists of a subset of alternative paths of a user. Obviously, a subgroup is a subgraph of the network. Therefore, subgroup and subgraph are used interchangeably in this dissertation.

3.2.1 Finding the most reliable path between the user to an access point

There might be more than one AP in a HetNet. For a given user, finding the most reliable path to connect an AP is important for service quality. The most reliable path from a user to an access point is found by Dijsktra's shortest path algorithm (Algorithm 3.1). It is a well known algorithm which labels every vertex v with its predecessor $p(v)$. In this algorithm, the distance from vertice r to v is defined as $l(v)$. More information can be found in Bondy and Murty (2008). As given in Equation 3.3, the algorithm works by minimizing the logarithm of link reliabilities which is equivalent to maximizing reliability of the path. Note that the reliability of the wireless link between user i and device j is calculated by $\max\{0, (r_j - d_{ij})/r_j\}$, where d_{ij} is the distance between user i and device j and r_j is the range of device j. Similarly, the reliability of the wireless link between device j and device k is calculated by $\max\{0, (r - d_{jk})/r\}$, where $r = \min\{r_j, r_k\}$.

Algorithm 3.1 Pseudocode of Dijkstra's shortest path algorithm (Bondy and Murty, 2008)

set $p(v) \leftarrow \emptyset$, $v \in V$, $l(r) \leftarrow 0$, $d(v) \leftarrow \infty$, $v \in V \setminus r$
while there is an uncolored vertex u with $l(u) < \infty$ **do**
  choose a vertex u with minimum $l(u)$
  color u black
  for $\forall$ uncolored neighbor v of u with $l(v) > l(u) + w(u, v)$ **do**
    replace $p(v)$ by u and $l(v)$ by $l(u) + w(u, v)$
  end for
end while
return (p, l)

$$\begin{aligned} \log R(P) &= \log \prod_{(i,j)\in P} p_{ij} \\ \log R(P) &= \sum_{(i,j)\in P} \log p_{ij} \end{aligned} \tag{3.3}$$

$R(P)$, reliability of a path of the user to an AP, is maximized when $\sum_{(i,j)\in P} \log(p_{ij})$ is maximized (equivalently, $-\sum_{(i,j)\in P} \log(p_{ij})$ is minimized), because $0 \leq p_{ij} \leq 1$, as given in Equation 3.4.

$$\max\{R(P) = \prod_{(i,j)\in P} p_{ij}\} \equiv \max\{\sum_{(i,j)\in P} \log(p_{ij})\} \equiv \min\{-\sum_{(i,j)\in P} \log(p_{ij})\} \tag{3.4}$$

After evaluating reliabilities from the user to all available APs (having enough capacity), the AP with the most reliable path is assigned to the user.

3.2.2 Identifying alternative paths between the user and any access point

Upon assigning the most reliable AP to a user, the next step is to identify all alternative paths from the user to all APs. This subproblem is computationally the most expensive part of the capacitated resilience calculation.

For each available AP, a k shortest path problem is solved to find the k most reliable paths from the user to that AP. Identifying all available paths might be intractable for large networks, therefore limiting the number of paths (k) reduces the complexity of this subproblem. Obviously, the solution becomes exact if k is sufficiently large.

The k shortest path problem is not new and there are many algorithms to solve it. Section 2.3 summarizes the k shortest path problem and its solution methods. Among them Yen (1971), Eppstein (1998), Katoh et al. (1978), and Katoh et al. (1982) are the most important ones. Katoh et al. (1982) is a generalization of Yen (1971). Eppstein (1998) is faster than Yen (1971) but it allows repeated vertices (which makes the search space larger). In this paper, Yen's (Yen, 1971) k shortest path algorithm has been used, because it provides an effective and easy implementation and allows only simple paths (no loops or repeated vertices).

Details of Yen's algorithm

The pseudo code of Yen's Algorithm to find k shortest paths is given in Algorithm 3.2 (the interested reader may refer to the original article (Yen, 1971) for more information):

Yen's algorithm starts with finding the most reliable path using any shortest path algorithm. In here, Dijkstra's shortest path algorithm (Algorithm 3.1) has been used because it is an exact algorithm and easy to implement. Note that solving the shortest path problem is equivalent to maximizing reliability of the path according to Equation 3.4. Upon identifying the shortest path between the user and the AP, the second shortest path is found by modifying the shortest path obtained in the first step. The main idea in this algorithm is to use previously obtained shortest paths to generate the remaining shortest paths. Therefore, the algorithm runs Dijkstra's shortest path algorithm from an intermediate node in the shortest path (called a spur) to the AP without using the rest of the edges in the shortest path. After finding the second shortest path the algorithm finds the rest of the k shortest paths, $(k-2)$

Algorithm 3.2 Pseudocode of Yen's k shortest path algorithm

1: **for** k=1 **do**
2: Step 1: Use Dijsktra to find shortest path from a fixed node to other nodes. The result will be A_1. This step assumes there are no negative loops in the shortest path.
3: Step 1a: Store A_1 into List A.
4: **end for**
5: **for** k=2 to K **do**
6: Step 2: Check if a node sequence $(1) - ... - i$ of A^{k-1} is same with first i nodes of previously generated path $j = 1, 2, ..., k-1$. Go to Step 3..
7: Step 3: Find the shortest path from i to N without including any nodes from $(1) - .. - i$ of A^j (which is called as R_i^k). Therefore, we are finding the shortest path of spur of A_i^k, which is S_i^k.
8: Step 3a: Add A_i^k (joins R_i^k and S_i^k) to candidate List B. Note that we need only $K - (k-1)$ many items in List B.
9: **if** Number of paths found at Step 3 + number of paths in List $A > K$ **then**
10: K Shortest paths found, save the paths to List A
11: Break
12: **else**
13: Step 4: Move path A^k from List B to List A.
14: Step 4a: Leave remaining items in B and k++. Repeat steps 2–4 until having K shortest paths.
15: **end if**
16: **end for**
17: **return** List A (k shortest paths)

paths in a similar fashion. To do this, new candidate shortest paths are found by using previously identified shortest paths. The algorithm ends when k shortest paths are found.

3.2.3 Determining independent subgroups of the alternative paths

Upon identifying all alternative paths from the user to all available APs, the next step is to group the alternative paths into independent subgroups. For any user having at least one alternative path (except the assigned path), there must be at least one independent subgroup of alternative paths. Each independent subgroup consists of paths having one or more common edges. Any path of an independent subgroup cannot have a common edge with a path of another independent subgroup. Independent subgroups are determined by a simple algorithm, which is summarized in Algorithm 3.3.

In this algorithm, all alternative paths are compared with each other to check for common edges. If so, one of the paths and all other paths in the same group are labeled as the other path's group. Therefore, the number of subgroups is dynamic in this algorithm.

Algorithm 3.3 Pseudocode of clustering alternative paths into independent groups

```
1: Save all alternative paths to List P
2: groupNo ← 1
3: for each path i ∈ P do
4:   if path i is not in a group then
5:     if i = 1 then
6:       Assign groupNo_i ← groupNo
7:       groupNo++
8:     end if
9:     for j = 1 to P do
10:      if i and j has a common edge then
11:        Assign groupNo_i to all paths of the group that j belongs to
12:      end if
13:    end for
14:  end if
15: end for
16: return Independent subgroups of alternative paths
```

3.2.4 Reliability calculation of independent subgroups

To find the capacitated resilience, reliabilities of the independent subgroups (that are identified in the previous step) must be calculated. However, minimum (or minimal) independent cuts must be found first to get reliability of each subgroup.

This problem is similar to *s-t* cut set problem, which is basically finding the minimum cut set between source and sink nodes. However, as there can be more than one AP in a subgroup, the problem is not the same as the *s-t* cut set problem. Another version of this problem is the multiterminal cut problem which is finding a set of edges that consist of non-terminal nodes to disconnect terminal nodes. Xiao (2010) and Hartvigsen (1998) stated that the multiterminal cut problem is NP-hard for $n \geq 3$, where n is the number of nodes in a graph. Again, this problem is not the same problem herein, because the terminal vertices, i.e. the user and APs, do not necessarily communicate with each other. In other words, APs do not communicate with each other because they only serve as a connection to the wired backbone. Also, unlike MANETs, users are not required to communicate with each other.

Thus, algorithms from the literature are not suitable for the capacitated resilience calculation. Therefore, an algorithm has been developed to find all minimum independent cut sets to disconnect the user from any AP with which user can communicate directly or indirectly (via RPs). One way to solve this problem is to enumerate all possible cut sets and then find the independent ones. The main drawback of this approach is the large computational effort for realistic sized networks. Although estimation methods can be used, such as Monte Carlo simulation (for example, Dengiz et al. (1997) and Deeter and Smith (1998) use Monte Carlo for all-terminal reliability estimation), an exact approach has been presented in this paper.

The pseudo code for finding minimum independent cut sets is given in Algorithm 3.4. This algorithm checks all combinations of edges beginning with one edge, two edges, three edges and so on. If any combination of the edges disconnects the user from all APs in the subgroup, then that combination of edges is a cut. If any combination of edges uses an edge

from a previously found cut set, that combination is not considered. The algorithm terminates when there are not enough edges left to form a unique combination or all combinations have been examined.

Algorithm 3.4 Pseudocode of finding minimum independent cutsets of a subgroup

```
1: Save all unique edges of alternative paths in the subgroup to set E
2: Cutsets ← ∅
3: for k = 1 to m (number of edges in E) do
4:    if |E| - number of unique edges in Cutsets < k then
5:       break
6:    end if
7:    for each unique combination of edges (C(m,k)) ∈ E do
8:       if C(m,k) ∩ Cutsets = ∅ then
9:          if removal of edges in C(m,k) disconnects user from all APs then
10:            Cutsets ← Edges in C(m,k)
11:         end if
12:      end if
13:   end for
14: end for
15: return Cutsets
```

Upon identifying all minimum independent cut sets of a subgroup, reliability of a subgroup is calculated. Failure of any cut set disconnects the user from APs. Therefore, all cut sets must be reliable to make the network reliable. Equation 3.5 presents the reliability calculation of a subgroup (S_i). In this formulation, C denotes the total number of combinations of edges that form cut sets. Similarly, $R(C_{ij})$ denotes the reliability of cut set combination C_{ij}. $R(C_{ij})$ is calculated by parallel reliability calculations (Equation 3.6). The reliability of cut C_{ij} requires that at least one edge (or serial edges from a RP to an AP), denoted as e in the Equation 3.6, to be operational.

$$\text{Reliability}(S_i) = \prod_{j=1}^{C} R(C_{ij}) \tag{3.5}$$

$$R(C_{ij}) = 1 - \prod_{e \in C_{ij}}^{C} (1 - r_e) \text{ , where } r_e \text{ is the reliability of edge } e \in C_{ij} \tag{3.6}$$

3.2.5 Calculating reliability of the alternative paths (resilience factor) using subgroup reliabilities

After calculating reliabilities of the independent subgroups, the last step is to calculate the reliability of the alternative paths (union of subgroups). This calculation is simply the parallel reliability calculation, where the system is reliable if at least one of the subgroups is reliable. Equation 3.7 shows this calculation where S denotes the number of independent subgroups and $R(S_i)$ denotes the reliability of subgroup i.

$$\text{Resilience Factor} = 1 - \prod_{i=1}^{S} [1 - R(S_i)] \tag{3.7}$$

3.3 An example network

In this section, calculations of capacitated resilience and the other reliability/survivability metrics are demonstrated on an example. The network presented in Figure 3.1 is used throughout this section. In this network, there are four APs (having enough capacities) and many rerouting options for a single user.

3.3.1 Capacitated resilience calculation

Capacitated resilience is calculated for both uncapacitated and capacitated cases. Although capacitated resilience considers capacities, uncapacitated case is important to show the effect of the capacity constraint because the other metrics neglect capacities.

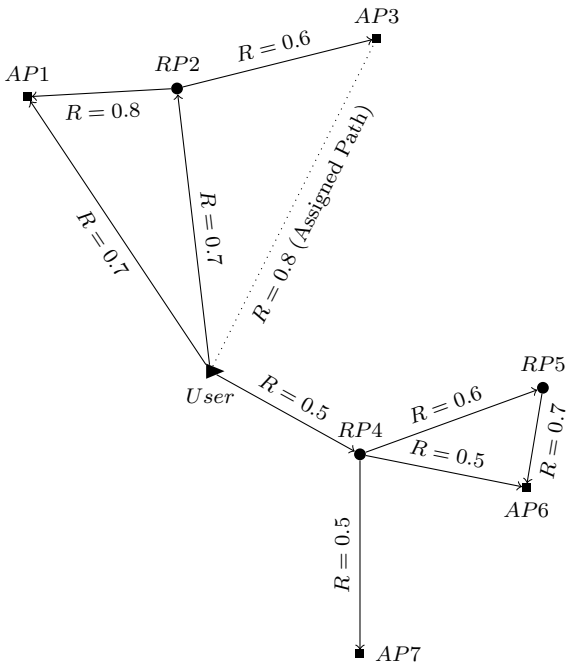


Figure 3.1: An example network

Uncapacitated case

Uncapacitated case considers that device capacities are larger than the user flow requirement and therefore the problem becomes uncapacitated. To calculate the capacitated resilience of this network, all possible alternative paths from the user to all available APs are found (where capacity is available) by Yen's k shortest path algorithm. In this dissertation, k is sufficiently large ($k = 7$) to find all shortest paths. Then, alternative paths from User to APs are grouped into independent groups. Lastly, reliability of the independent subgroups are calculated.

Independent subgroups 1, 2, and 3 are identified using the procedure given in Section 3.2.3.

The cut set of subgroup 1 (Figure 3.2) is $U - AP1$ and the reliability of the subgroup is 0.7.

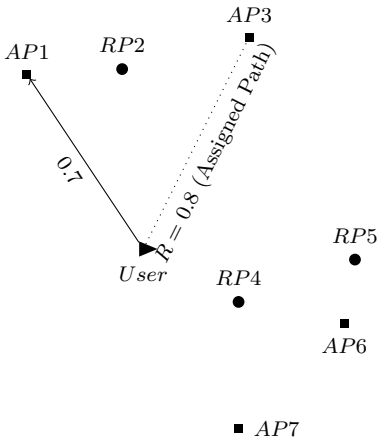


Figure 3.2: Independent subgroup 1 of the network in Figure 3.1

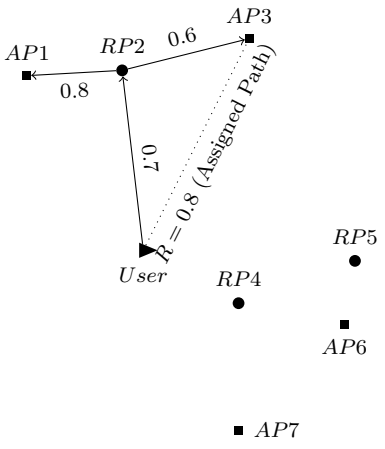


Figure 3.3: Independent subgroup 2 of the network in Figure 3.1

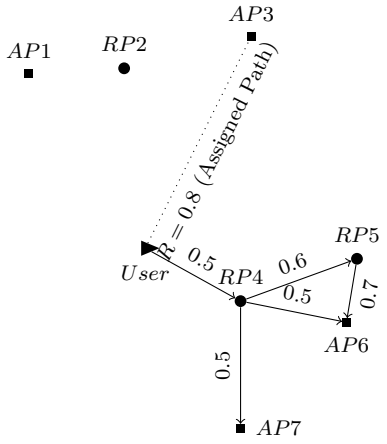


Figure 3.4: Independent subgroup 3 of the network in Figure 3.1

The cut set of subgroup 2 (Figure 3.3) consists of $(U - RP2)$ and $(RP2 - AP1,\ RP2 - AP3)$. The reliability of the subgroup is 0.644. The calculation steps are presented in Equation 3.8.

$$\begin{aligned} R(S2) &= R(U - RP_2)\ \ (R(RP_2 - AP_1, RP_2 - AP_3)) \\ &= 0.7(1 - (1 - 0.8)(1 - 0.6)) = 0.7(0.92) = 0.644 \end{aligned} \tag{3.8}$$

The cut set of subgroup 3 (Figure 3.4) consists of $(U - RP4)$, and $(RP4 - AP6,\ RP4 - AP7,\ RP4 - RP5$ or $RP5 - AP6)$. The reliability of the subgroup is 0.4275 (Equation 3.9).

$$\begin{aligned} R(S3) &= R(U - RP4)\ \ (R(RP4 - AP6, RP4 - AP7, RP4 - RP5 or RP4 - AP6)) \\ &= 0.5\left((1 - (1 - 0.5)(1 - 0.5)[1 - (0.6 * 0.7)])\right) = 0.5(0.855) \\ &= 0.4275 \end{aligned} \tag{3.9}$$

After calculating reliabilities of the subgroups, the reliability of the alternative path system (resilience factor) can be calculated by:

$$1 - (1 - 0.7)(1 - 0.644)(1 - 0.4275) = 0.938857$$

The resilience factor is the reliability of alternative path system (rerouting options). The alternative path system remains reliable if at least one subgroup is reliable. The capacitated resilience of the network has been defined in Equation 1.3. Therefore, the capacitated resilience (for the single user) can be calculated as:

$$Capacitated\ resilience = 0.8(0.938857) = 0.751086$$

Capacitated resilience for the entire network is calculated by the weighted average (in terms of flows) of user level capacitated resiliences. In this example there is only one user.

Capacitated case

For the capacitated case (Figure 3.5), the calculations of subgroups 1 and 2 remain unchanged. However, the reliability calculation of subgroup 3 is changed due to lack of capacity of AP6 and AP7. The flow of the user must be split between AP6 (traffic flow of 10) and AP7 (traffic flow of 10), therefore the subgroup can only be operational if the two split paths, U-RP4-AP7 and U-RP4-AP6 (or U-RP4-RP5-AP6), are both reliable. The calculation is given in Equation 3.10. The reliability of the subgroup 3 reduced from 0.4275 to 0.1775. The system reliability becomes 0.912157 (Equation 3.11). The capacitated resilience is calculated as 0.72973 (Equation 3.12). The reduction in the capacitated resilience is not very high (from 0.7511 to 0.7297) because there are three alternative path systems in this example. If the subgroup 3 was the only alternative path system, the reduction in the capacitated resilience due to limited capacity would be more significant, i.e., from $0.8 * 0.4275 = 0.342$ to $0.8 * 0.1775 = 0.142$.

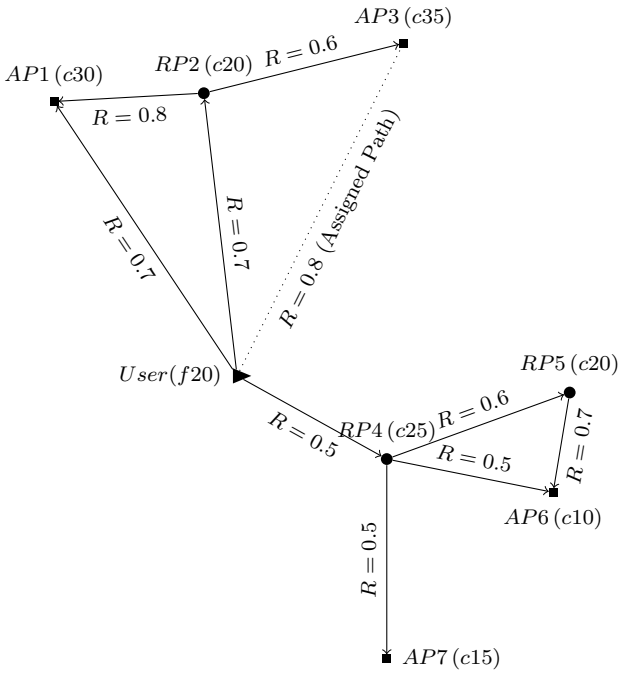


Figure 3.5: Capacitated version of the example in Figure 3.1 (c and f denote device capacity and user flow requirement, respectively

$$
\begin{aligned}
R(S3) &= R(U - RP4)\ \ (R(RP4 - AP6, RP4 - AP7, RP4 - RP5 or RP4 - AP6)) \\
&= 0.5\,(0.5(1 - [1 - 06(0.7)][1 - 0.5])) = 0.5(0.355) \\
&= 0.1775
\end{aligned}
\tag{3.10}
$$

$$1 - (1 - 0.7)(1 - 0.644)(1 - 0.1775) = 0.912157 \tag{3.11}$$

$$Capacitated\ resilience = 0.8(0.912157) = 0.72973 \tag{3.12}$$

The next sections demonstrate the calculation of the other reliability/survivability metrics. As explained in Section 2.2, capacity is not considered for those metrics.

3.3.2 Connectivity measures

The connectivity measures are also investigated for the same example (Figure 3.1). Two-connectivity (two node-disjoint paths) is assured by the paths U-AP3 and U-RP2-AP3. Similarly, the same set of paths (U-AP3 and U-RP2-AP3) ensures two edge-disjoint paths. Therefore, this network is both two node-connected and two edge-connected.

3.3.3 Two-terminal and all-terminal reliability

Similar to the connectivity measures, two-terminal reliability is calculated for U-AP3. The paths leading to AP3 from the user are U-AP3 and U-RP2-AP3. Therefore, at least one of the paths must be operational to reach AP3. Two-terminal reliability becomes $1 - (1 - 0.8)\,(1 - 0.6(0.7)) = 0.844$.

All-terminal reliability considers all paths from the user to all available APs. Therefore, the parallel reliability calculation of the paths from user to AP1, AP3, AP6 and AP7 yields all-terminal reliability: $1-(1-0.8)(1-0.42)(1-0.7)(1-0.56)(1-0.25)(1-0.25)(1-0.21) = 0.993196$. Not surprisingly, all-terminal reliability is larger than two-terminal reliability because it considers all possible APs whereas two-terminal reliability considers only AP3.

3.3.4 Traffic efficiency

Traffic Efficiency/Resilience (Konak and Bartolacci, 2007; Kubat, 1989) is calculated by both exact method and simulation. The exact method yields 0.9878, whereas simulation (using 5000 iterations, according to Algorithm 4.6 in Section 4.3.2) yields 0.9902.

3.3.5 ELT

The ELT calculation requires the input of the splits of the flow. Assuming the user traffic is splitted equally to U-AP3 and U-AP1, the ELT is calculated by (according to Section 2.2.3):

$$
\begin{aligned}
ELT &= \left[(1 - R(U - AP1))(0.5) + (1 - R(U - AP3))(0.5)\right](M_0) \\
&= \left[(1 - 0.7)(0.5) + (1 - 0.8)(0.5)\right] 5.26(10^5) \\
&= 0.25(5.26(10^5)) = 131,500 \text{ demand-minutes/year}
\end{aligned}
$$

The value of the constant M_0 is $5.26(10^5)$ to convert the fraction of lost flow to demand-minutes per year (Grover, 2004). However, the resulting number is not scaled, it is the expected number of lost demand-minutes over a year. Considering the fraction of the flow that cannot be routed is 0.25, then a conversion of ELT can be done to obtain a scaled measure (like reliability), i.e., ELT becomes 0.75 $(1 - [(1 - 0.7)(0.5) + (1 - 0.8)(0.5)])$.

3.3.6 Summary

The procedures for finding the above listed reliability/survivability measures are summarized in detail in Section 4.3. Each metric presents a different way to evaluate the network. Capacitated resilience emphasizes the importance of rerouting options in case of failure of the assigned path. Therefore, it considers the redundancy level of the network under capacity constraints. Connectivity metrics focus on redundant paths, but they do not consider their reliability. Two terminal and all terminal reliabilities are calculated by parallel reliability calculations and they focus on the redundancy in the network. In the example given in Figure 3.1, terminal reliability values are high, however, these values can be misleading because these metrics do not consider the common links that are used by many paths. In case of a failure of these links, the redundancy of the network can be reduced or lost. Capacitated resilience considers common links and their impact on reliability of redundant paths in its subgroup and cut set calculations. Traffic efficiency also indirectly considers common links in the reliability calculation. However, it does not consider capacity. As explained in Section 6.3, all these differences result in different network designs.

Chapter 4

Proposed Method and Solution Approach

The network design problem, even without survivability/reliability constraints (or objectives), is a well known NP-hard problem. In this dissertation, an Evolutionary Strategies (ES) algorithm has been developed to solve this problem. Both single objective and bi-objective models are considered in the proposed ES.

The main reasons to use ES for solving the resilient heterogeneous wireless network design problem are summarized below:

- The capacitated resilience calculation is nonlinear and traditional optimization methods (such as mixed integer programming) cannot be applied effectively. And for computational reasons, for large problem instances, an alternative methodology other than traditional exact optimization methods is needed to solve the problem. This is the main motivation for using a metaheuristic such as ES.

- ES is known for its success on continuous problems. The problem solved in this dissertation has device coordinates as continuous decision variables as well as some discrete variables (e.g., device types). Also, ES is a flexible method that can handle many restrictions and assumptions.

- Bi-objective optimization is one of the main contributions of this research. ES can easily be extended to bi-objective optimization using "Pareto optimality" (defined in Section 4.2.1). Here, bi-objective optimization is ensured by "non-domination ranking" (see Section 4.2.1) as used in NSGA-II (Deb et al., 2002). NSGA-II is a multiobjective Genetic Algorithm (GA) technique, however, its main idea has been adapted to ES in this dissertation.

There are some drawbacks of using ES. First of all, it does not guarantee optimality. Second, choosing a good representation (encoding) of problem and finding efficient mutation operators are generally challenging issues. Third, parameter tuning is important for better solution quality; for example, premature convergence can be observed due to certain parameter settings. These issues have been addressed in this dissertation.

4.1 Single objective Evolutionary Strategies (ES) model

Cost and capacitated resilience are two objective functions that are considered in this dissertation. The single objective ES uses cost (capacitated resilience) as the objective function and capacitated resilience (budget) as a constraint. The parameters of the ES model are given in Table 4.1. Budget is important because it determines the maximum number of devices in the network. As there are two types of devices (AP and RP), the number of devices and their types change the cost. As explained in Section 4.1.9, the cost and the capacitated resilience of a solution are penalized if the solution is infeasible. A solution becomes infeasible if any user is not assigned to a device or a device does not have a feasible route.

Table 4.1: Input parameters used in the ES model

Parameter	Description
# of users and their locations	Locations as (x, y) coordinates
flow of the users	Traffic requirement of each user
gridX and gridY	The limits of the area in which nodes are deployed
budgetLimit	Budget limit for installing devices
maxDevice	Maximum number of devices
Cost of devices	Cost to deploy a relay or an access point
Ranges of devices	Ranges of relay or access points
Capacity of devices	Capacities of relay or access points

The decision variables of the ES model are listed in Table 4.2. The objective function of the ES model is defined in detail in Section 4.1.5.

Table 4.2: Decision variables used in the ES model

Decision Variable	Description
# of devices	Number of devices used in the solution ($\leq$ MaxDevice)
Locations of devices	Defined as (x, y) coordinates
Mixture of type of devices	Either router or access point
User to device assignments	Each user i is assigned to a device j
Routing of the flow	From a router to an access point the route of user traffic
Reliability among devices	It depends on the locations of devices and routing
Reliability among users and devices	It depends on the locations of users and devices

4.1.1 Pseudocode of ES model

The pseudocode of the single objective ES model is given in Algorithm 4.1. This ES model maximizes capacitated resilience (minimizes cost) under a budget (capacitated resilience) constraint.

At each generation, a parent is selected from the population randomly and saved as a child solution. Then, mutation and swap operators are applied to the child. Mutation and swap operators are explained in Section 4.1.6. In the proposed ES, the device types of a solution are changed every ten generations because finding good coordinates for a given set of devices (mixture of APs and RPs) is very hard. Therefore, 10 generations give ES enough time to adjust device coordinates for a given set of devices. Children are replaced with population members according to $(\mu + \sigma)$ rule (see Section 4.1.7 for more details). The best solution is saved throughout the generations.

In the proposed ES algorithm, there are ES specific operations. First of all, mutation success is calculated for a predetermined number of generations (g'). A successful mutation means that the child solution (mutated solution) has a better objective function values than its parent (lower cost or higher resilience, according to the objective function). Note that the success of a mutation is defined differently in bi-objective optimization (see Section 4.2 for details). According to the "rate of successful mutations", σ is dynamically adjusted at n generations. If the percentage of successful mutations is less than the rate of successful mutations, the search space is narrowed by 0.85. In this dissertation, rate of successful

mutations is selected as 0.2, which is very common in ES literature and known as the "one fifth rule" (Dréo et al., 2006). σ is the mutation rate for coordinates and is explained in Section 4.1.6 in detail. At the beginning of the ES, σ_x and σ_y are initialized to one third of the grid dimensions (gridX and gridY of Table 4.1). The algorithm terminates if the maximum number of generations is reached or the best solution has not updated for a given number of generations (Section 4.1.8). This ES algorithm yields the best network design (does not guarantee optimality), the lowest cost or the highest resilience, according to the objective function.

Parameters for running the ES model are given in Table 4.3.

Table 4.3: Model parameters used in the ES model

Parameter	Description
N_p	Population size
N_c	Number of children
P_m	Probability to mutate
maxGen	Maximum number of generations
maxNonImprovingGen	Maximum number of non-improving generations
numberOfAlternativePaths	Number of alternative paths in capacitated resilience calculation
cutSetSize	The maximum number of edges in a cut set
successRate	Also known as one fifth rule. It is a parameter to adjust σ in ES
sigmaAdjuster	Adjustment rate of σ in ES
numberOfMutationsToAdjustSigma	It is used to estimate successful mutations in ES
penaltyUnassignedUser	Penalty value if a user is not assigned to a device
penaltyInfeasibleRouting	Penalty value if a device cannot connect to the wired backbone

4.1.2 Encoding

An adjacency matrix is used to represent the network structure. This matrix is n x n, where n is the maximum number of devices. Different from a traditional adjacency matrix, reliabilities between devices are used instead of 0/1 values in the matrix. In other words, if devices i and j can communicate with each other (both devices are within the communication ranges of each other), then the entry ij of the adjacency matrix is the reliability between devices i and j. Also, device types and (x, y) coordinates of the devices are stored in an

Algorithm 4.1 Pseudocode of the single objective ES model

Ensure: Maximum budget (minimum capacitated resilience) constraint, assigning all users to a device, feasible (defined in Section 4.1.9) routings

Random initialization of population (Section 4.1.3)
Sort population (from the best solution to the worst)
bestSolutionSoFar $\leftarrow$ Population(0)
$g \leftarrow 0$ //g is the generation counter
while (g $<$ maxGen) **do**
  for ($i = 0$ to $N_c - 1$) **do**
    Randomly select a parent (i) from population to mutate, $i \in \{0, 1, ..., (N_p - 1)\}$
    Mutate device coordinates of Child[i]
    Select two random devices j and k of Child[i]
    Swap device types of j and k within Child[i]
    if ($g\%10 = 0$) **then**
      Mutate device types of Child[i]
    end if
  end for
  Sort population and children (from the best solution to the worst)
  Replace worst population members with new generated children
  Sort population (from the best solution to the worst)
  if (Population[0] is better than bestSolutionSoFar) **then**
    bestSolutionSoFar $\leftarrow$ Population[0]
  end if
  if ($g\%g' = 0$) **then**
    if (Mutation success rate > 0.20) **then**
      Increase σ_x and σ_y to 1/0.85 of their values
    else
      Decrease σ_x and σ_y to 0.85 of their values
    end if
  end if
  if (bestSolutionSoFar has not been updated for maxNonImprovingGen generations) **then**
    Terminate ES
  end if
  g $\leftarrow$ g+1
end while
return Network design with maximum capacitated resilience (or minimum cost)

array with the size of the maximum number of devices. Users to device assignments are stored in a separate array with the size of the number of users.

4.1.3 Population initialization procedure

The population is randomly initialized using a different random number seed than the one used in the ES. In the initialization procedure, only device types and device coordinates (in continuous space) are created. Users are assigned to the closest devices and a routing algorithm (Section 4.1.4) is applied to route user flows. Cost, reliability and capacitated resilience values are calculated according to the equations of Section 4.1.5.

4.1.4 Edge selection algorithm

The edge selection algorithm is basically the routing of flow from device (or user) i to j. Device capacities are checked by using the adjacency matrix, and the best path from user i to AP_j is determined by using Dijkstra's shortest path algorithm. Dijkstra's algorithm minimizes the total cost of the links of a path, however, in this routing problem cost is meaningless because wireless links do not incur costs. Instead, the logarithms of link reliabilities are used as "cost" in Dijkstra's algorithm. The algorithm maximizes the reliability between user i and an access point. This procedure was explained in Equation 3.4. Thus, an access point with the most reliable path is selected using Dijkstra's algorithm. If the device is already an access point, Dijkstra's algorithm is not applied, because it is already connected to the wired backbone.

The main issue of this algorithm is the capacity of devices. Users are sequentially assigned to the devices and routed over the devices. In other words, Dijkstra's algorithm is applied to each user in a randomized order. Therefore, depending on the order of assigning flows to devices a better path may become infeasible later in the algorithm (due to lack of capacity of a device). It is possible that a user's flow cannot be routed due to insufficient capacity of devices. Also, the ES may yield non-optimal results due to this ordering issue.

On the other hand, this algorithm works satisfactorily for problem instances in which the capacities of devices are not highly constrained. Although the effect of ordering on the solution quality has not been observed in the preliminary testing of the ES, this issue will be investigated as a future work of this dissertation.

4.1.5 Calculation of cost, reliability and capacitated resilience

The cost calculation is the summation of the deployment costs of devices (Equation 4.1). Also, penalty values are added to the cost function. As given in Table 4.3, two different penalties are defined in this ES model. A penalty is added to the cost if a user is not assigned to a device and another penalty is added if a device does not have a feasible path to an AP. Section 4.1.9 discusses these penalties in detail.

$$\begin{aligned} &\text{Cost} = \sum_{i \in Devices} c_i * x_i \\ &\text{where } c_i \text{ is the cost of device } i, \text{ and} \\ &x_i = 1, \text{ if the device } i \text{ is included in the solution} \end{aligned} \tag{4.1}$$

The capacitated resilience calculation is given in Equation 1.4 (in Section 1.3). According to this equation, the weighted average (in terms of flows) of all resilience values for each device is calculated to find capacitated network resilience. A penalty is applied to capacitated resilience if the budget is exceeded. Capacitated resilience is also penalized if there are unassigned users or infeasible device paths. These penalties are discussed in Section 4.1.9 in detail.

The reliability of a network is calculated similarly to the capacitated resilience calculation. The reliabilities of assigned paths of all users are combined using the weighted average of user flows. Therefore, the network reliability is the weighted average of the user reliabilities.

4.1.6 Mutation operators

The first mutation is to alter device coordinates. In this mutation, the coordinates of a device are changed by $N(0, \sigma_x)$ and $N(0, \sigma_y)$ for x and y axes, respectively. The (x, y) coordinates of devices are restricted to be inside the predefined grid (given in Table 4.3). This is also known as "self-adaptive Gaussian" mutation (Dréo et al., 2006) because the mutation rates of coordinates, σ_x and σ_y, are dynamically adjusted during the ES. The search is widened by increasing σ parameters by $1/0.85$ if the percentage of successful mutations is larger than 0.2 ("one fifth rule"). Otherwise, the search is focused on a smaller region by decreasing σ values by 0.85. This decision is made every n generations.

A swap of device types is the second type of mutation. In this mutation, also known as "2-opt swap", the two devices to swap are randomly selected and device coordinates are not changed. To perform the swap operator, the devices are not required to be in the solution. This encourages the optimizer to search more diverse candidate solutions. Also, it works faster since the devices are not checked whether they are in the solution or not.

The last mutation is to change a device type of a child using a variation of bit-flip mutation (Dréo et al., 2006). In this mutation, an RP (AP) device changed to an AP (RP) if a random number is less than the mutation probability (given in Table 4.3). Similarly, the status of the device is changed by this mutation operator. For example, a device is included in the solution if another random number is less than the mutation probability and the device was not included in the solution prior to the mutation. If a previously not included device is included in the solution, the coordinates of the device are assigned using an algorithm instead of random assignment. In this algorithm, the number of devices in each quadrant are calculated and the device coordinates are randomly assigned within the quadrant having the least amount of devices. There are four quadrants (I, II, III and IV) in the xy plane, the first quadrant is the area where $x > 0$ and $y > 0$, the second is the area where $x < 0$ and $y > 0$, the quadrant is the area where $x < 0$ and $y < 0$ and the last is the area where $x > 0$

and $y < 0$. A slight increase in the solution quality has been observed with using this device coordinate assignment instead of random coordinate assignment.

The coordinate change and swap mutations are done at each generation. However, device type change mutation is done at every 10 generations to give the ES enough time to optimize device coordinates as explained in Section 4.1.1. The swap mutation is done at each iteration instead of doing it infrequently because the probability of changing a device type (from AP to RP or RP to AP) is not very high and therefore the effect of swap on the solution quality is not very large. Since only two devices are swapped and there are two device types (AP and RP), the probability of getting a different device type after the swap operator is 0.5 if there are equal amount of APs and RPs in the solution. Also, only two devices in a solution are swapped, therefore the net effect of the change (if there is any) on the solution quality is very limited. In fact, the initial experimentation of the model showed that swapping at each iteration outperformed swapping infrequently. Obviously, the main mutation operator is the coordinate change. Note that the original solution, which was mutated and saved as the child solution, stays in the population as the parent solution.

After mutations, users are assigned to the devices having the most reliable path to an AP using the routing algorithm. Cost, reliability and capacitated resilience values are calculated.

Budget is not considered in mutation operators. However, a penalty of exceeding the budget is included in the capacitated resilience calculation (see Section 4.1.9 for details), which is applied after mutation. The reason of not including budget control in mutations is to encourage the optimizer to find more diversified solutions even though this may temporarily create some infeasible solutions.

4.1.7 Replacement of population with children

After generating children, population and children solutions are combined and the best solutions are selected for the next generation of population. This is also known as $(\lambda + \mu)$ replacement in ES literature.

As an elitist approach, the "best so far" solution is stored and updated throughout generations. At each generation, the worst individual of the population is replaced with the "best so far" solution if the "best so far" is better than the current best individual of the population. The reason of this approach is to maintain the "best so far" solution in the population at all times.

4.1.8 Stopping criteria

The stopping criteria are the total number of generations and the number of non-improved generations. The number of non-improved generations criterion prevents the ES running longer without improving solution quality. The key is to select the number of non-improved generations is to balance early termination prevention and avoid running excessively.

4.1.9 Penalty functions

Three main penalties are used in the proposed ES: (1) Unassigned users, (2) Infeasible device routes, and (3) Budget overruns. These penalties change cost (penalties 1 and 2) and capacitated resilience (penalties 1, 2 and 3) to ensure that infeasible solutions do not proliferate in the population.

The cost of a solution is penalized if a user is not assigned to a device. A user may not be assigned to a device due to lack of capacity of the device, being beyond the ranges of devices, or lack of connection of device to an AP. Another penalty is applied to the cost when the routing of a device is infeasible. Infeasible routing occurs when an RP cannot connect to an AP. There is no routing needed for APs, therefore this penalty does not apply to APs.

Each penalty is a "death penalty" (Dréo et al., 2006), in other words, these penalty values are very large to prevent reproduction of infeasible solutions. These penalties are defined in Table 4.3 for each user and device. An additional cost penalty is not applied for exceeding the budget limit, but capacitated resilience is penalized.

Capacitated resilience is scaled by percentage of assigned users. For example, if one out of ten users is not assigned to a device, the capacitated resilience value is scaled by 0.9 (= 9/10). The number of devices with infeasible paths is incorporated in the capacitated resilience calculation in a similar way. The percentage of devices with feasible paths scales capacitated resilience. In addition to these two penalties, capacitated resilience is further penalized by the cost of the solution. If the cost of the devices (not including penalties) is greater than the budget, capacitated resilience is scaled by "budget/cost of devices". This dynamically penalizes capacitated resilience for exceeding the number of devices and forces the ES to reduce the total number of devices or number of APs (either by removing them or replacing them with RPs).

The details of the penalty functions are given in Algorithm 4.2.

4.2 Bi-objective Evolutionary Strategies (ES) model

The bi-objective ES simultaneously optimizes capacitated resilience and cost using Pareto optimality. The parameters and decision variables of the bi-objective ES model are same as in the single objective model (Tables 4.1, 4.2 and 4.3).

4.2.1 Definitions: "Non-dominated rank", "Crowding distance" and "Pareto optimality"

In bi-objective optimization of resilient heterogeneous wireless networks, solution i dominates solution j only if its cost (capacitated resilience) is lower (higher) than j and capacitated resilience (cost) is not lower (higher) than j (Equation 4.2).

Algorithm 4.2 Pseudocode of penalty functions

Ensure: Penalize cost and capacitated resilience of solution[i] (if the conditions are met)

Define n_{UU} and n_{ID} as the number of unassigned users and the number of infeasible devices, respectively
Define $Cost_{Devices}$ as the deployment cost of devices in solution[i]
Set $n_{UU} \leftarrow 0$, $n_{ID} \leftarrow 0$ and $Cost_{Devices} \leftarrow 0$
Calculate solution[i].cost and solution[i].CR // CR denotes Capacitated Resilience
for ($\forall$ user[j] in solution[i]) **do**
  if (user[j] is not assigned to a device) **then**
    $n_{UU} \leftarrow n_{UU} + 1$
  end if
end for
if ($n_{UU} > 0$) **then**
  solution[i].cost $\leftarrow$ solution[i].cost $+ n_{UU} * penaltyUnassignedUser$
  solution[i].CR $\leftarrow$ solution[i].CR $* \frac{\text{Total number of users in solution}[i] - n_{UU}}{\text{Total number of users in solution}[i]}$
end if
for ($\forall$ device[j] in solution[i]) **do**
  if (device[j] is not connected to an AP (has an infeasible route)) **then**
    $n_{ID} \leftarrow n_{ID} + 1$
  end if
end for
if ($n_{ID} > 0$) **then**
  solution[i].cost $\leftarrow$ solution[i].cost $+ n_{ID} * penaltyInfeasibleRouting$
  solution[i].CR $\leftarrow$ solution[i].CR $* \frac{\text{Total number of device in solution}[i] - n_{ID}}{\text{Total number of device in solution}[i]}$
end if
for ($\forall$ device[j] in solution[i]) **do**
  $Cost_{Devices} \leftarrow Cost_{Devices} + device[j].cost$
end for
if ($Cost_{Devices} > budgetLimit$) **then**
  solution[i].CR $\leftarrow$ solution[i].CR $* \frac{budgetLimit}{Cost_{Devices}}$
end if

Solution i dominates j if

$$\begin{aligned} &Cost_i \leq Cost_j \text{ and } Cap.\ Resilience_i > Cap.\ Resilience_j \text{ , or} \\ &Cost_i < Cost_j \text{ and } Cap.\ Resilience_i \geq Cap.\ Resilience_j \end{aligned} \tag{4.2}$$

Deb et al. (2002) defined "non-dominated ranking" and "crowding distance" to ensure population diversity for their multiobjective GA. A "non-dominated rank" of a population is a set of non-dominated solutions. In other words, a solution does not dominate another within a rank. For a population having more than one solution, there is at least one non-dominated rank which is called the first rank. There can also be other ranks in a population; i.e., second, third and so on. A solution of a lower non-dominated rank (e.g., first rank) dominates the solutions of higher ranks (e.g., second rank). The solutions in the first rank are "Pareto optimal" because they are not dominated by any solution in the population.

"Crowding distance", as first defined by Deb et al. (2002), measures "uniqueness" of a solution in objective function space. It is defined for solutions within the same non-dominated rank. In this dissertation the crowding distance is calculated according to the algorithm proposed by Deb et al. (2002). This algorithm (Algorithm 4.3) deals with each non-dominated rank separately. Within each rank, the crowding distances of solutions are calculated for each objective function. For any objective function, the crowding distance of a solution becomes infinity (or a very large number) if its objective function value is the minimum (or maximum) of its rank. For all other solutions crowding distance is calculated by adding the difference of objective function values of two neighboring solutions ($i+1$ and $i-1$). Therefore, a smaller crowding distance means that the objective function value of solution i is very close to its neighbor solutions $(i+1)$ and $(i-1)$. Note that, the neighboring solutions of a solution are defined for each objective function and they can be different for each objective. In Figure 4.1, the crowding distance values of solutions 1 and n (where $n = |r|$) are infinity. As given in Algorithm 4.3, the crowding distance is normalized for each

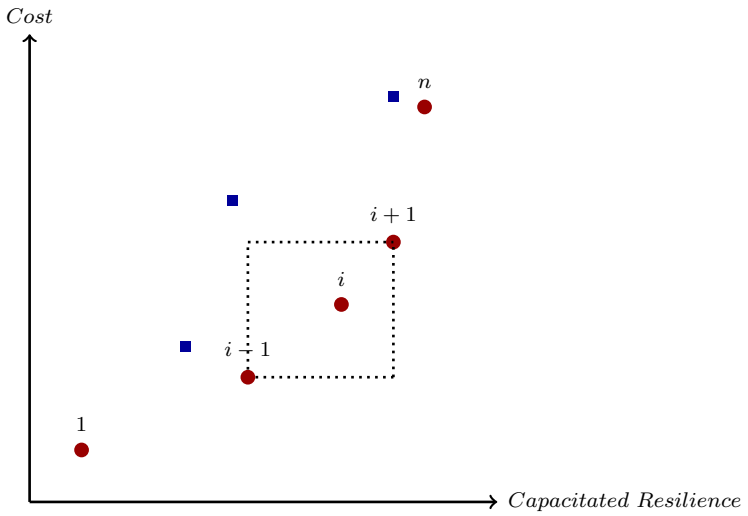


Figure 4.1: Crowding distance calculation

objective function m by $(f_m^{max} - f_m^{min})$. Although the closeness of the objective function values of two solutions does not guarantee their similarity in solution space, initial experimentation showed that two solutions are very similar in solution space if their objective function values (cost and capacitated resilience) are close to each other.

Algorithm 4.3 Pseudocode of crowding distance calculation (Deb et al., 2002)

for ($r = 1$ to total number of non-dominated ranks) **do**
  for (each objective m (cost or capacitated resilience)) **do**
    $Solution(i).distance \leftarrow 0$, $\forall i \in$ Rank r
    Sort solutions within the rank according to objective m
    $n \leftarrow$ number of solutions in Rank r, $Solution(1) \leftarrow \infty$, $Solution(n) \leftarrow \infty$
    $f_m^{max} \leftarrow$ Maximum value of objective function m within rank r
    $f_m^{min} \leftarrow$ Minimum value of objective function m within rank r
    for ($i = 2$ to $(n-1)$) **do**
      $Solution(i).distance = Solution(i).distance + \frac{Solution(i+1).m - Solution(i-1).m}{f_m^{max} - f_m^{min}}$
    end for
  end for
end for

4.2.2 Pseudocode of ES model

The pseudocode of the bi-objective ES model is given in Algorithm 4.4. The algorithm is exactly the same as the single objective ES except that the sorting is different and Pareto

optimality is used. The solutions within a population are sorted according to non-dominated ranks and then the crowding distances within the rank (see Section 4.2.4 for details). This idea of sorting was first introduced by Deb et al. (2002) for a multiobjective Genetic Algorithm (GA). Each rank has non-dominated solutions. However, by definition, the solutions of the second rank are dominated by the solutions of the first rank. The first rank is also called the "Pareto Front" which includes the best solutions of the population. Non-dominated rank and crowding distance are used to keep population diversified for bi-objective optimization.

4.2.3 Global Pareto front operations

The proposed bi-objective ES keeps a "global" Pareto front throughout the search. The global Pareto front is different than the current Pareto front of the population. The curent Pareto front consists of solutions of the current population, wheras the global Pareto front includes solutions from any population generated during the ES search. In other words, the global Pareto front keeps the best solutions that have been found so far. Therefore, the global Pareto front solutions may be different than the Pareto front of the current population. A solution may be included in the current Pareto front but it may also be dominated by some solutions of the global Pareto front. At the first generation the global Pareto front is empty and at the end of the last generation there is at least one non-dominated solution in the Pareto front. At each generation the population members are modified by mutation operators. Therefore, new solutions are generated and they are compared with existing global Pareto front solutions. If a new solution is better than an existing Pareto front solution, the solution becomes a member of the Pareto front and the dominated solutions of the Pareto front are discarded. Thus, the size of the global Pareto front is dynamic.

Pareto front diversity

Global Pareto front members do not necessarily exist in the population. They can be lost during the iterations due to mutation operators. To increase variability within the

Algorithm 4.4 Pseudocode of the bi-objective ES model

Ensure: Maximum budget (minimum capacitated resilience) constraint, assigning all users to a device, feasible routings

Initialization of population
Calculate non-domination rankings and crowding distances of solutions
Sort population (according to non-domination rankings and crowding distances, explained in Section 4.2.4)
bestSolutionSoFar $\leftarrow$ Population(0)
g $\leftarrow$ 0
while (g $<$ maxGen) **do**
  for ($i = 0$ to $N_c - 1$) **do**
    Randomly select a parent (i) from population to mutate, $i \in \{0, 1, ..., (N_p - 1)\}$
    Mutate device coordinates of Children[i]
    Select two random devices j and k of Children[i]
    Swap device types of j and k within Children[i]
    if ($g\%10 = 0$) **then**
      Mutate device types of Children[i]
    end if
  end for
  Sort population and children (according to non-domination rankings and crowding distances)
  Replace worst population members with new generated children
  if (If the solution Population[i] is better than bestSolutionSoFar, $i \in (0, 1, .., N_p)$ **then**
    bestSolutionSoFar $\leftarrow$ Population[i]
  end if
  Update Pareto front with new solutions (Add new non-dominated solutions from Population and remove dominated solutions)
  Add some Pareto front members to the population (see Algorithm 4.5)
  if ($g\%n = 0$) **then**
    if (Mutation success rate $>$ 0.20) **then**
      Increase σ_x and σ_y to 1/0.85 of their values
    else
      Decrease σ_x and σ_y to 0.85 of their values
    end if
  end if
  if (bestSolutionSoFar has not been updated for maxNonImprovingGen generations) **then**
    Terminate ES
  end if
  g $\leftarrow$ g+1
end while
return Network design with maximum capacitated resilience (minimum cost)

population and increase the solution quality of future generations some inferior population members are replaced with some non-dominated solutions from the global Pareto front. Details of this process are given in Algorithm 4.5. In this algorithm, the population is sorted for increasing crowding distance. A larger crowding distance is preferred because it shows the "uniqueness" of a solution. Therefore, sorting the population according to the crowding distance is equivalent to sorting it from the least unique solution to the most unique. To have a more diversified Pareto front and a better solution quality during the search, the population is preferred to have a larger number of "unique" solutions. The proposed algorithm compares each solution in the Pareto front with all members of the population and the number of similar solutions in the population is saved for each Pareto front solution. A population solution is determined to be similar to the Pareto front solution if its cost and capacitated resilience values are within a predetermined percentage of the Pareto front solution's cost and capacitated resilience. If a Pareto front solution has a smaller number of similar solutions than a given threshold, that solution is replaced with the population member having the smallest crowding distance value. Other Pareto front solutions are added to the population in a similar way. This process ends if there is no Pareto front solution qualifying for the given conditions or a predetermined number ($countMax$ parameter in the algorithm) of Pareto front members are added to the population. The value of $countMax$ has been selected as four after initial testing of the ES. Thus, at most four least unique solutions in the population (size of the population is selected as 30) are replaced with the most unique global Pareto front solutions.

4.2.4 Multiobjective sorting

Deb et al. (2002) defined "partial order" ($\prec_n$) where Solution(i) $\prec_n$ Solution(j) means that Solution(i) is better (or more preferable) than Solution(j). A solution having a lower non-dominated rank is better than the other. Within the same rank, solutions with larger

Algorithm 4.5 Pseudocode of adding Pareto front solutions to the population

```
1: n ← number of solutions in Pareto front
2: minNumber ← N_p/10, countMax ← N_p/5, counter ← 0
3: Initialize array numberOfSolutionsWithinRange with size of n
4: Sort Population for increasing order of crowding distance
5: for (i = 0 to n − 1 ) do
6:    for (j = 0 to N_p − 1 ) do
7:       if (Cost_i ∗ (1 − rangePercentage) < Cost_j < Cost_i ∗ (1 + rangePercentage) and
         CR_i ∗ (1 − rangePercentage) < CR_j < CR_i ∗ (1 + rangePercentage)) then
8:          numberOfSolutionsWithinRange[i] ← numberOfSolutionsWithinRange[i] ++
9:       end if
10:   end for
11:   for (i = 0 to n − 1 ) do
12:      if (numberOfSolutionsWithinRange[i] < minNumber) then
13:         Population[counter] ← ParetoFrontSolution[i]
14:         counter ← counter + 1
15:      end if
16:      if (counter ≥ countMax) then
17:         break
18:      end if
19:   end for
20:   Sort Population (multiobjective sort)
21: end for
```

"crowding distance" values are more preferable than ones with lower crowding distance values. Equation 4.3 formally defines this preference rule between solutions.

$$\begin{aligned} &Solution(i) \prec_n Solution(j) \text{ if} \\ &Solution(i)_{rank} < Solution(j)_{rank} \\ &\text{or} \\ &Solution(i)_{rank} = Solution(j)_{rank} \text{ and } Solution(i)_{Crowd.\ Dist.} > Solution(j)_{Crowd.\ Dist.} \end{aligned} \tag{4.3}$$

This partial ordering process is used for the multiobjective sorting in this dissertation. The solutions are first sorted according to increasing order of non-dominated ranks. Then, the solutions within the same rank are sorted according to the decreasing order of crowding distance values.

4.3 Calculation of other survivability and reliability metrics

In this dissertation the proposed metric, capacitated resilience, is compared with the metrics explained in Section 2.2. The calculation steps of these metrics are explained in the following sections.

4.3.1 k and all-terminal reliabilities

k-terminal reliability is a special case of all-terminal reliability. In the literature, the parameter k is usually selected as two. Similar to capacitated resilience, k-terminal reliability is first calculated at the user level and then the network level k-terminal reliability is obtained by a weighted average in terms of user traffic requirements (Equation 4.4).

$$k\text{-terminal reliability} = \sum_{U_i \in\ Users} w_i * k\text{-terminal reliability}(U_i) \tag{4.4}$$

In this dissertation, two-terminal reliability is calculated at the user level by calculating the parallel reliability of the assigned path of the user to an AP and a disjoint path from the user to the same AP (Equation 4.5). All terminal reliability is the parallel reliability of the assigned user path and all other disjoint paths from the user to all available APs.

$$\text{Two-terminal reliability}(U_i) = 1 - \prod_{p_i \in \text{ Paths from } U_i \text{ to the AP}} [1 - R(p_i)] \tag{4.5}$$

4.3.2 Traffic efficiency (TE)

Due to intractability of exact calculation of TE, Konak and Bartolacci (2007) proposed a simulation based method, Sequential Construction. One replication of their simulation algorithm is summarized in Algorithm 4.6.

In this algorithm, both node and edge failures are considered. The algorithm randomly starts network components (node or link) from either operative or failed states according to reliabilities (x_{ij} is 1 if the component is operational) and improves connectivity incrementally by repair of each component (changing its state from failed to operative), where component (i, j) is an edge if $i \neq j$, a node otherwise $(i = j)$. The permutation of the component states is randomly generated (according to node and link reliabilities) for each replication of the algorithm. The algorithm calculates the weighted average of the successful traffic flow to estimate TE using the total traffic demand on the network (γ) and the traffic demand between nodes i and j (t_{ij}).

In this dissertation, the Sequential Construction algorithm of Konak and Bartolacci (2007) has been used. They used 300 iterations for all solutions and 5000 iterations for promising solutions to estimate TE. In here, the number of replications is selected as 2000 to estimate TE because the initial experimentation showed that the difference between the estimations by 2000 and by 5000 replications is negligible.

Algorithm 4.6 Pseudocode of Sequential Construction method to estimate TE (Konak and Bartolacci, 2007)

Sample network state x by assigning a random number U for each component (i, j), if $U > p_{ij}$ then $x_{ij} = 0$ ($x_{ij} = 1$, otherwise).
Generate random permutation Π of components in operative and failed states
Set $T \leftarrow 0$, $h \leftarrow 1$, $c \leftarrow 0$, $a \leftarrow 0$, label$[i] \leftarrow i$ for nodes $i = 1, 2, ..., n$ and $x_{ij} \leftarrow 0$ for each component (i, j).
// T, h, a and c are the temporary variables used in the simulation.
for ($r \in 1, 2, ..., (n + m)$) **do**
    $c = c(\frac{n+m-r+1}{r})(\frac{p_{[r]}}{1-p_{[r]}})$ where $p_{[r]}$ is the reliability of the rth component in permutation Π that is sampled in the previous step.
    Repair the rth component in permutation Π.
    if (rth component is a link $(i_{[r]}, j_{[r]})$, and both nodes $i_{[r]}$ and $j_{[r]}$ are in the operative state and label$[i_{[r]}] \neq$ label$[j_{[r]}]$) **then**
        Set the label of each node whose label is equal to the label of node $i_{[r]}$ to the label of node $i_{[r]}$
    end if
    if (rth component is a node $(i_{[r]}, i_{[r]})$) **then**
        for (each link $(i_{[j]}, r)$ incident to node $i_{[r]}$) **do**
            if (link $(i_{[j]}, r)$ and node j are operative and label$[i_{[r]}] \neq$ label$[j_{[r]}]$) **then**
                Set the label of each node whose label is equal to the label of node j to the label of node $i_{[j]}$
            end if
        end for
    end if
    for (each node pair i and j) **do**
        if (label$[i]$ = label$[j]$) **then**
            $T = T + t_{ij}$
        end if
    end for
    $h = h + c$, $a = a + c * T/\gamma$
end for
return Estimation of TE as a/h.

4.3.3 k-connectivity

k-connectivity is calculated similar to k-terminal reliability. The user level k-connectivity is checked and the network is said to be k-connected if all users are k-connected. If any user is not k-connected then network k-connectivity fails.

In this dissertation, k is selected as two. Vertex (node) and edge disjoint connectivity are both examined. A user is two vertex connected if two different paths connecting the user to its assigned AP have no common vertex. For two edge connectivity, the two paths must not have a common edge.

Chapter 5

Preliminary Results

This chapter summarizes the performance of the proposed single and bi-objective ES models. It shows the variation of the ES in terms of solution quality and Pareto front diversity. It also compares single and bi-objective ES performance to validate that the ES is working satisfactorily prior to the work presented in Chapter 6. This chapter also explains the key concepts of Pareto front diversity and non-dominated solutions which were introduced in the previous chapter.

The problem size is selected as 10 users and the budget is constrained to 600 (17 devices) because it is a small sized scenario with short solution times. The budget is selected as 600 to make problem quite constrained and therefore harder than a scenario with a larger budget. Capacitated resilience, cost, and Pareto front diversity are compared for different problem instances. The bi-objective ES ran for 2000 generations with an early termination criterion of 500 non-improved generations, whereas the single objective optimizer ran for 1000 generations with an early termination criterion of 250 non-improved generations. Both single objective and bi-objective ES use 30 population members and 30 children. These parameters were specified during the experimentation of the model building phase with consideration of the trade-off between solution time and solution quality. The ES is not very sensitive to the parameter values of size of population and children, however, a larger size of population (and children) helps improve solution quality with an increase of the solution time. Also, it improves population diversity for bi-objective optimization. After extensive experimentation with different parameter values, these values were selected. For example, bi-objective ES runs longer (2000 generations) than single objective ES (1000 generations) because Pareto optimality requires more time to find good solutions. Running ES longer increases solution

time but provides a slight improvement on the solution quality (more on this in Sections 5.2 and 5.3). The selected parameter values were tested on a wide range of test problems and they are valid for different problem sizes.

The bi-objective Pareto front is updated at each generation. Therefore, the size of the Pareto front is dynamic. On the other hand, the single objective ES is solved for maximum capacitated resilience under a budget constraint. The single objective "Pareto front" was generated using different budget values, i.e., 26 problem instances with different budget constraints (from 350 to 600 in increments of 10) are solved for maximum resilience. The best solutions of these 26 runs were combined and the non-dominated solutions are extracted to form the "Pareto front" of the single objective ES. For the bi-objective Pareto front, eight replications (different random number seeds) were used for each problem instance, however, only one random number seed was used to generate the single objective Pareto front.

The Pareto fronts of the single and bi-objective ES are compared in Section 5.1. In Section 5.2, the bi-objective ES is run longer to match the time of the single objective Pareto front generation process (the total time of 26 single objective runs) and its performance is compared with the single objective version. The variation of the single objective ES is investigated in Section 5.3. Note that variation in the bi-objective ES has been investigated with eight random number seeds in Sections 5.1 and 5.2.

5.1 Single objective vs. bi-objective ES

For each problem instance, eight random number seeds were used for the bi-objective optimization, and they were compared with the solutions that were generated from the single objective ES.

5.1.1 Pareto front diversity

The ranges of capacitated resilience and the cost of single and bi-objective Pareto front solutions are given in Figures 5.1, 5.2 and 5.3. These are the non-dominated solutions. To

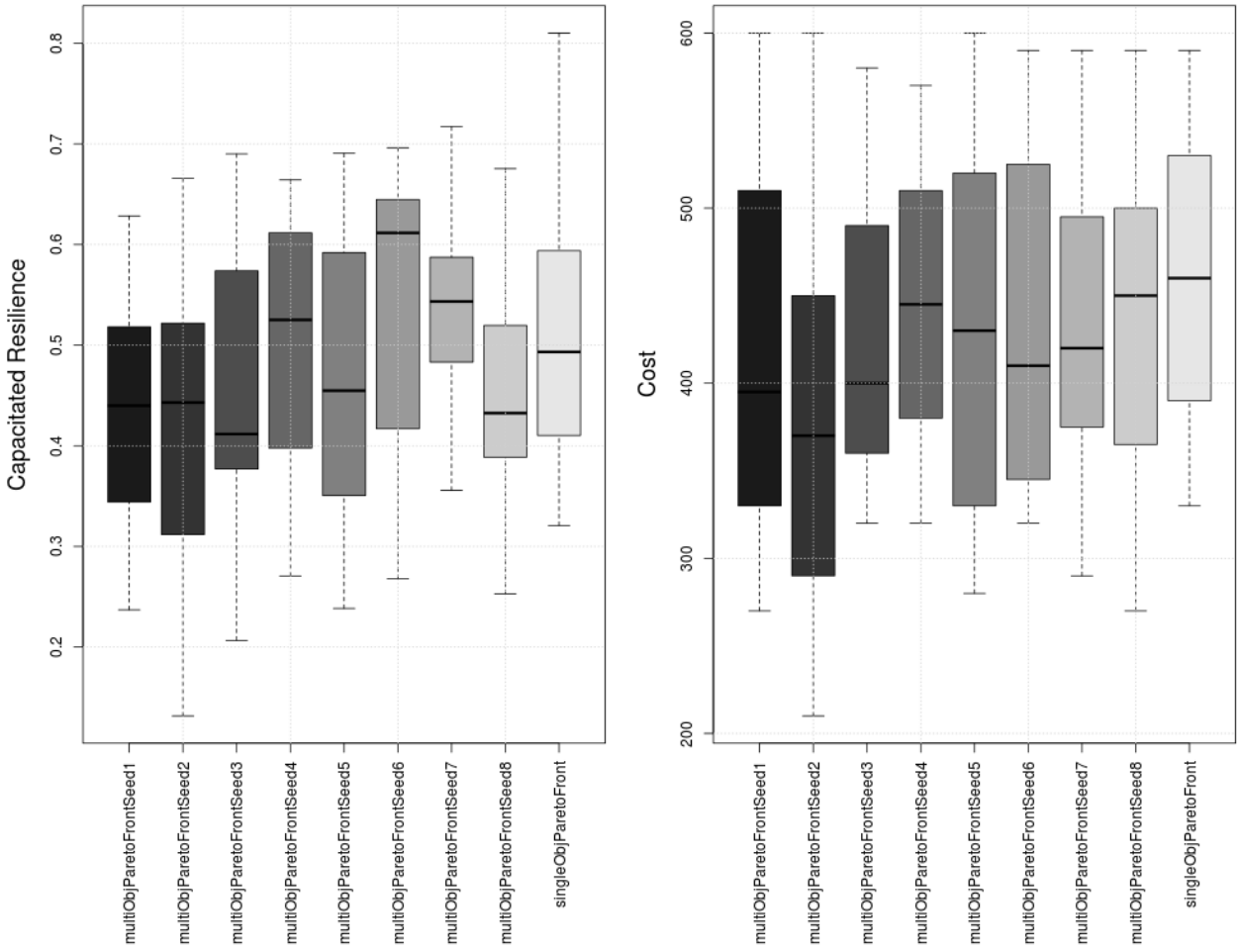
find the non-dominated solutions, single and bi-objective Pareto fronts are compared with each other. A solution of the single objective Pareto front is non-dominated if it is not dominated by any solution of the bi-objective Pareto front. Non-dominated solutions of bi-objective ES are found similarly. The total number of solutions in each Pareto front and the number of non-dominated solutions are given in the next section.

The performance of the bi-objective ES varies with random number seed. Only one random number seed is used to generate the single objective Pareto front due to the long solution time (requires 26 runs) of the Pareto front generation process. The variation of the single objective ES is investigated in Section 5.3, For example, random number seed five of problem instance two has a better Pareto front diversity and a better capacitated resilience value than the single objective Pareto front. However, random number seed seven of problem instance one has a worse diversity and lower capacitated resilience value. For problem instances two and three, bi-objective ES outperforms single objective in terms of capacitated resilience. However, single objective performs better for problem instance one. Therefore, the performance comparison for capacitated resilience is not conclusive. However, the bi-objective Pareto front has a better diversity in terms of cost.

5.1.2 Comparison of non-dominated solutions for bi-objective and single objective Pareto fronts

The Pareto front solutions of bi-objective ES were compared with the Pareto front solutions of single objective ES, and vice versa.

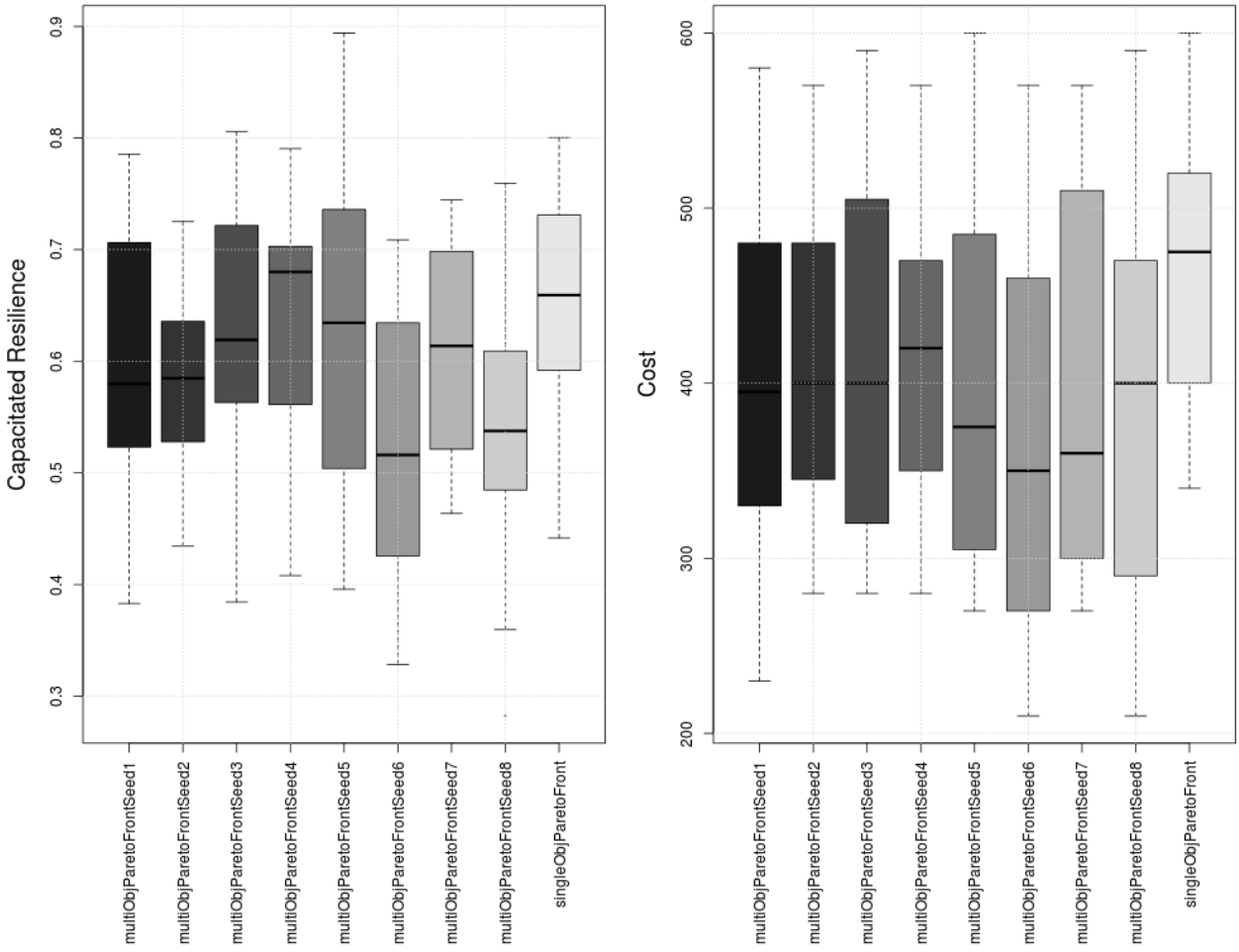
In Table 5.1, the non-dominated solutions are reported for each Pareto Front. In the table, the non-dominated solutions of bi-objective ES were found by calculating the number of bi-objective Pareto front solutions that are not dominated by single objective Pareto front solutions. The single objective Pareto front solutions were compared with the bi-objective Pareto front solutions in the same way. Bi-Objective outperforms single objective for some scenarios (e.g., random number seed four of problem instance three) but single objective



(a) Capacitated Resilience Comparison

(b) Cost Comparison

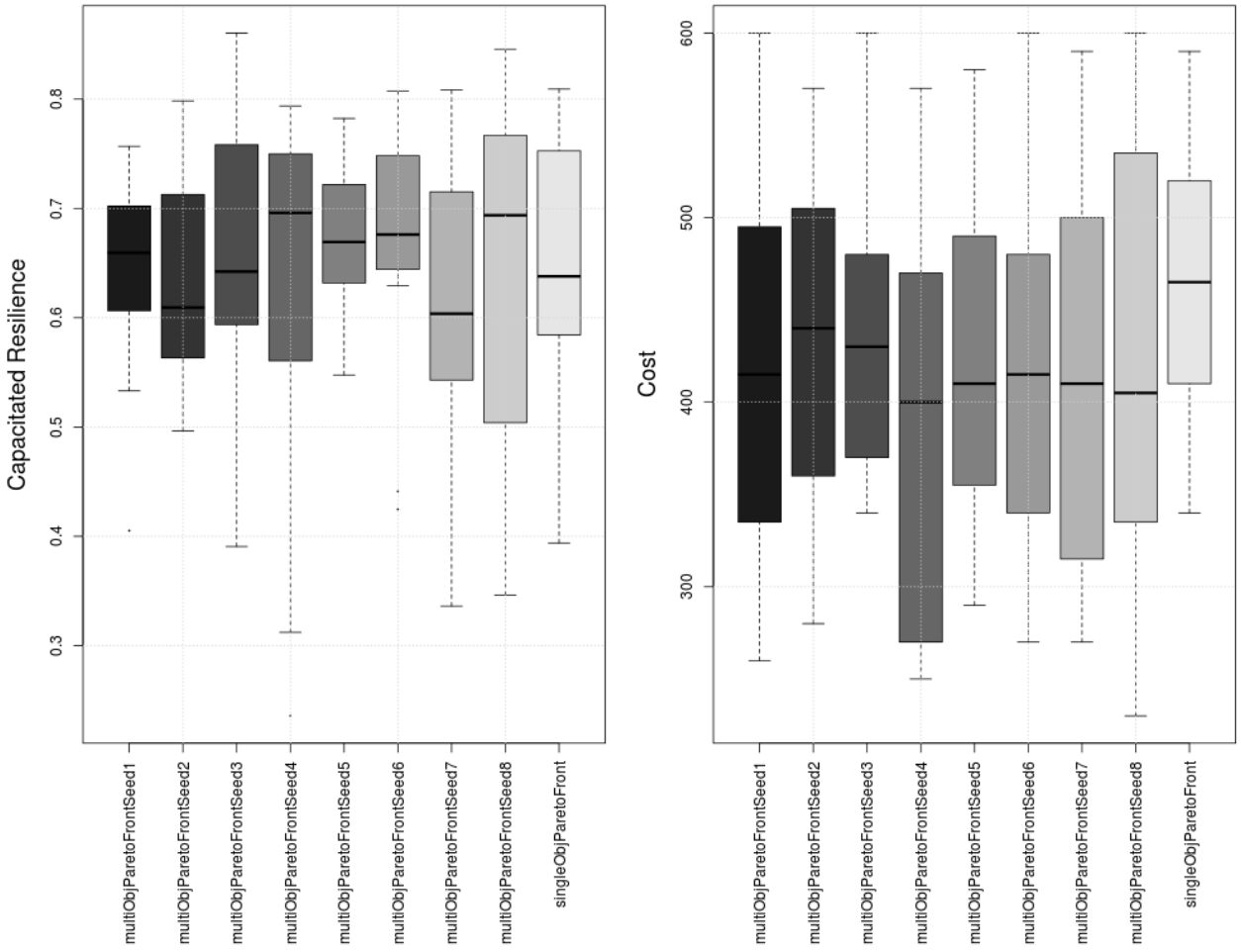
Figure 5.1: Comparison of Pareto front solutions (Problem instance 1)



(a) Capacitated Resilience Comparison

(b) Cost Comparison

Figure 5.2: Comparison of Pareto front solutions (Problem instance 2)



(a) Capacitated Resilience Comparison

(b) Cost Comparison

Figure 5.3: Comparison of Pareto front solutions (Problem instance 3)

dominates bi-objective for most scenarios. However, these results can be misleading because the solution times of bi-objective and single objective are not equal (single objective combines 26 different runs to generate the Pareto front). Therefore, Section 5.2 repeats the same analysis presented in this section where the bi-objective optimizer runs longer.

Table 5.1: Performance of single and bi-objective ES for three problem instances

		Method	
		Bi-Objective	Single Objective
Problem Instance 1	Rnd Number Seed 1	4/14*	6/6
	Rnd Number Seed 2	8/14	5/6
	Rnd Number Seed 3	1/15	6/6
	Rnd Number Seed 4	2/10	5/6
	Rnd Number Seed 5	5/15	5/6
	Rnd Number Seed 6	6/11	3/6
	Rnd Number Seed 7	3/11	5/6
	Rnd Number Seed 8	2/15	6/6
Problem Instance 2	Rnd Number Seed 1	4/13	7/8
	Rnd Number Seed 2	6/12	7/8
	Rnd Number Seed 3	11/16	5/8
	Rnd Number Seed 4	9/10	7/8
	Rnd Number Seed 5	9/21	7/8
	Rnd Number Seed 6	7/17	7/8
	Rnd Number Seed 7	7/15	7/8
	Rnd Number Seed 8	7/17	8/8
Problem Instance 3	Rnd Number Seed 1	11/12	4/9
	Rnd Number Seed 2	6/11	8/9
	Rnd Number Seed 3	12/15	4/9
	Rnd Number Seed 4	9/11	2/9
	Rnd Number Seed 5	4/9	6/9
	Rnd Number Seed 6	11/14	3/9
	Rnd Number Seed 7	6/10	5/9
	Rnd Number Seed 8	6/12	8/9

* Number of nondominated solutions (single and bi-objective Pareto fronts are compared with each other) / total solutions in the Pareto front

5.1.3 Single and bi-objective Pareto Fronts

This section shows the Pareto fronts of both single and bi-objective ES. Specifically, it presents the diversity of the Pareto front solutions. Also, single and bi-objective Pareto front solutions are graphically compared. Figures 5.4, 5.5 and 5.6 compare the population and Pareto front members of single and bi-objective ES. Population members of the single

objective are the 26 solutions that are generated using different budget constraints. Non-dominated solutions of the single objective population members form the single objective Pareto front.

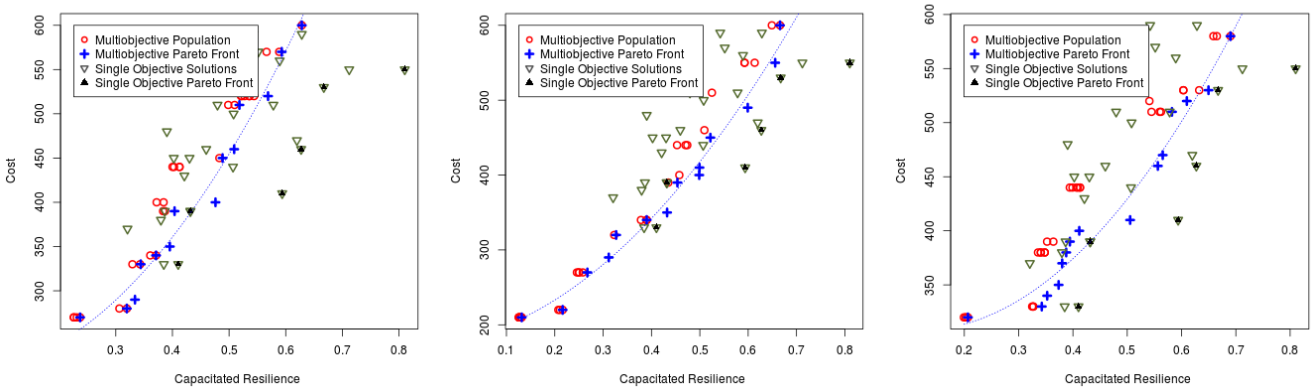
Except from some instances (e.g., random number seed four of problem instance three), the single objective outperforms bi-objective. However, the bi-objective Pareto front has better diversity than the single objective. There is also a gap between the bi-objective population and the bi-objective Pareto front in terms of solution quality and therefore these results confirm the need of keeping a global Pareto front in an external repository for bi-objective ES (Section 4.2.3).

5.2 Equalizing computational effort

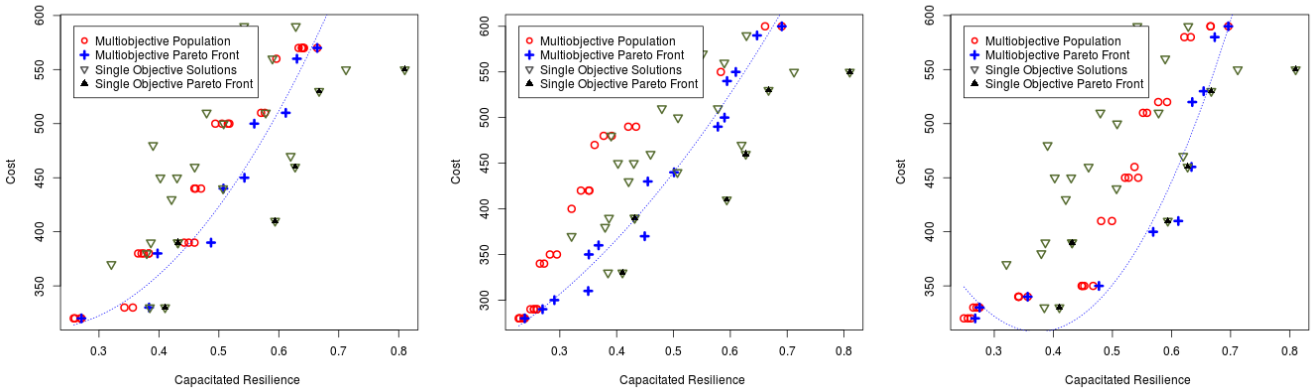
In this section, different from Section 5.1, the bi-objective ES was run longer to match the time of the single objective Pareto front generation process. To match the time of generating 26 different budget values, the bi-objective optimizer was run for 16,000 generations with an early termination of 4000 non-improving generations. The rationale of using 16,000 generations instead of 26,000 (26 runs of 1000 generations each) is that the total time of 26 single objective runs for budget values from 350 to 600 is equivalent to the running time of 16,000 bi-objective generations. For each problem instance, the single objective was run for only one random number seed (due to the long process of generating single objective Pareto front) and the bi-objective optimizer was run for 10 random number seeds. Variation in the single objective ES is discussed in Section 5.3.

5.2.1 Pareto front diversity

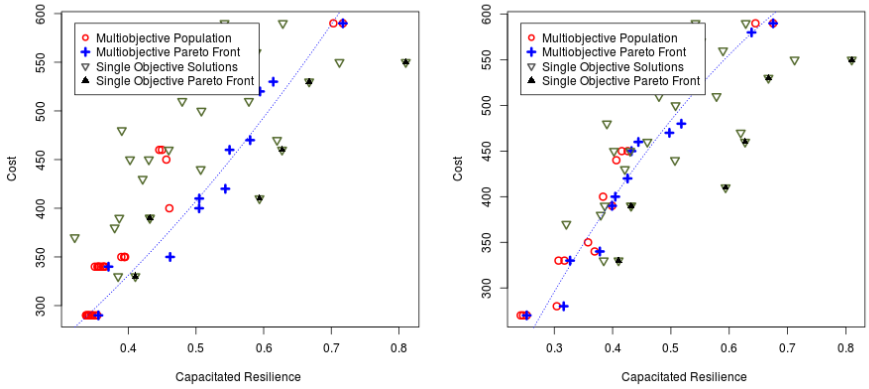
Figures 5.7, 5.8 and 5.9 summarize the diversity of the bi-objective and single objective Pareto fronts. For three problem instances, bi-objective ES has a better diversity (in cost and capacitated resilience) than the single objective version. Also, the capacitated resilience



(a) Random Number Seed 1 (b) Random Number Seed 2 (c) Random Number Seed 3

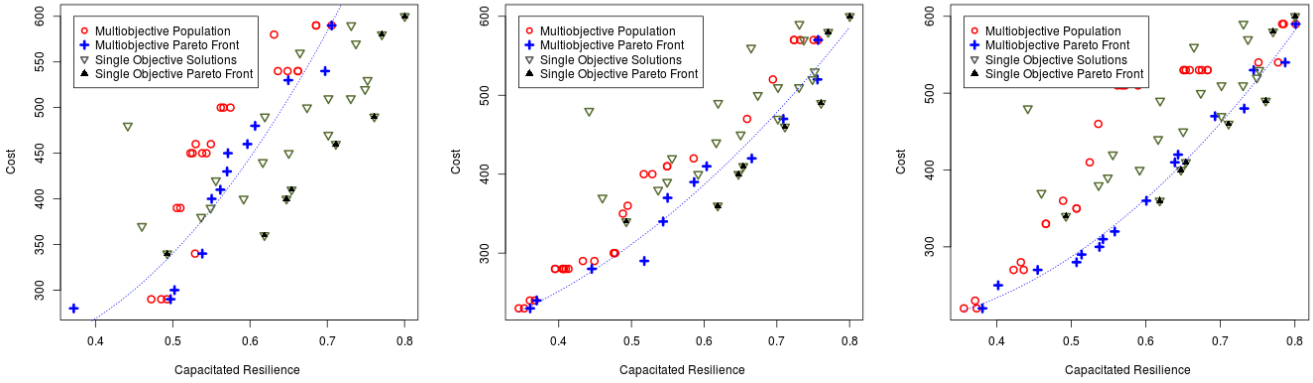


(d) Random Number Seed 4 (e) Random Number Seed 5 (f) Random Number Seed 6

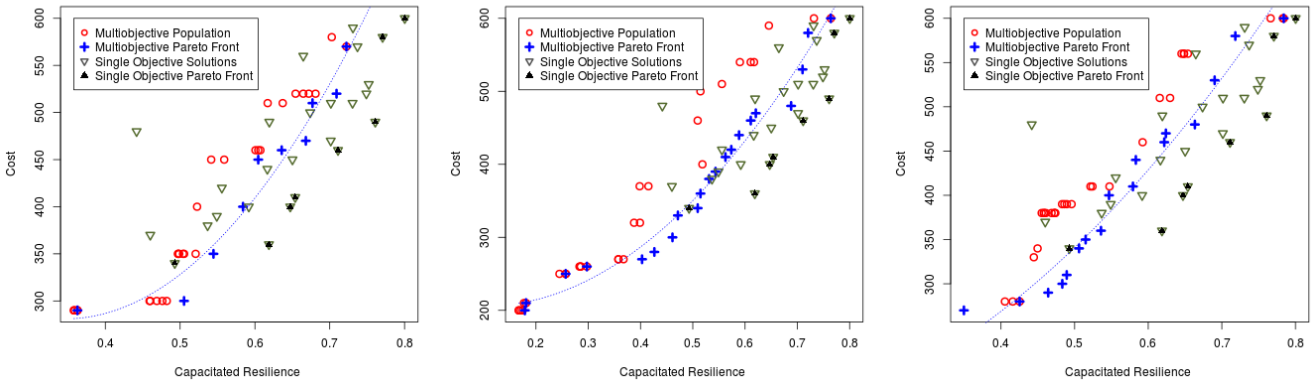


(g) Random Number Seed 7 (h) Random Number Seed 8

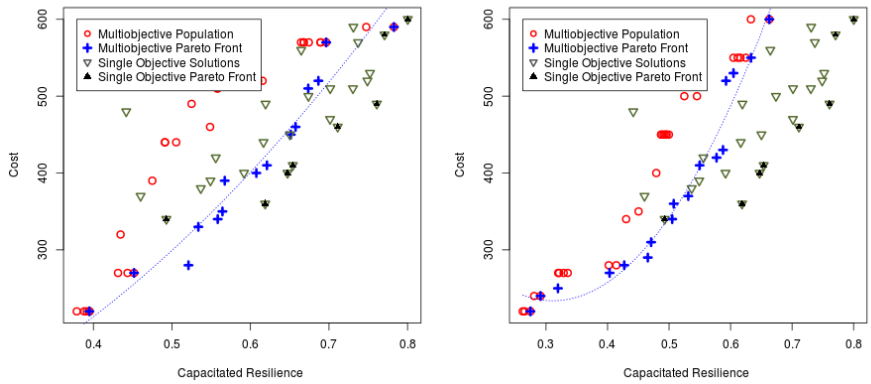
Figure 5.4: Pareto fronts for problem instance 1: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds)



(a) Random Number Seed 1 (b) Random Number Seed 2 (c) Random Number Seed 3

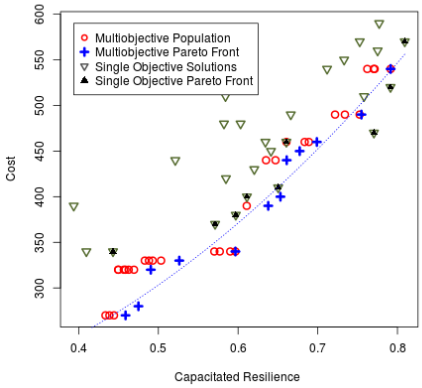


(d) Random Number Seed 4 (e) Random Number Seed 5 (f) Random Number Seed 6

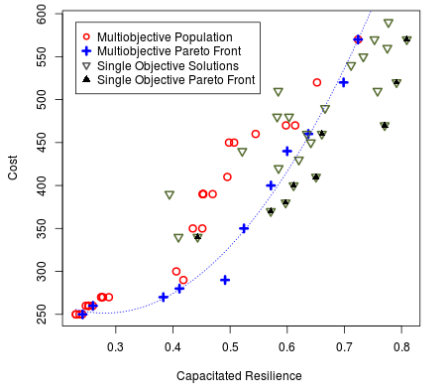


(g) Random Number Seed 7 (h) Random Number Seed 8

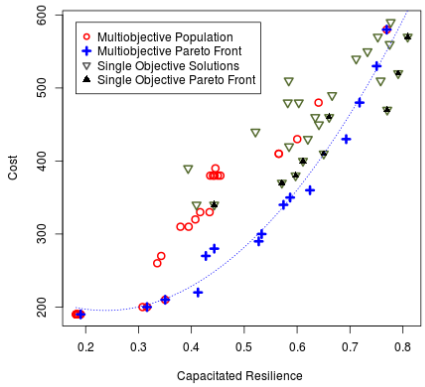
Figure 5.5: Pareto fronts for problem instance 2: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds)



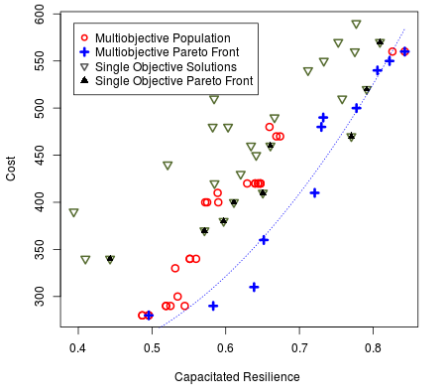
(a) Random Number Seed 1



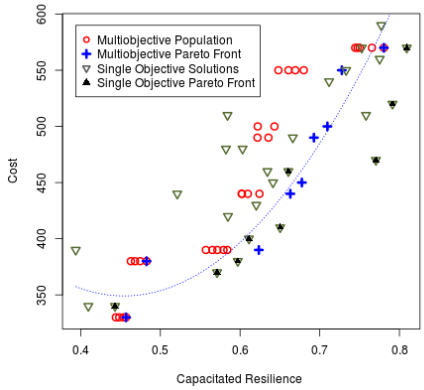
(b) Random Number Seed 2



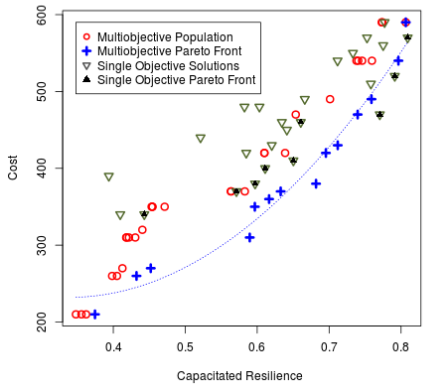
(c) Random Number Seed 3



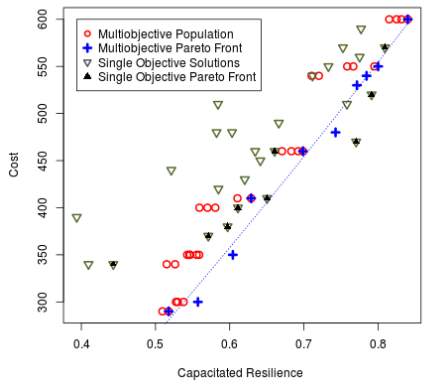
(d) Random Number Seed 4



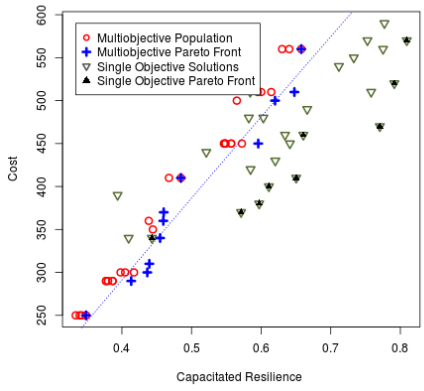
(e) Random Number Seed 5



(f) Random Number Seed 6

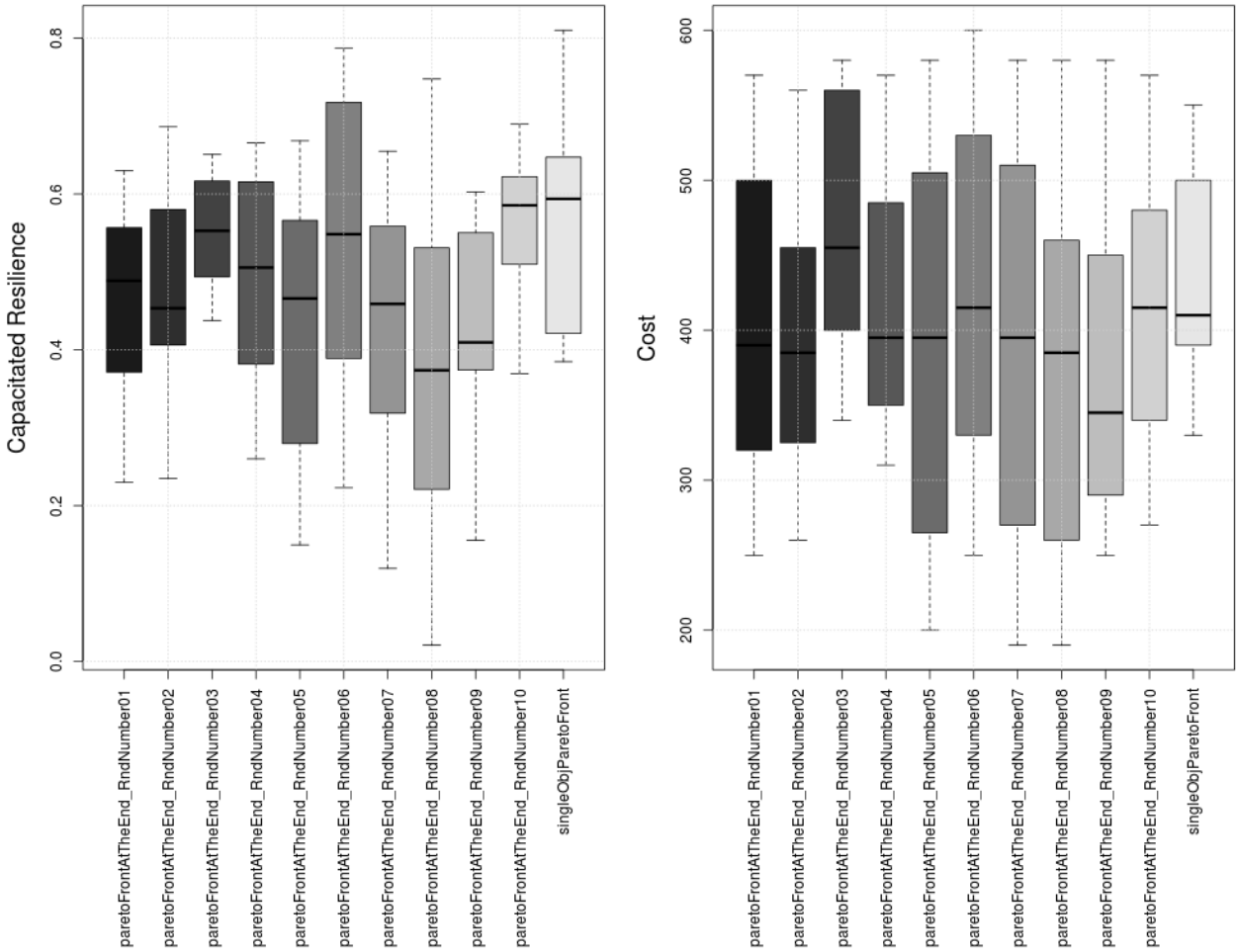


(g) Random Number Seed 7



(h) Random Number Seed 8

Figure 5.6: Pareto fronts for problem instance 3: Single objective (obtained by 26 different budget values) and bi-objective (for different random number seeds)



(a) Capacitated Resilience Comparison

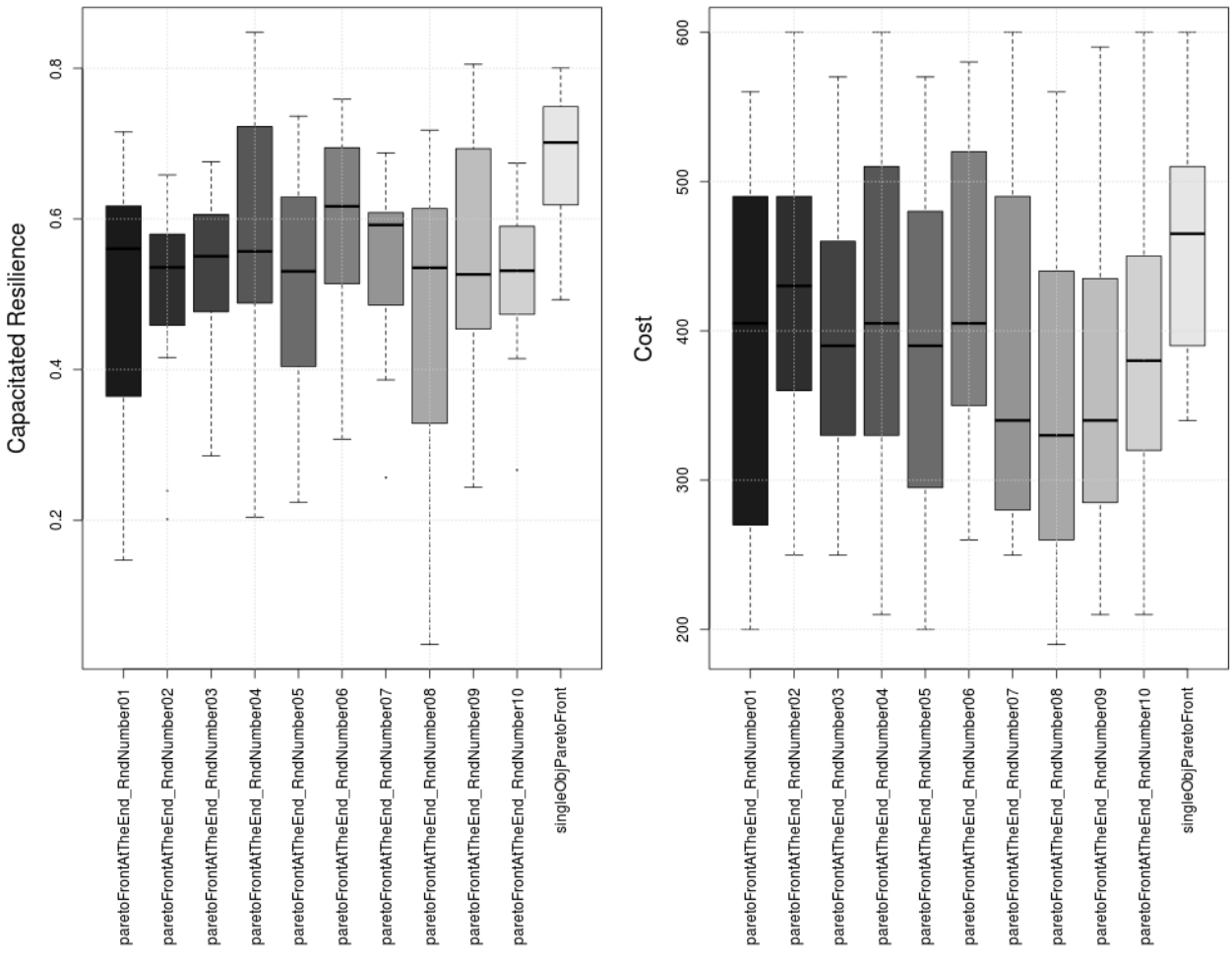
(b) Cost Comparison

Figure 5.7: Comparison of Pareto front solutions (Problem instance 1)

values have been improved over the shorter bi-objective runs with the longer solution times (in comparison to Section 5.1).

5.2.2 Comparison of non-dominated solutions for bi-objective and single objective Pareto fronts

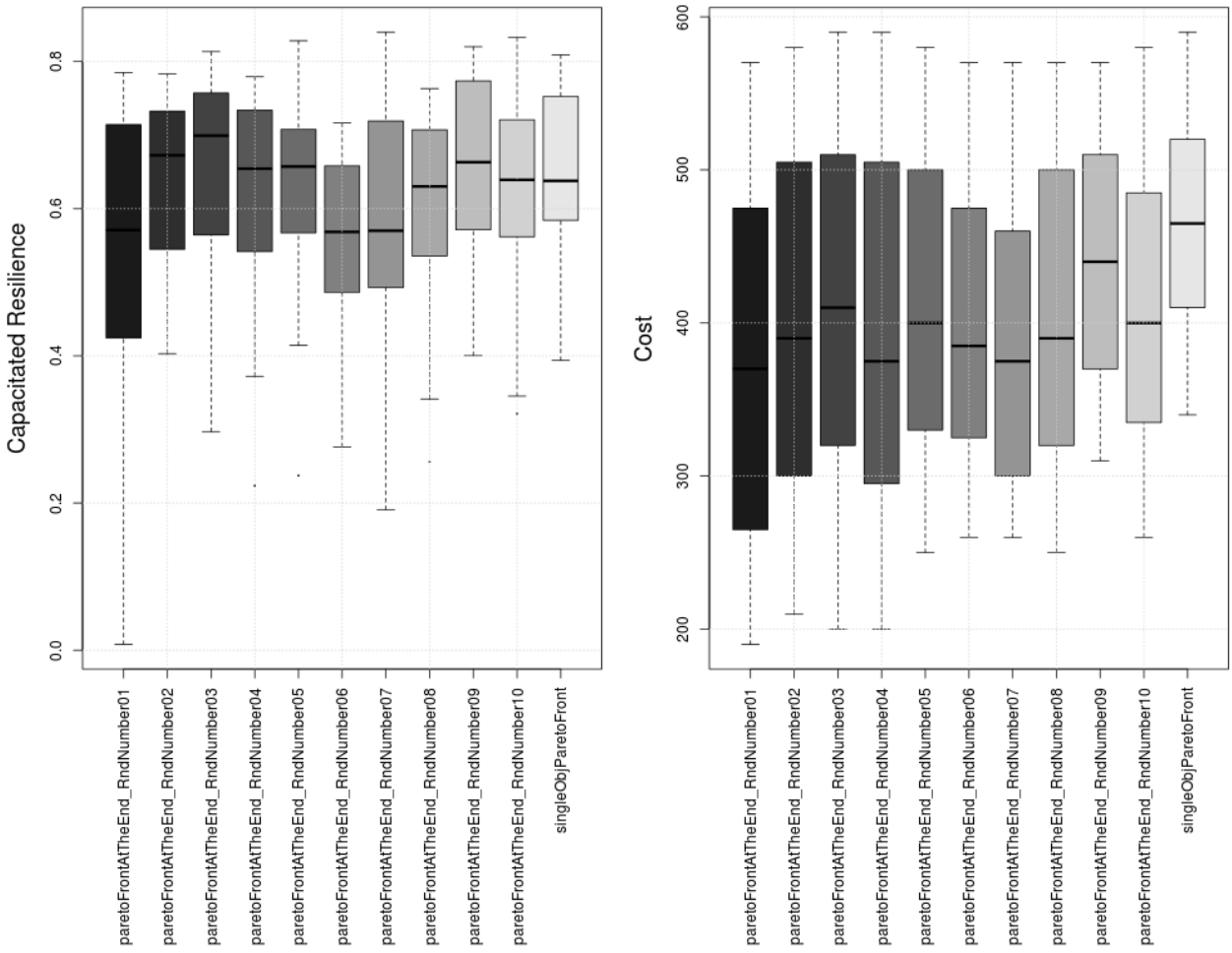
The Pareto front solutions of bi-objective ES were compared with the Pareto front solutions of single objective ES, and vice versa. Table 5.2 summarizes the number of non-dominated solutions in each combined Pareto front. As expected, running the bi-objective



(a) Capacitated Resilience Comparison

(b) Cost Comparison

Figure 5.8: Comparison of Pareto front solutions (Problem instance 2)



(a) Capacitated Resilience Comparison

(b) Cost Comparison

Figure 5.9: Comparison of Pareto front solutions (Problem instance 3)

Table 5.2: Comparison of Pareto solutions of single (1000 generations) and bi-objective (16,000 generations) ES for three problem instances

		Method	
		Bi-Objective	Single Objective
Problem Instance 1	Rnd Number Seed 1	6/13*	5/6
	Rnd Number Seed 2	9/12	2/6
	Rnd Number Seed 3	3/10	5/6
	Rnd Number Seed 4	5/12	6/6
	Rnd Number Seed 5	8/16	5/6
	Rnd Number Seed 6	9/14	4/6
	Rnd Number Seed 7	11/18	5/6
	Rnd Number Seed 8	9/18	6/6
Problem Instance 2	Rnd Number Seed 1	6/14	7/8
	Rnd Number Seed 2	3/13	8/8
	Rnd Number Seed 3	5/13	7/8
	Rnd Number Seed 4	8/18	6/8
	Rnd Number Seed 5	7/19	8/8
	Rnd Number Seed 6	6/18	7/8
	Rnd Number Seed 7	7/13	7/8
	Rnd Number Seed 8	7/13	7/8
Problem Instance 3	Rnd Number Seed 1	9/15	5/9
	Rnd Number Seed 2	10/15	4/9
	Rnd Number Seed 3	12/17	3/9
	Rnd Number Seed 4	11/16	3/9
	Rnd Number Seed 5	14/17	3/9
	Rnd Number Seed 6	5/12	8/9
	Rnd Number Seed 7	12/14	4/9
	Rnd Number Seed 8	7/13	5/9

* Number of nondominated solutions (single and bi-objective Pareto fronts are compared with each other) / total solutions in the Pareto front

version longer helped improve the Pareto front; the bi-objective Pareto front has better solutions (in terms of number of non-dominated solutions) than the ones in Section 5.1. And, the number of solutions in the bi-objective Pareto front is increased. Comparing to the single objective Pareto front, the bi-objective Pareto front performs better for problem instances one and three (for most random number seeds), however, the single objective Pareto front has more non-dominated solutions for problem instance two (except for random number seed four).

5.3 Variation of single objective ES

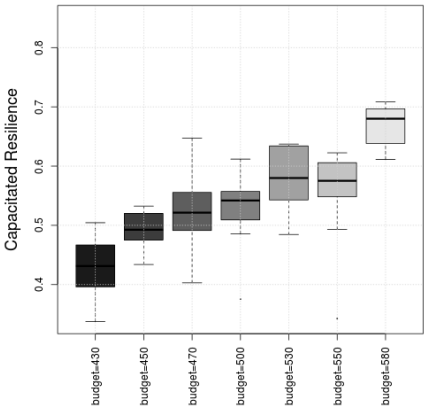
Variation of the single objective ES is summarized in this section. To show the variation in capacitated resilience and cost, three problem instances of the single objective ES was run for 10 random number seeds under different budget constraints (for 1000 generations).

Figure 5.10 summarizes the results. Interestingly, there is variation in cost, which means that the optimizer can not always fully utilize the budget. This also causes variation in the capacitated resilience. To see the relationship between the variation and the number of generations, the same experiment was performed with 5000 generations (Figure 5.11). A similar variation has been observed for 5000 generations. In summary, the single objective ES is sensitive to random number seed.

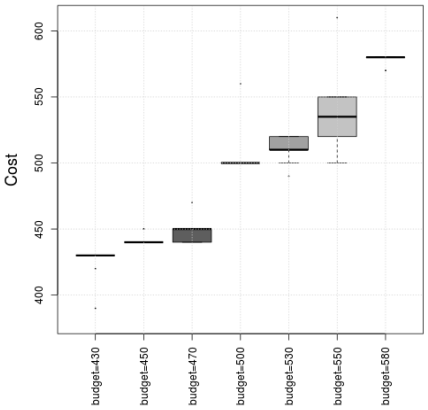
A comparison of the variability of single and bi-objective ES is provided in Section 5.4.

5.4 Summary

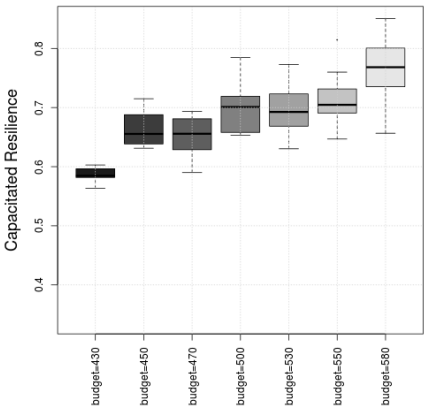
According to the preliminary results presented in this chapter, the performance of single and bi-objective ES are comparable in terms of Pareto front diversity, capacitated resilience and cost. The Pareto front diversity of the bi-objective version was compared with the single objective Pareto front and it was found satisfactory (in solution quality and diversity) given that the single objective Pareto front requires a longer time to construct. According to the two-sample t-tests, the single objective ES found better capacitated resilience values than bi-objective only for problem instance 4 (p-value=0.0202) of the budget=500 case and problem instances 3 (p-value=0.0220) and 10 (p-value=0.1002) of the budget=600 case. For the remaining problem instances, the difference in capacitated resilience between single and bi-objective ES was not statistically significant. Also, the difference in cost was not statistically significant in any of the problem instances. The advantage of using bi-objective over single objective ES is that the bi-objective provides a non-dominated set of solutions instead of a single solution. A decision maker can decide on the cost and capacitated resilience trade-off



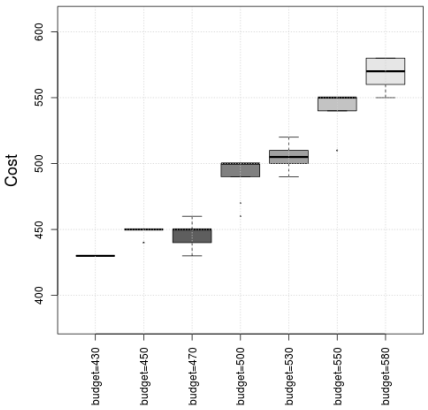
(a) Problem instance 1 (Capacitated Resilience)



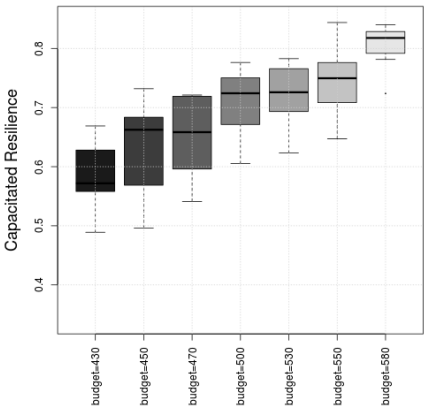
(b) Problem instance 1 (Cost)



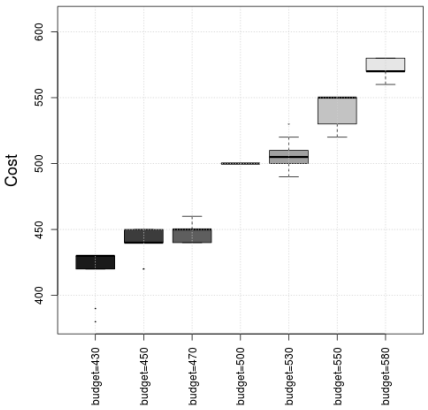
(c) Problem instance 2 (Capacitated Resilience)



(d) Problem instance 2 (Cost)

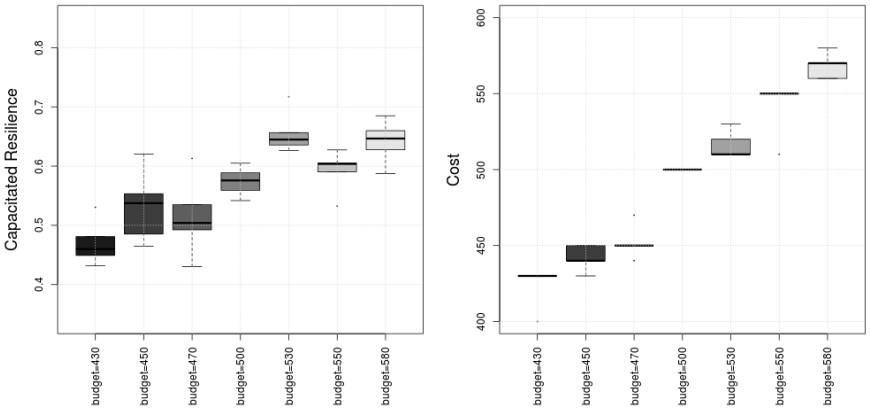


(e) Problem instance 3 (Capacitated Resilience)



(f) Problem instance 3 (Cost)

Figure 5.10: Variation of the single objective ES for different budget values



(a) Problem instance 1 (Capacitated Resilience)

(b) Problem instance 1 (Cost)

Figure 5.11: Variation of the single objective ES for different budget values (5000 generations, 10 random number seeds)

and select the best non-dominated solution. Also, as seen in Sections 5.1 and 5.2, single run of bi-objective ES provides a better diversity than multiple runs of single objective ES and runs faster.

Both single and bi-objective ES have some variation in solution quality. Table 5.3 shows the variation in capacitated resilience of the single and bi-objective ES for the 10 user scenario with two budget levels (10 problem instances with 10 random number seeds each). The standard deviation of the runs does not exceed 0.105 for the bi-objective (problem instance 2/budget=500) and 0.166 for single objective (problem instance 9/budget=600). Also, the standard deviations of the single and bi-objective ES are very similar. According to the F-tests for equal variance of capacitated resilience, the variances of single and bi-objective ES are not statistically different except in four problem instances: problem instances 4 (p-value=0.0351), 7 (p-value=0.0343) and 8 (p-value=0.0435) of the budget=500 case and problem instance 9 (p-value=0.0105) of the budget=600 case. It can be concluded that the variation to seed in capacitated resilience is acceptable (standard deviation is 0.05 on average) for both single and bi-objective ES.

This chapter analyzed the performance of the proposed model on a small sized problem. Succinctly, the performance of the bi-objective ES is comparable to the single objective. The next chapter compares capacitated resilience with other reliability/survivability metrics.

Table 5.3: Mean and standard deviation of capacitated resilience for single and bi-objective runs (10 problem instances of 10 user scenario with 10 random number seeds for each problem instance. Capacitated resilience is calculated for number of alternative paths=10 and cut set size=4.)

	Budget = 500				Budget = 600			
	Single Objective		Bi-Objective		Single Objective		Bi-Objective	
Problem Instance	Mean	Std. Deviation	Mean	Std. Deviation	Mean	Std. Deviation	Mean	Std. Deviation
1	0.53649	0.02250	0.55387	0.03767	0.68558	0.04548	0.62108	0.07699
2	0.65178	0.06636	0.68746	0.10526	0.75458	0.06276	0.73677	0.03069
3	0.72454	0.05161	0.69514	0.06401	0.79778[a]	0.02430	0.73450	0.04064
4	0.84978[a]	0.02366[b]	0.81169	0.00691	0.84355	0.01812	0.83708	0.01468
5	0.75027	0.04256	0.71340	0.02207	0.84719	0.04205	0.80499	0.05130
6	0.64184	0.05897	0.66774	0.05940	0.82046	0.04186	0.73924	0.07213
7	0.75881	0.05361[b]	0.74815	0.01556	0.81342	0.02375	0.80472	0.02362
8	0.70869	0.01769	0.70261	0.05704[b]	0.82065	0.03138	0.76657	0.04530
9	0.82082	0.04539	0.80234	0.02511	0.79605	0.16603[b]	0.81480	0.03496
10	0.59992	0.05781	0.63547	0.02829	0.77596[a]	0.04754	0.68771	0.02491

[a] Significantly larger mean (p-value < 0.05)
[b] Significantly larger standard deviation (p-value < 0.05)

Chapter 6

Results

6.1 Comparison of single (capacitated resilience) and bi-objective (cost and capacitated resilience) ES

In this section, single and bi-objective ES are compared for capacitated resilience. The Pareto front comparison of single and bi-objective ES was presented in Section 5.1, therefore Pareto fronts are not analyzed here.

Figures 6.1 through 6.5 summarize the results of the 10, 25, 50, 75 and 100 users scenarios, respectively. These figures compare the best capacitated resilience values of the single and bi-objective runs. Ten problem instances with five replications are analyzed for each scenario. The capacitated resilience in these figures is exactly calculated by setting the number of alternative paths to 10 and the cut set size to 4.

As seen from the figures, the single objective ES outperforms bi-objective in most problem instances. This is expected because the bi-objective ES maintains a diversified Pareto front to optimize the two objective functions simultaneously. For some problem instances (e.g., problem instance 1 of the 50 user scenario) the bi-objective ES finds better capacitated resilience values than the single objective. However, according to the two-sample t-tests, single objective is significantly better for only some problem instances. The single objective was significantly better for the 10 user scenario problem instance 3 with budget 600 (p-value=0.0220), problem instance 4 with budget 500 (p-value=0.0202), problem instance 10 with budget 600 (p-value=0.0102). The single objective ES was also statistically better than bi-objective ES for the 25 user scenario problem instance 2 with budget 1000 (p-value=0.0345) and budget 1200 (p-value=0.0053), problem instance 5 with budget 1200 (p-value=0.0111), problem instance 7 with budget 1000 (p-value=0.0011), problem instance

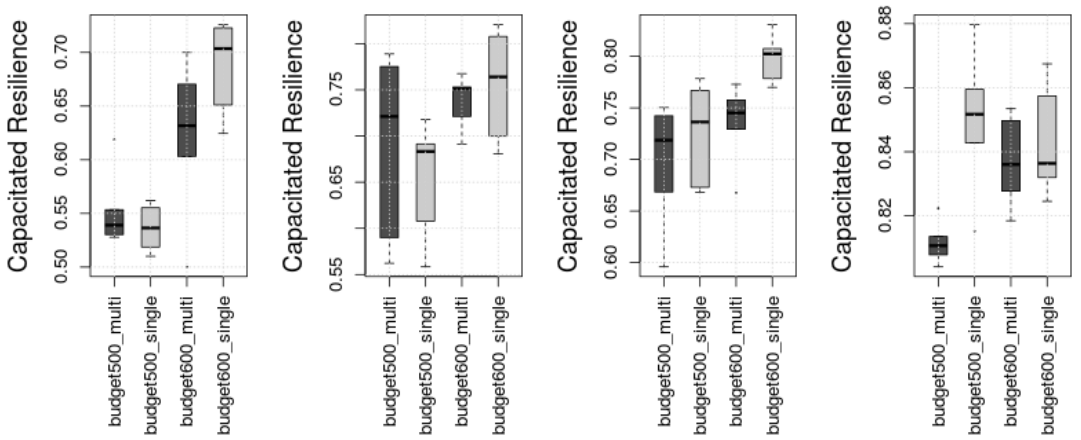
9 with budget 1000 (p-value=0.0049) and budget 1200 (p-value=0.0073), problem instance 10 with budget 1000 (p-value=0.0009) and budget 1200 (p-value=0.0257). For the 75 user scenario, the single objective was better than the bi-objective only for problem instance 2 with budget 2700 (p-value=0.0057) and problem instance 8 with budget 2700 (p-value=0.00185). For the 50 and 100 user scenarios the single objective ES was not significantly better for any problem instance. On the other hand, there was no problem instance that the bi-objective ES was statistically better than the single objective ES. To sum up, according to the results of the t-tests, the performance of the bi-objective ES is comparable to the single objective ES.

The performance gap between single and bi-objective optimizers reduces as the problem size increases (from 50 users to 100 users). For the 100 user scenario (Figure 6.5) bi-objective optimization was able to find better solutions than the single objective for most problem instances. For example, bi-objective was statistically better than the single objective for problem instance 8 at 10% significance level (p-value=0.0622). This may suggest that the bi-objective performs well for larger sized problems. Its diversified population due to the Pareto front helps to search the solution space more effectively than the single objective.

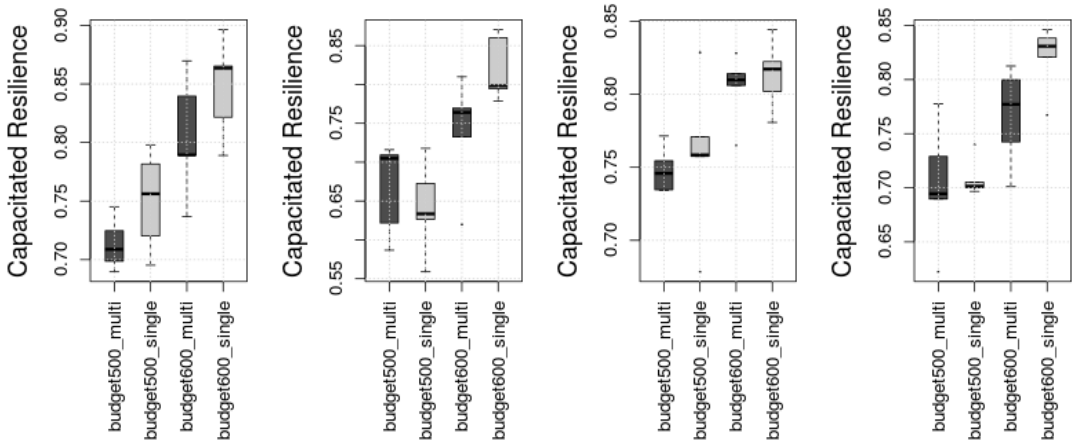
6.2 Correlation among capacitated resilience and other metrics

In this section, correlations among the capacitated resilience and the other metrics are investigated. The main reason to check correlations is to identify possible differences or similarities of the network structures.

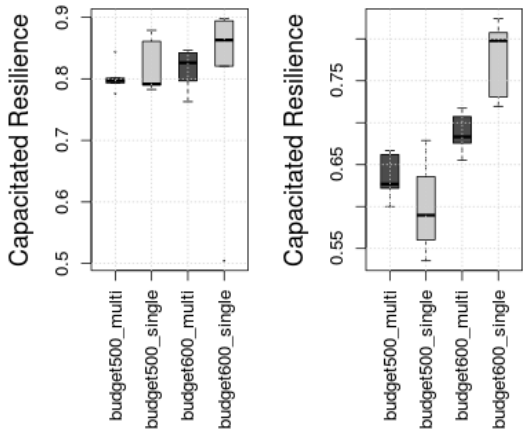
Table 6.1 summarizes the correlations using the best solutions over five random seeds of each problem instance (total of 10). In this table, each row shows correlations among the objective function and the other metrics. For example, the first row presents the correlations among capacitated resilience and the other metrics for the network designs found by optimization for capacitated resilience. The correlation between capacitated resilience



(a) Problem Instance 1 (b) Problem Instance 2 (c) Problem Instance 3 (d) Problem Instance 4

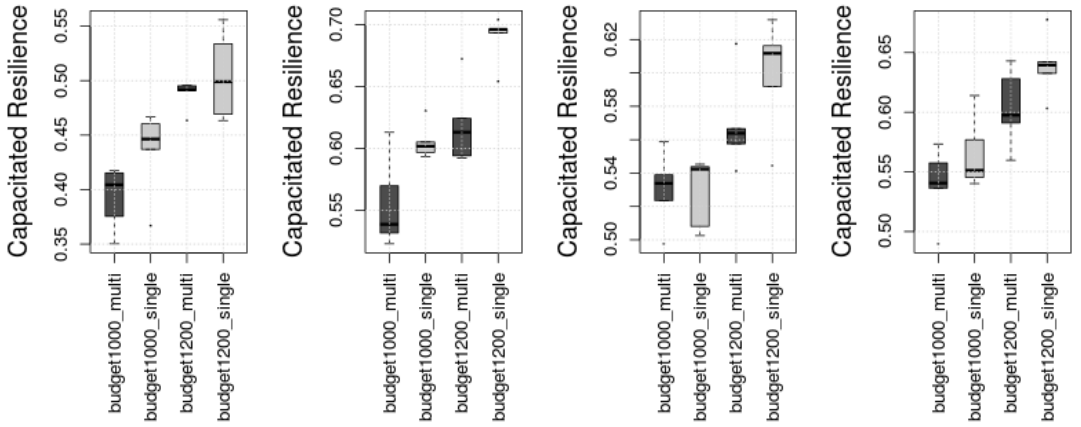


(e) Problem Instance 5 (f) Problem Instance 6 (g) Problem Instance 7 (h) Problem Instance 8

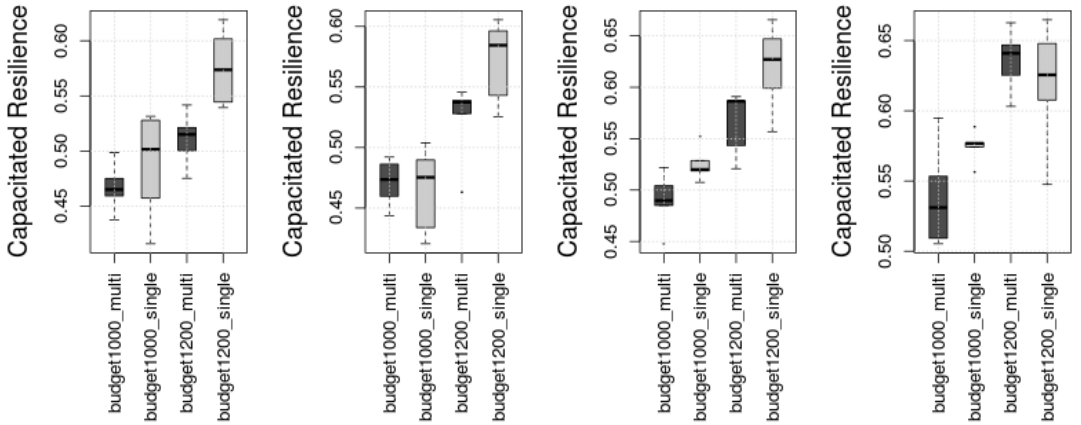


(i) Problem Instance 9 (j) Problem Instance 10

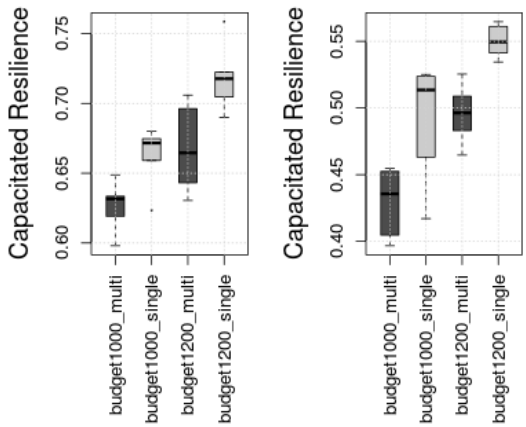
Figure 6.1: Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 10 user scenario



(a) Problem Instance 1 (b) Problem Instance 2 (c) Problem Instance 3 (d) Problem Instance 4

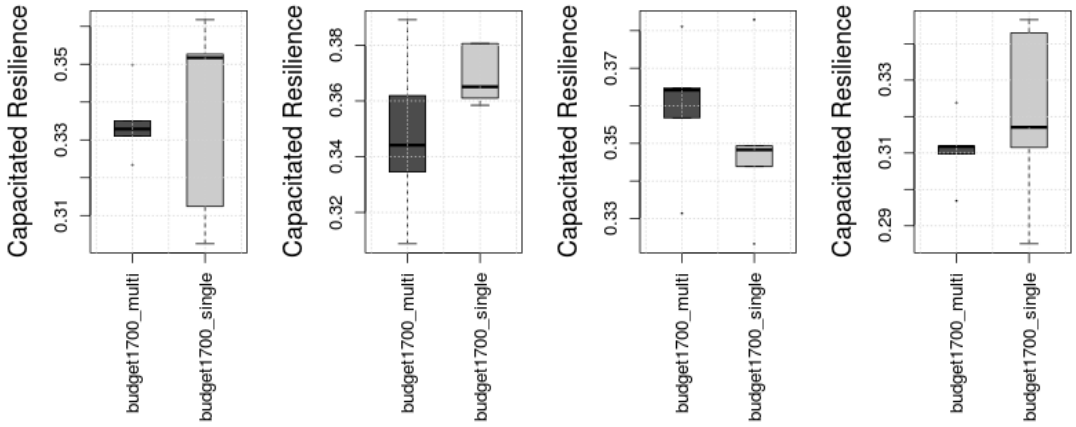


(e) Problem Instance 5 (f) Problem Instance 6 (g) Problem Instance 7 (h) Problem Instance 8

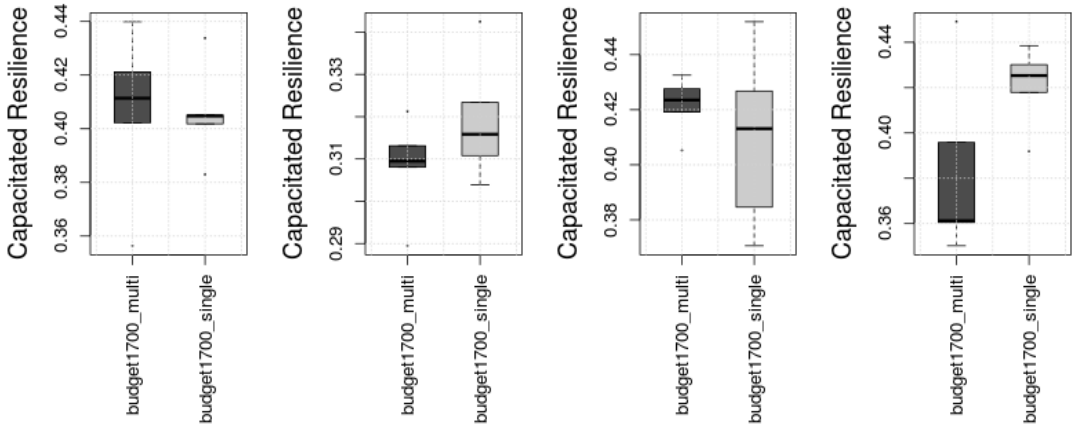


(i) Problem Instance 9 (j) Problem Instance 10

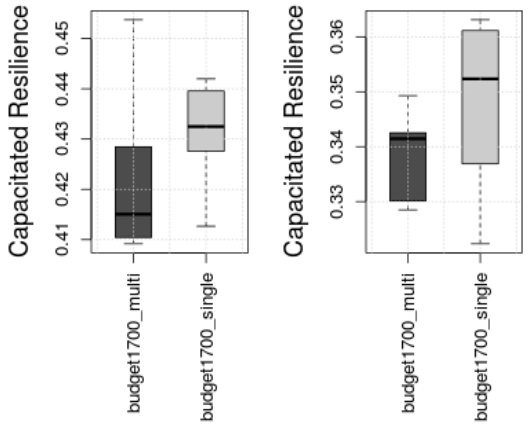
Figure 6.2: Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 25 user scenario



(a) Problem Instance 1 (b) Problem Instance 2 (c) Problem Instance 3 (d) Problem Instance 4

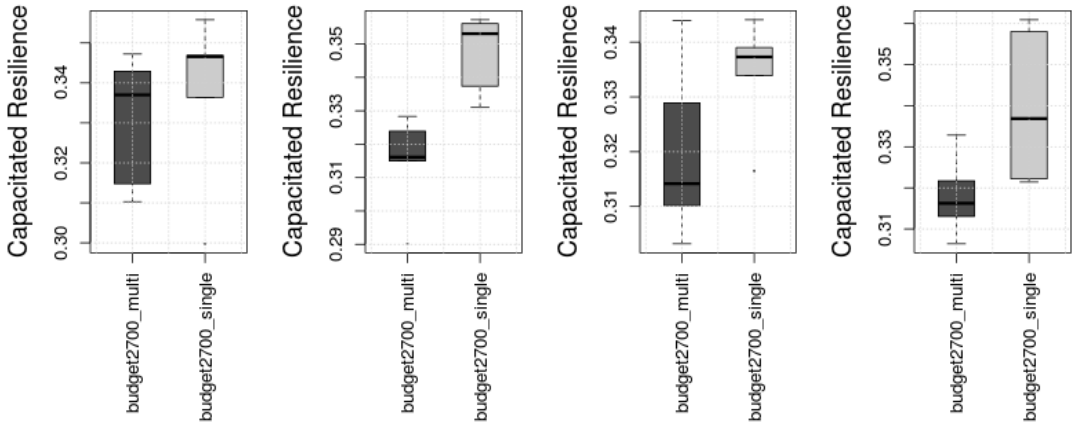


(e) Problem Instance 5 (f) Problem Instance 6 (g) Problem Instance 7 (h) Problem Instance 8

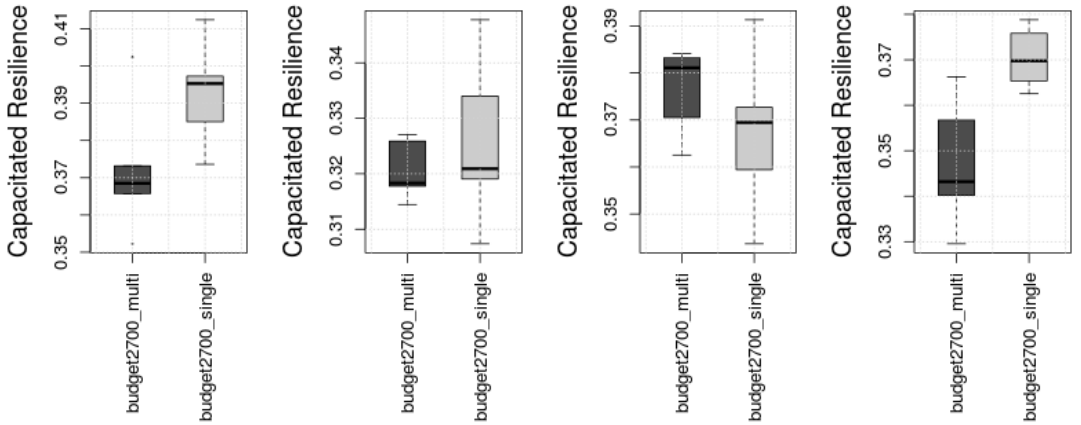


(i) Problem Instance 9 (j) Problem Instance 10

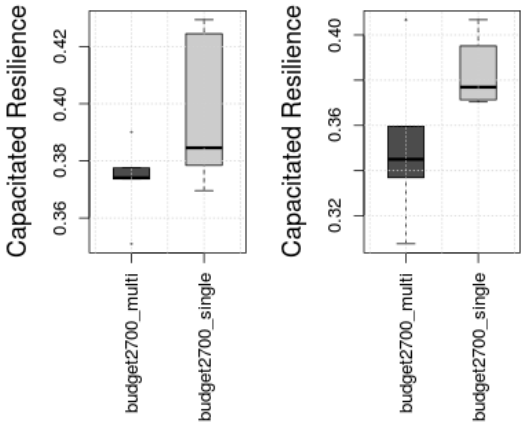
Figure 6.3: Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 50 user scenario



(a) Problem Instance 1 (b) Problem Instance 2 (c) Problem Instance 3 (d) Problem Instance 4

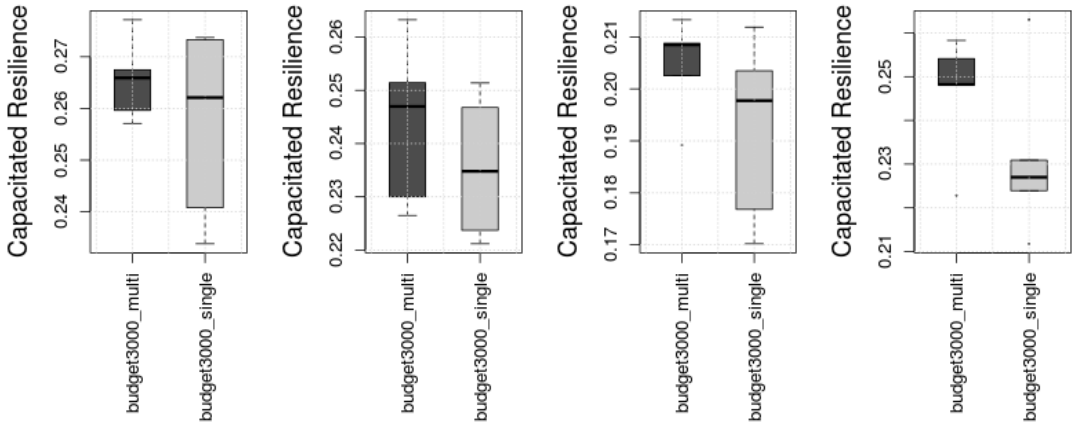


(e) Problem Instance 5 (f) Problem Instance 6 (g) Problem Instance 7 (h) Problem Instance 8

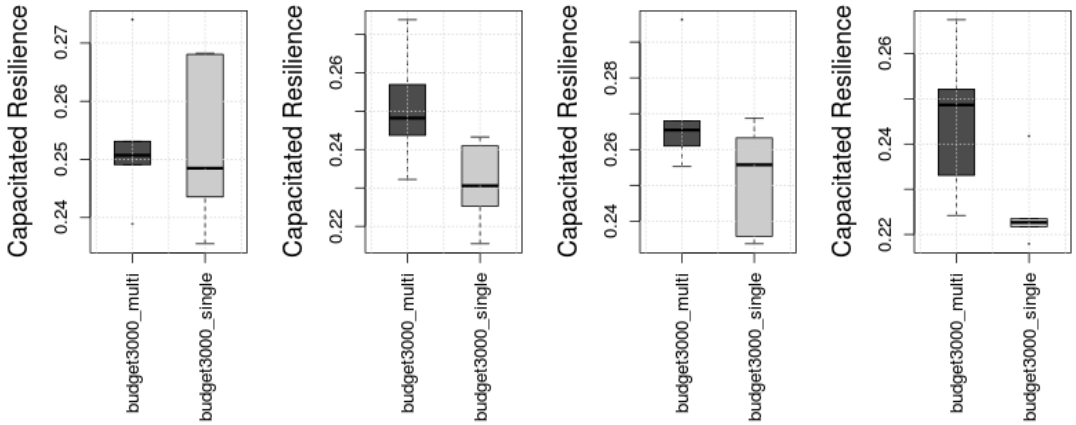


(i) Problem Instance 9 (j) Problem Instance 10

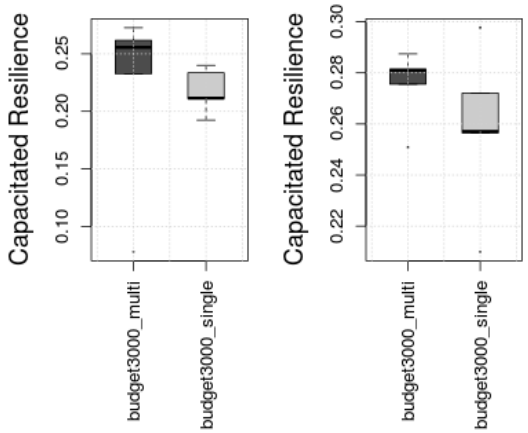
Figure 6.4: Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 75 user scenario



(a) Problem Instance 1 (b) Problem Instance 2 (c) Problem Instance 3 (d) Problem Instance 4



(e) Problem Instance 5 (f) Problem Instance 6 (g) Problem Instance 7 (h) Problem Instance 8



(i) Problem Instance 9 (j) Problem Instance 10

Figure 6.5: Comparison of single and bi-objective ES performance for capacitated resilience (using number of alternative paths and size of cut-sets as 10 and 4, respectively), 10 problem instances with 5 replications of the 100 user scenario

and all-terminal reliability is high because both capacitated resilience and all-terminal reliability are calculated by considering all paths from a user to all available APs. There is a similarity between their network designs such that both favor redundancy in their designs. Also, the correlation of two-terminal reliability and all-terminal reliability is high when optimizing for two-terminal reliability because two-terminal reliability is simply a subset of all-terminal reliability where only one AP is considered. However, the correlation between two-terminal and all-terminal reliabilities is low when optimizing for all-terminal reliability. Therefore, there is a difference between the designs of two-terminal and all-terminal reliabilities. Similarly, the correlation between capacitated resilience and two-terminal reliability is high when optimizing for capacitated resilience but the correlation is low when optimizing for two-terminal reliability. Hence, there are differences in their network designs. Also, the correlation between capacitated resilience and TE is low and there are significant differences in their network designs. The correlations among traffic efficiency, two-terminal and all-terminal reliabilities are high when optimizing for traffic efficiency, however, the correlations are low when optimizing for two-terminal or all-terminal reliabilities. Therefore, the network designs of traffic efficiency, two-terminal reliability and all-terminal reliability are different. The similarities and the differences in the network designs are discussed in detail in the Section 6.3.

Table 6.1: Correlations calculated using the best solutions of 10 problem instances of the 10 user scenario

	Correlation			
Optimized for	CR	TE	Two-term.	All-term.
CR	-	-0.058 *	0.9184	0.8841
TE	0.3075	-	0.6558	0.6925
Two-term.	0.2991	0.2135	-	0.9038
All-term.	0.7318	-0.1063	0.3712	-

* Correlation between capacitated resilience and traffic efficiency (when the network is designed for capacitated resilience)

6.3 Network structure: Comparison of capacitated resilience and other metrics

In this section, the differences between the network structures obtained by optimization for capacitated resilience and the other metrics are compared. For this comparison, different problem instances are solved for each metric. Input data of a problem instance consists of user locations and traffic requirements. In the proposed ES model of this dissertation, a user represents an area which may consist of some users. If the number of users in the optimization model were equal to the number in the physical world, then that model would have a real representation of users. However, this increases the number of users dramatically and therefore makes the problem intractable for real life applications. Hence, the traffic requirements of users represent the total traffic requirements of the users in an area. Also, because of this aggregation of data, the user locations are combined to a single location in the model for simplification.

The main goal of this section is to identify the key differences in the network structures in terms of number, types and locations of the devices. For this comparison 10 user and 25 user scenarios are used due to their small problem size (which helps to visually detect the differences in the network structures). The single objective ES was run for each metric and the resulting network structures are presented.

The network designs for capacitated resilience are obtained by two cases: unconstrained and constrained capacity cases. For the first case, capacities of APs and RPs are set to 150 and 80 (large enough to assume uncapacitated operation), respectively. For the constrained capacity case, capacities are set to 30 and 20, respectively. There are two cases for capacitated resilience for two reasons. The first is to make a fair comparison with other metrics, one basically uncapacitated and the other quite capacitated. Recall that the other metrics do not consider capacity. The second is to assess the effects of different levels of capacity constraint on the network designs.

6.3.1 A problem instance of the 10 user scenario

Figure 6.6 presents the input data of problem instance 1 of the 10 user scenario. The devices are deployed to optimize capacitated resilience, TE, two-terminal reliability and all-terminal reliability for the given user coordinates and traffic requirements.

Network structures obtained by optimization for capacitated resilience

Problem instance 1 of the 10 user scenario is optimized for capacitated resilience for two capacity cases. First, the unconstrained capacity case is solved for budget constraints of 500 and 600. Figure 6.7 shows the network structure obtained by optimization of problem instance 1 for capacitated resilience under a budget constraint of 500. The network structure shows the allocation of device redundancy in the high traffic requirements areas. U0, U1 and U8 are the low traffic flow users (which have smaller weights in the capacitated resilience calculation), therefore they are not prioritized by the ES. In other words, APs (or RPs) are not located near these users because their effect on the capacitated resilience is limited due to their low traffic requirements (see Equation 3.1 for the definition of user weight). For example, the two APs (AP5 and AP8) are not located close to U0 due to its low traffic requirement. Also, AP5 is located far from U5 but close to U9. The reason is to increase the reliability of the alternative path (U9 - AP5) without losing connectivity of U5 and U0. The traffic requirements of U9 and U5 are very high (14.803 and 15.753, respectively) and it might be argued that the location of AP8 nearer U5 would provide a slightly higher capacitated resilience because of its higher traffic requirement. However, this improvement would be very limited due to the small difference between the traffic requirements of U9 and U5. If the budget was larger (e.g., 600) capacitated resilience would be increased by adding a second AP close to U5 to improve its assigned path and use AP5 as its alternative path. Similarly the ES preferred to allocate redundancy for U2 and U3 instead of U7 and U8. This decision is reasonable because U8 is a low traffic node (3.737) and U3 and U7 are comparable in terms of their traffic requirements (14.729 and 14.269). Also, locating redundancies around

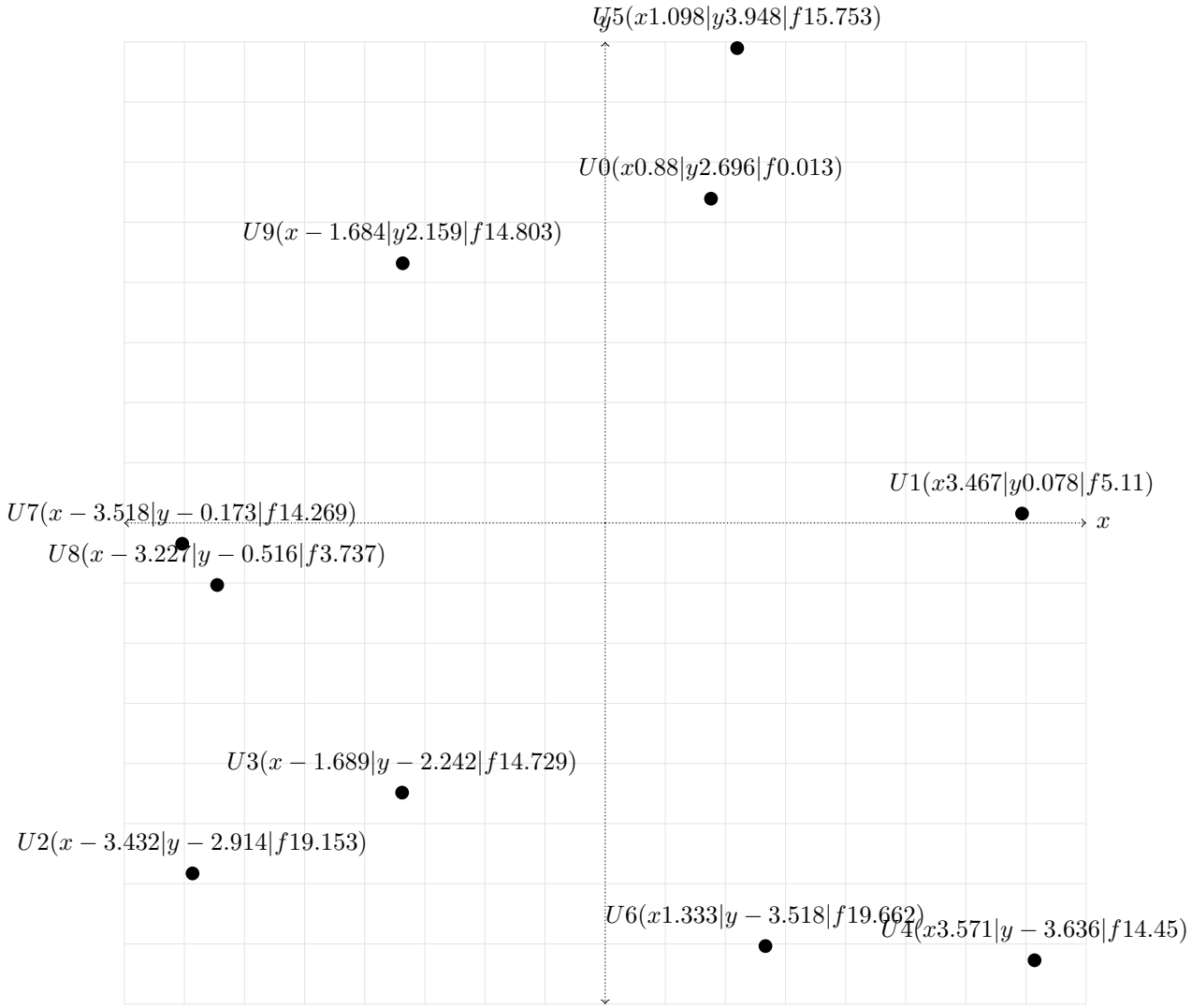


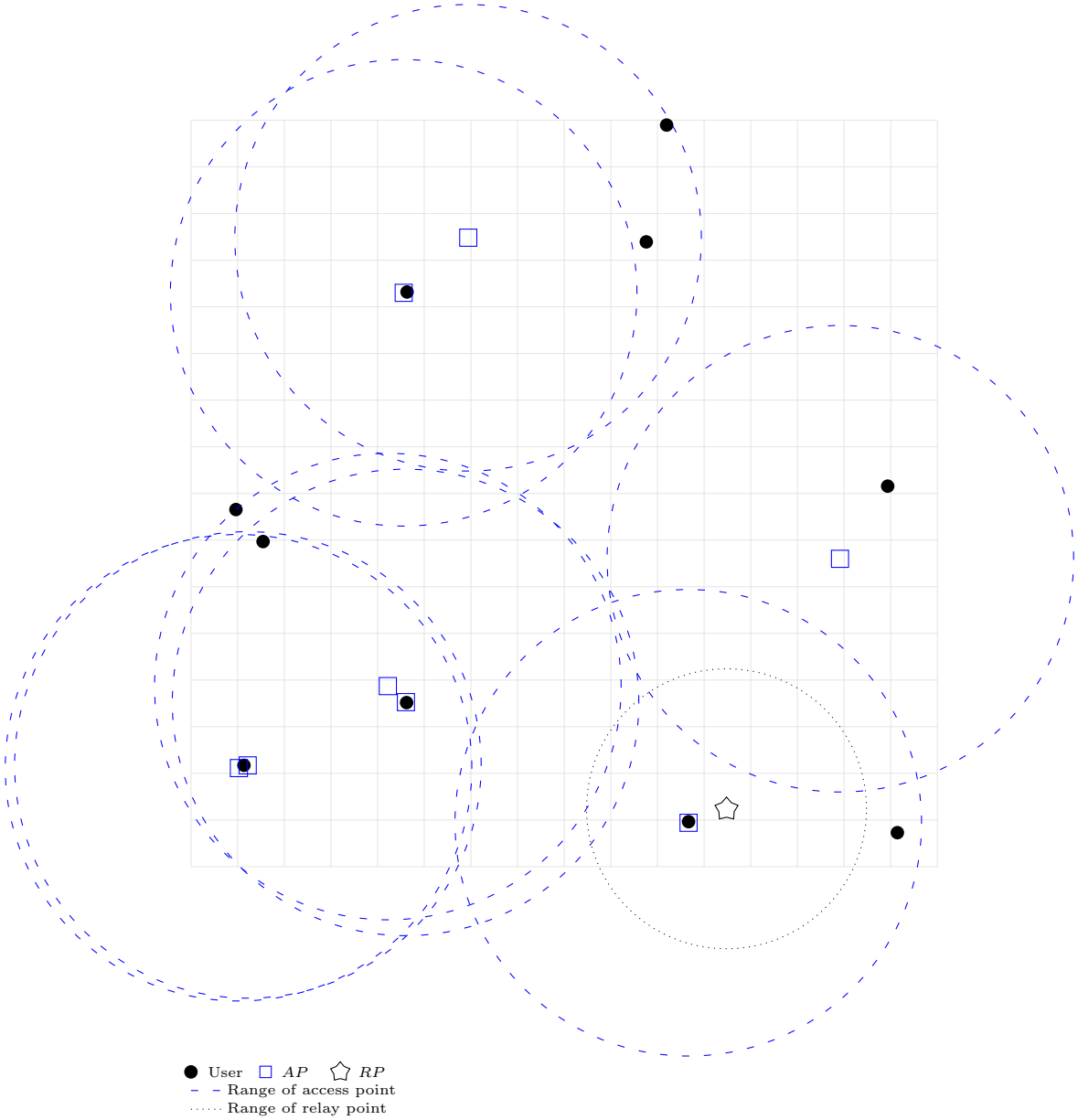
Figure 6.6: Inputs of 10 users scenario (problem instance 1): User locations (x and y are the coordinates, f is the traffic flow requirement)

U2 and U3 helps to further improve capacitated resiliences of those nodes because the nodes are located close to each other and this improves the network capacitated resilience. The capacitated resilience of this network is 0.5054 and the TE is 0.3043.

Figure 6.8 shows the network structure of the same problem with a larger budget (600 instead of 500). As the budget increases, redundancies are added to U0, U4, U5, U6, U7 and U8. On the other hand, U9 has lost its redundant path due to this new arrangement of the devices. However, since the traffic flow requirements of U7 and U9 are comparable (14.269 and 14.803, respectively) this new arrangement does not have an adverse effect on the network level capacitated resilience. Also, locating devices in the more crowded areas (in this case, the areas of U7, U8, U3 and U2) helps improving the overall capacitated resilience. Note that U1 still has no redundancy, which is similar to the budget=500 scenario, due to its low traffic requirement (which is 5.11).

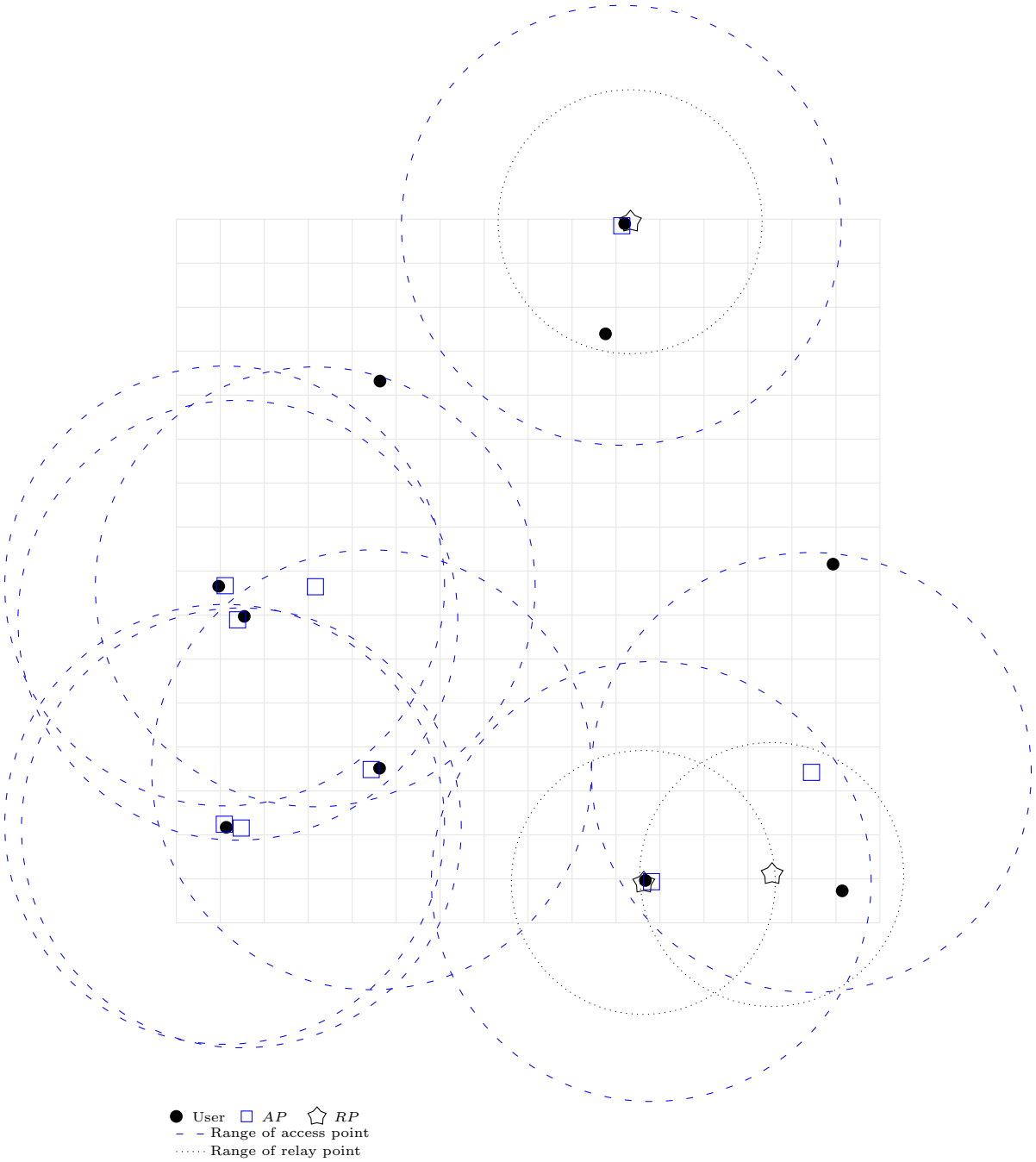
Figures 6.9 and 6.10 show the solutions for constrained capacity case. For the same problem instance, the capacity of the APs and RPs are set to 30 and 20, respectively. Both network designs, with budget=500 (Figure 6.9) and budget=600 (Figure 6.10) have lower capacitated resilience values than the previous designs (Figures 6.7 and 6.8) due to the limited capacity. However, the designs are similar because the users are connected with an AP and then the redundancies are provided for high traffic users. Isolated users (such as U0, U1, U4 and U5 of Figure 6.9) are connected with distant APs that are primarily serving the high traffic users. For the budget constraint of 600 case (Figure 6.10), similar design rules have been observed. High traffic user (e.g., U5 and U7) have strong connections with a high level of redundancy ensured by nearby APs. Isolated users (such as, U1) have no or limited redundancy. The capacity constraint does not change the design rules.

One of the common design rules from the network structure obtained by optimizing for capacitated resilience is that the redundancy is allocated to the most promising area (crowded or having larger traffic nodes) instead of to isolated nodes or low traffic nodes. As the capacitated resilience of a user becomes zero if there are no alternative paths available,



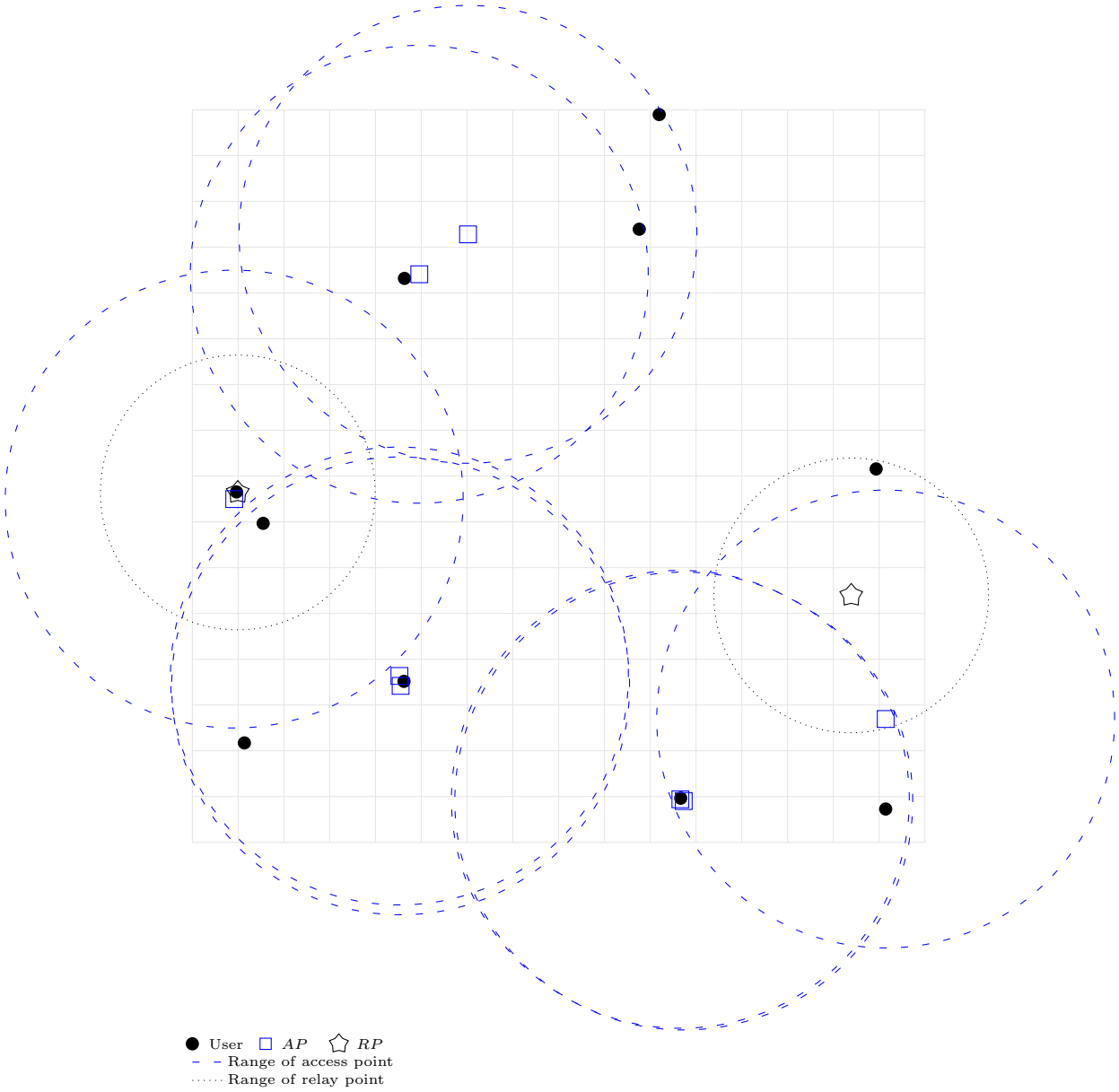
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
490, 0.5054, 0.3043, 0.604, 0.6114
# of APs = 8, # of RPs = 1

Figure 6.7: Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for Capacitated Resilience with unconstrained capacities



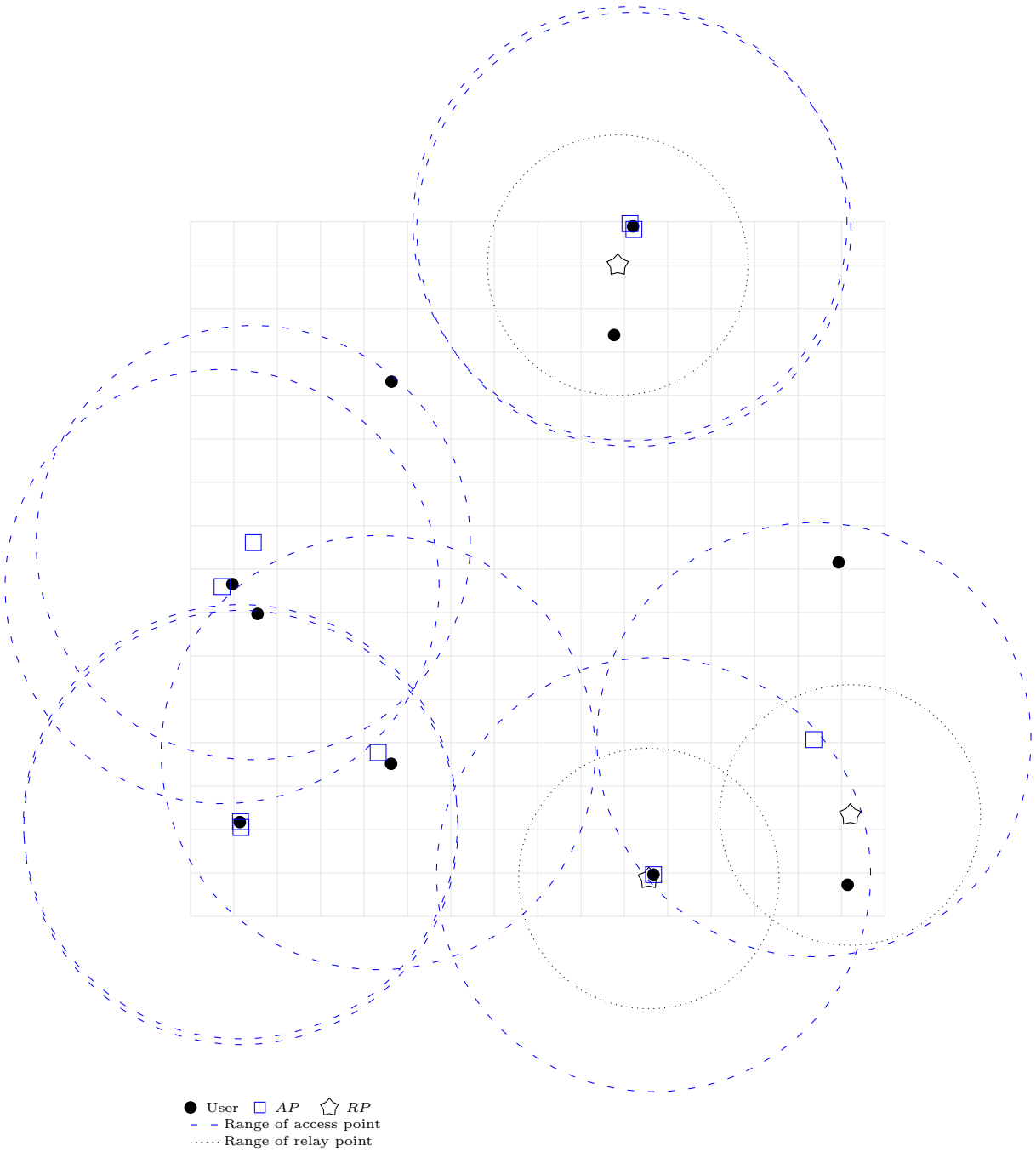
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
570, 0.6429, 0.642, 0.7675, 0.7875
# of APs = 9, # of RPs = 3

Figure 6.8: Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for Capacitated Resilience with unconstrained capacities



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
500, 0.4975, 0.4502, 0.6556, 0.7047
# of APs = 8, # of RPs = 2

Figure 6.9: Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for Capacitated Resilience with constrained capacities



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
570, 0.5839, 0.5826, 0.7408, 0.7651
# of APs = 9, # of RPs = 3

Figure 6.10: Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for Capacitated Resilience with constrained capacities

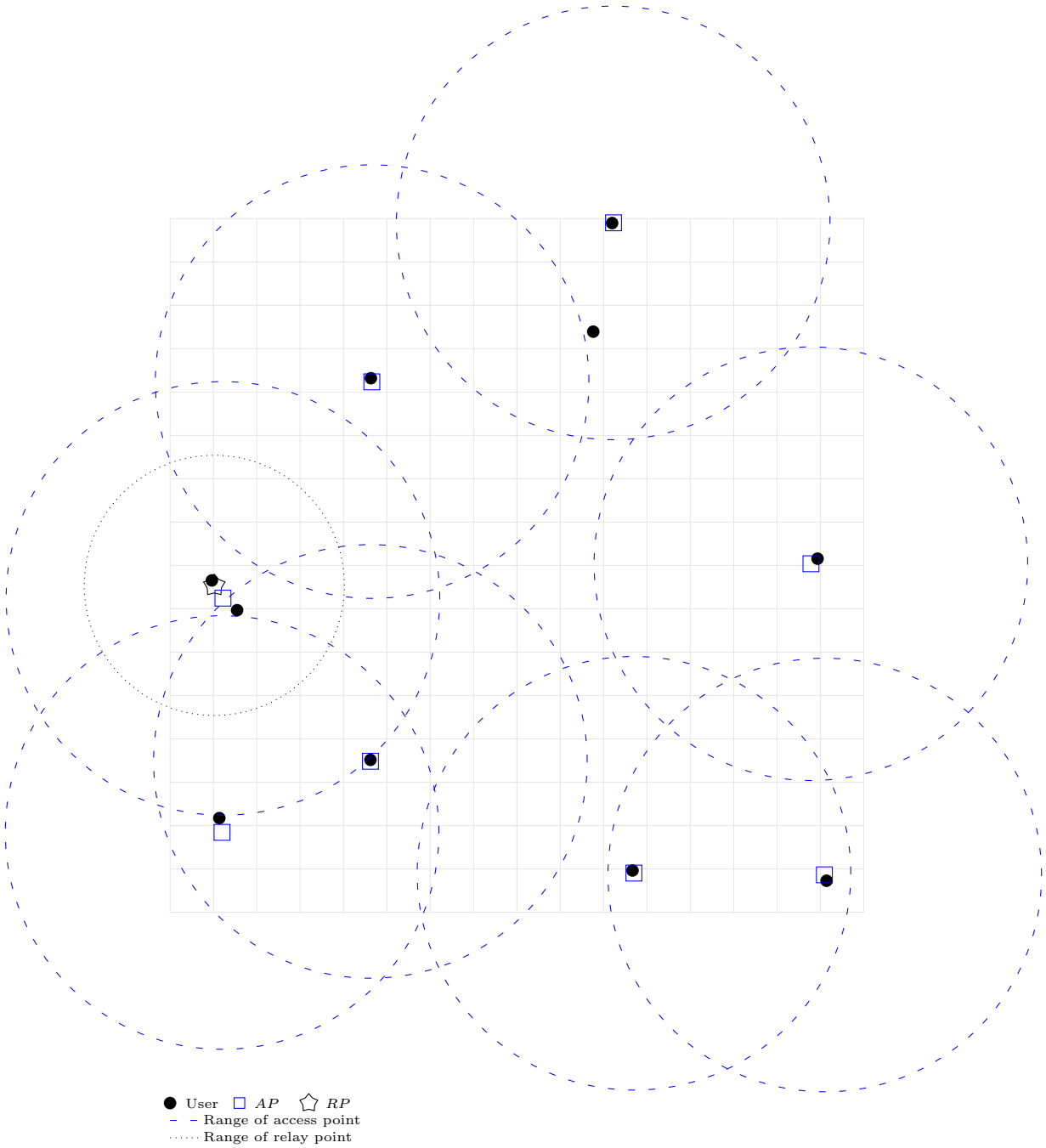
the location of the assigned devices are strategically made to improve reliabilities of the alternative paths of other users (see U1 and AP0 in Figure 6.8).

Capacitated resilience vs. TE

Figure 6.11 shows the network structure generated by traffic efficiency optimization under budget constraint of 500. The major difference of this network between the one generated by capacitated resilience is the redundancy allocation. In the one generated by TE optimization, there is no or very limited redundancy in the network. Instead an AP is located for most users. For the users that cannot have a "dedicated" access point, they share an access point which is close to another user. For example, U0 and U8 are the low traffic nodes and they are connected to the network by a distant AP. However, this decision of assigning the dedicated APs to high traffic nodes is needed to maximize TE.

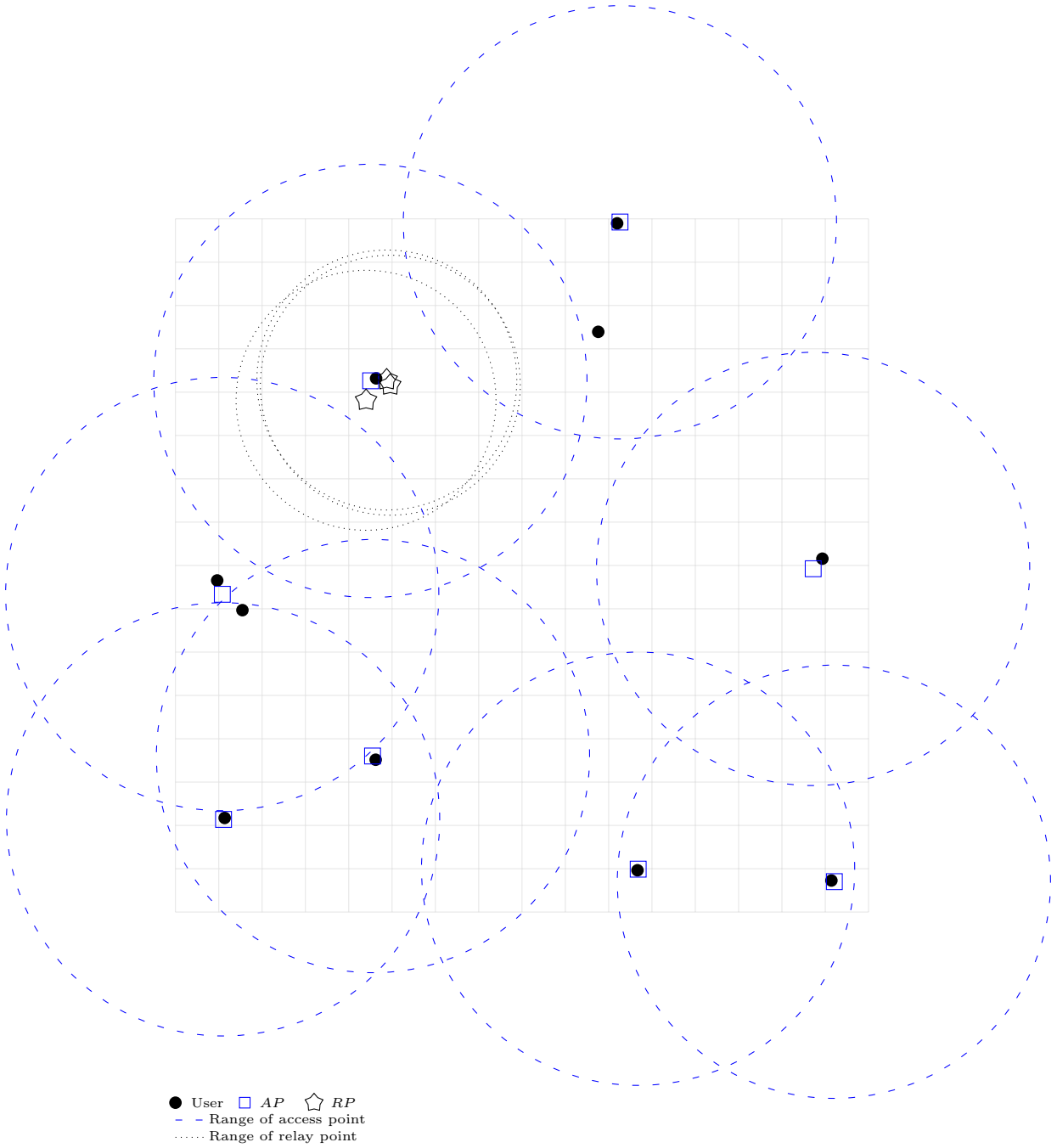
In terms of redundancy, the network has very limited redundancy but high reliability for the node connections. Only U2, U3, U4, U6, U7 and U8 have alternative paths. The other users have no alternative paths, therefore their capacitated resilience values are zero. The capacitated resilience and TE of this network are 0.2056 and 0.8577, respectively. In comparison to the network optimized by capacitated resilience (Figure 6.7), TE improved from 0.3043 to 0.8577, but the capacitated resilience reduced dramatically from 0.5054 to 0.2056 due to lack of alternative paths in the network. This difference is caused by the structural differences between the two networks.

Figure 6.12 has a larger budget (600 instead of 500), but the structure of the resulting network is similar to the one of smaller budget (500). The devices are located very close to users so that the reliabilities between users and APs are very high. Only U0 does not have an AP located nearby, but U0 has a very small traffic requirement (0.023) therefore it does not have a significant effect on the TE. The capacitated resilience of the network is 0.2039, which is similar to one given in Figure 6.11, but the TE has improved from 0.8577 to 0.9854 because of the larger number of RPs and their locations.



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
490, 0.2056, 0.8577, 0.9759, 0.9795
# of APs = 8, # of RPs = 1

Figure 6.11: Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for TE



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
510, 0.2039, 0.9854, 0.977, 0.9787
# of APs = 8, # of RPs = 3

Figure 6.12: Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for TE

In summary, the typical network structure obtained by TE optimization is to have high reliability for high traffic nodes by locating APs near them, and to have some redundancy for high traffic nodes within the given budget and allow low reliability connections for low traffic nodes (if the budget is tightly constrained).

Capacitated resilience vs. two-terminal reliability

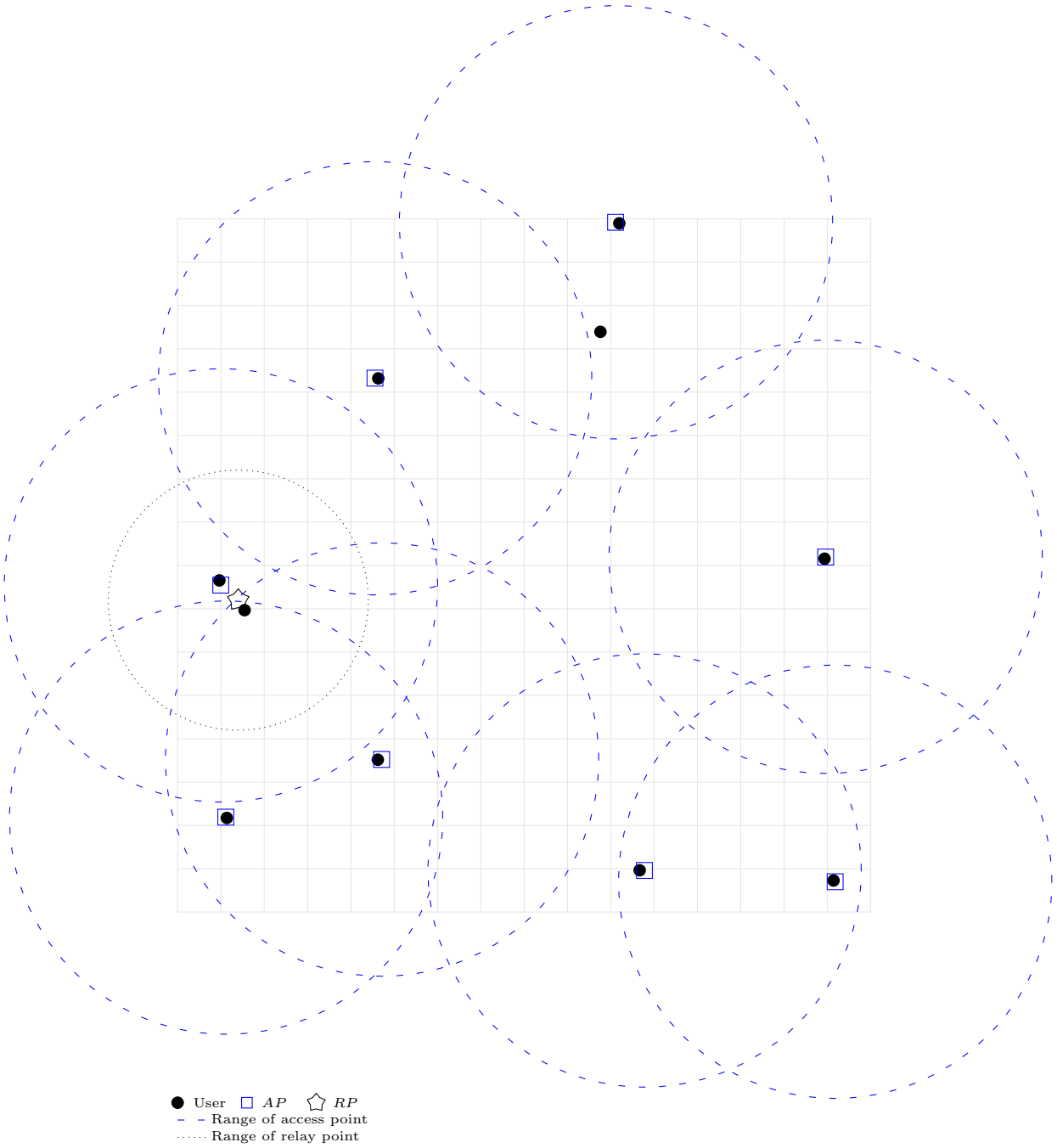
As seen in Figure 6.13, which is similar to the structure obtained by TE optimization, the APs (or RPs) are located very close to the users to get maximum reliability. U0 is assigned to an AP which is not very close to its location, (which was also seen in Figure 6.12), but it does not affect two-terminal reliability significantly due to its low weight in the network level two-terminal reliability calculation. The two-terminal reliability of this network is 0.9802 and capacitated resilience is 0.1054. Similar to TE optimization, the main reason of the low capacitated resilience value for two-terminal optimization is the lack of redundancy in the network.

As the budget increases from 500 to 600 (Figure 6.14), an AP has been assigned to the U0 user to increase overall two-terminal reliability. Also, an additional device has been located near U6, which is a high traffic node with traffic requirement of 19.662. This improved capacitated resilience from 0.1054 to 0.3695, but two-terminal reliability improved slightly (from 0.9802 to 0.9831). The network structure remained similar.

Thus, the structure obtained by two-terminal reliability optimization is very similar to the one obtained by TE optimization. It allocates APs very close to the high traffic nodes. If the budget is not available, the low traffic nodes are connected to a distant AP which serves a nearby high traffic user.

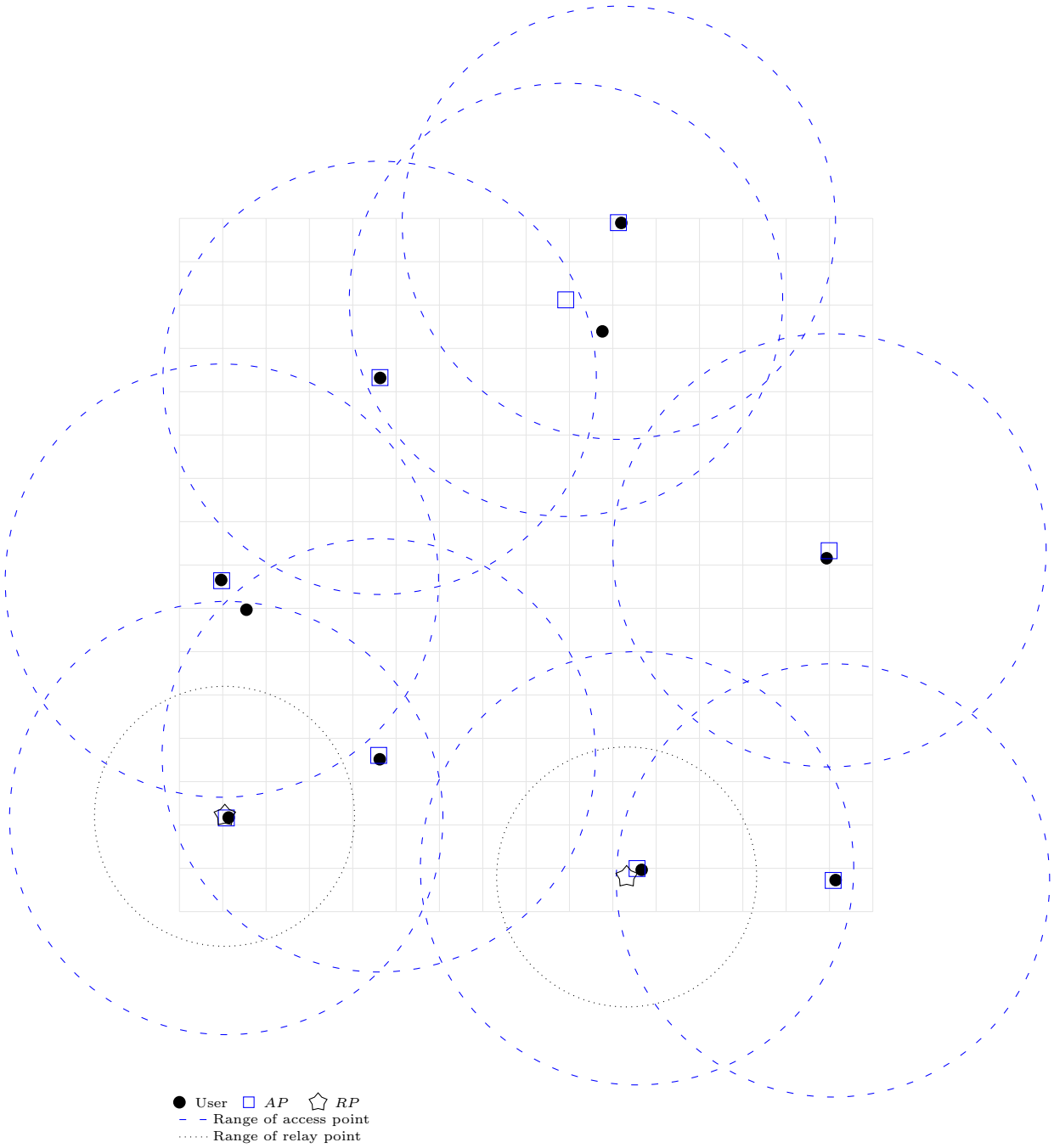
Capacitated resilience vs. all-terminal reliability

The network structure found by all-terminal reliability optimization is very similar to the one of two-terminal reliability. With a budget constraint of 500, APs are located very



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
490, 0.1910, 0.7134, 0.9855, 0.9869
# of APs = 8, # of RPs = 1

Figure 6.13: Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for two-terminal reliability



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
560, 0.3695, 0.8565, 0.9861, 0.9862
# of APs = 9, # of RPs = 2

Figure 6.14: Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for two-terminal reliability

close to high traffic nodes (Figure 6.15). When the budget increases to 600 (Figure 6.16), some redundancies are added to the high traffic nodes and all-terminal reliability increases from 0.9831 to 0.9877. Not surprisingly, capacitated resilience has increased from 0.2836 to 0.3298 because of the new alternative paths.

It can be said that the network structure obtained by optimization for all-terminal reliability is very similar to the one obtained from two-terminal reliability optimization. However, the redundancies for all-terminal reliability are more decentralized. For example, in Figure 6.16, AP2 and AP12 provide redundancies for all users on the left side (i.e., $x < 0$) of the network. This is different than Figure 6.14 in which the redundancies are located near the users for two-terminal maximization.

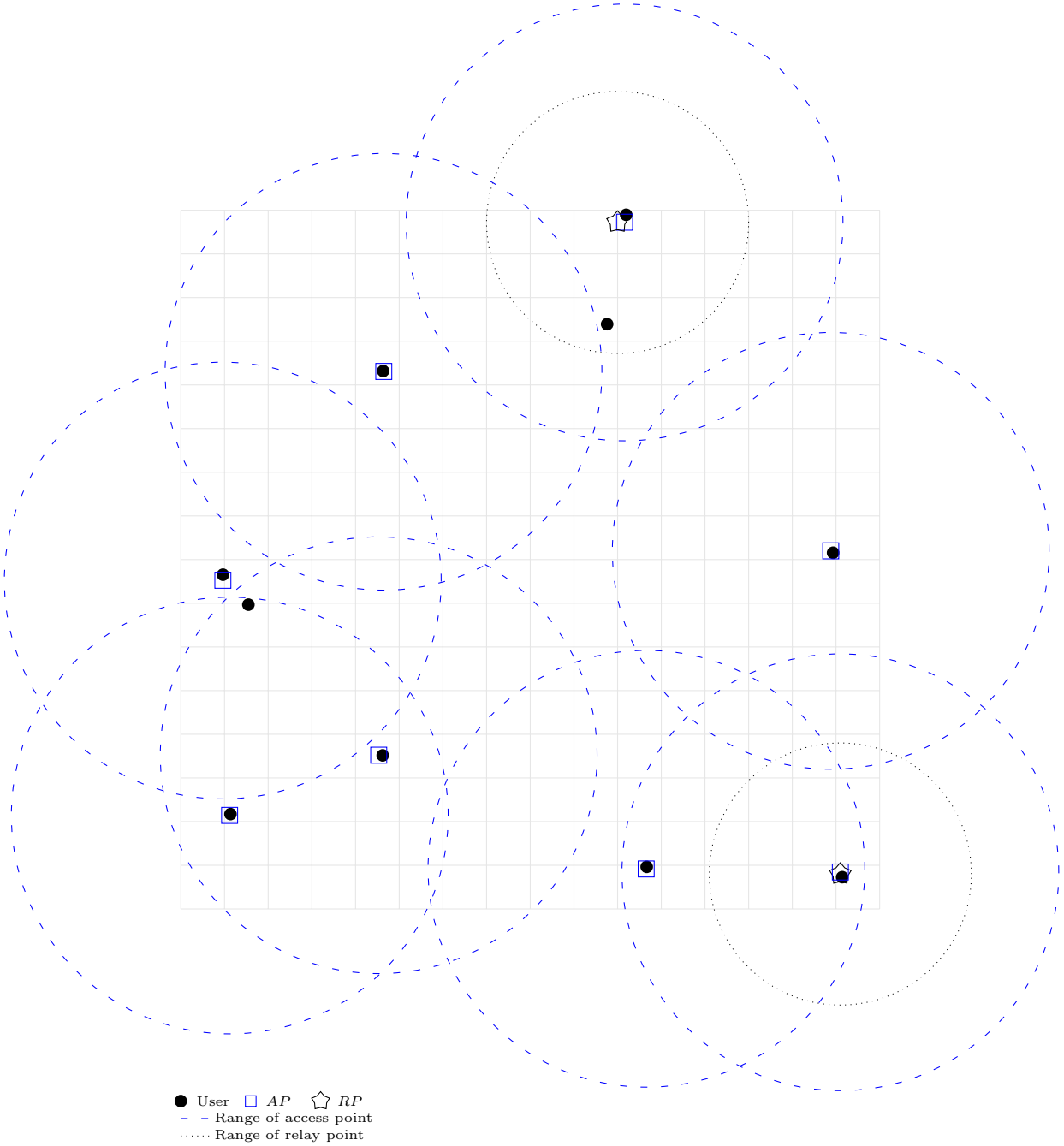
Table 6.2 summarizes the values of all metrics for all network designs presented in this section. Figure 6.17 and 6.18 summarize the comparison of the network designs presented in this section.

Table 6.2: Summary of all metrics for the 10 user scenario, problem instance 1

		Results			
Budget	Optimized by	CR	TE	2-term	All-term
500	CR (unconstrained capacities)	0.5054	0.3043	0.6040	0.6114
	CR	0.4975	0.4502	0.6556	0.7047
	TE	0.2056	0.8577	0.9759	0.9795
	Two-terminal rel. (2-term)	0.1910	0.7134	0.9855	0.9869
	All-terminal rel. (all-term)	0.3284	0.7238	0.9852	0.9871
600	CR (unconstrained capacities)	0.6429	0.6420	0.7675	0.7875
	CR	0.5839	0.5826	0.7408	0.7651
	TE	0.2039	0.9854	0.9770	0.9787
	Two-terminal rel. (2-term)	0.3695	0.8565	0.9861	0.9862
	All-terminal rel. (all-term)	0.3298	0.7115	0.9720	0.9877

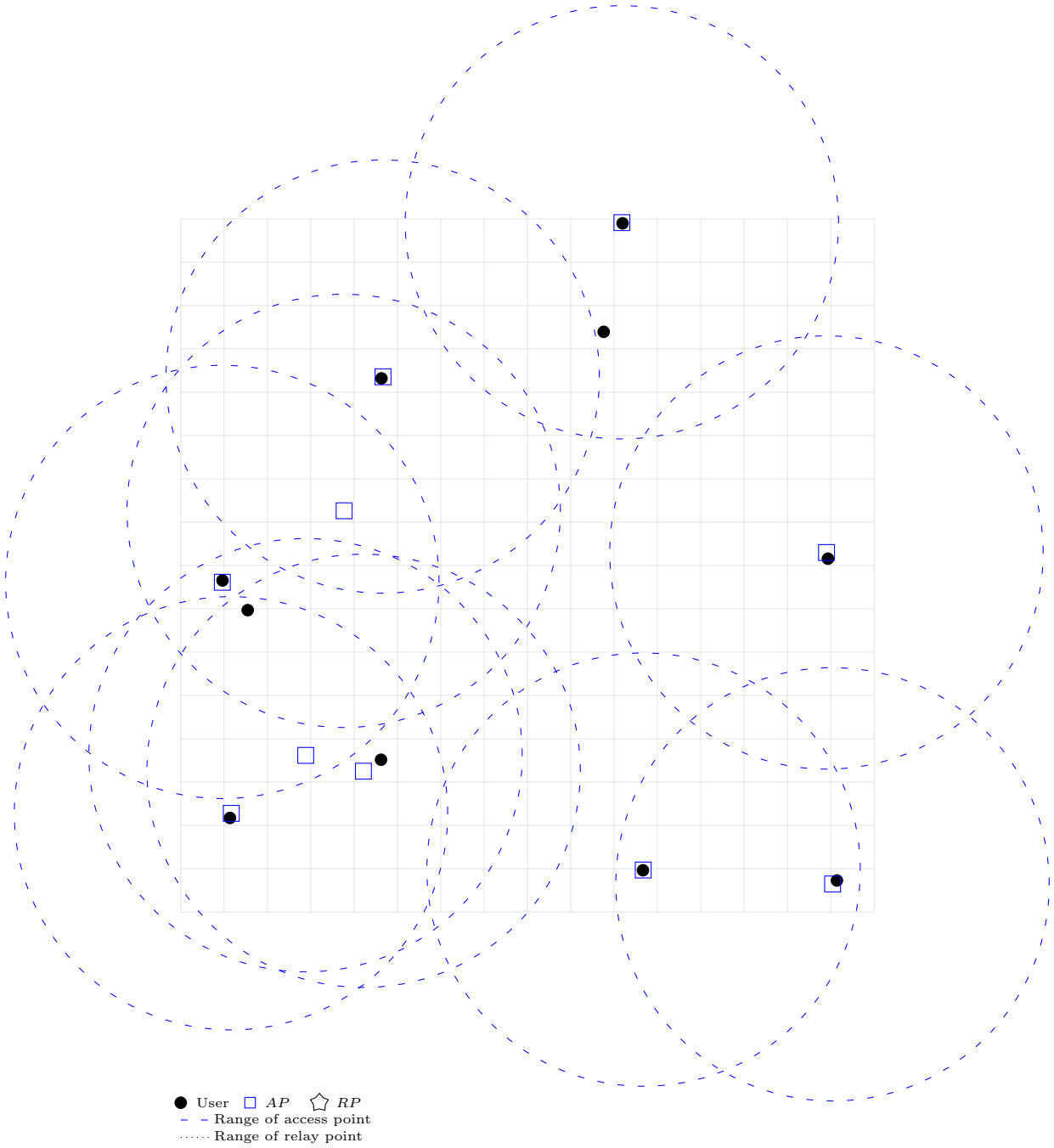
6.3.2 Another problem instance of the 10 user scenario

In this section the network structures obtained by optimization for different metrics are compared using another problem instance of the 10 user scenario. The location and flows of the users are given in the Figure 6.19.



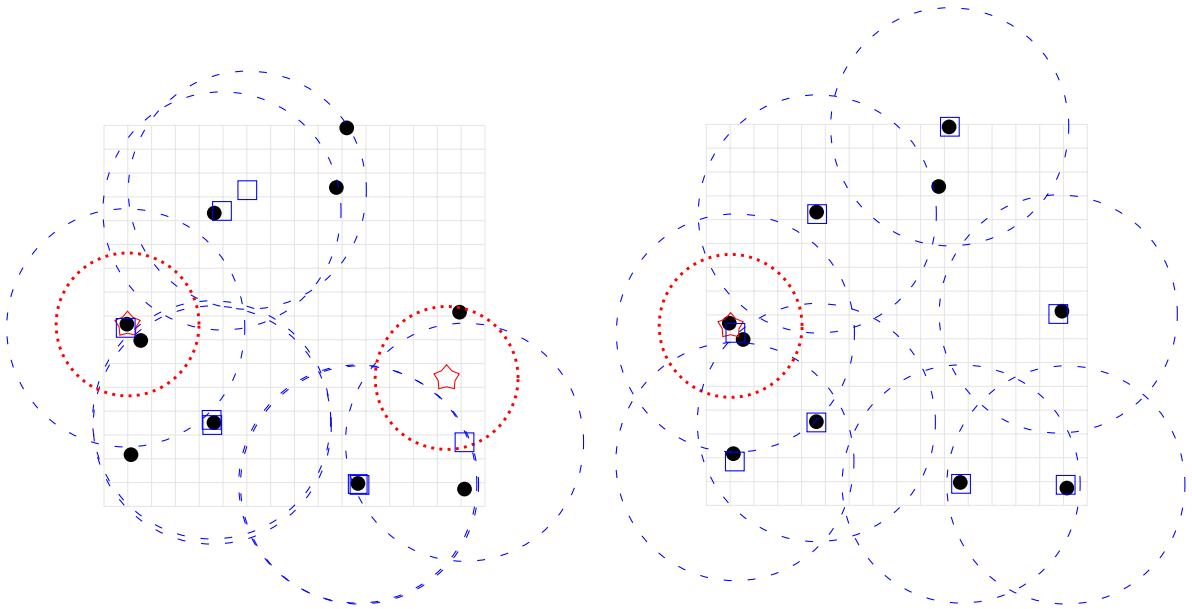
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
500, 0.3284, 0.7238, 0.9852, 0.9871
# of APs = 8, # of RPs = 2

Figure 6.15: Network structure of the 10 user problem (problem instance 1, budget=500) found by optimization for all-terminal reliability



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
600, 0.3298, 0.7115, 0.972, 0.9877
# of APs = 10, # of RPs = 0

Figure 6.16: Network structure of the 10 user problem (problem instance 1, budget=600) found by optimization for all-terminal reliability



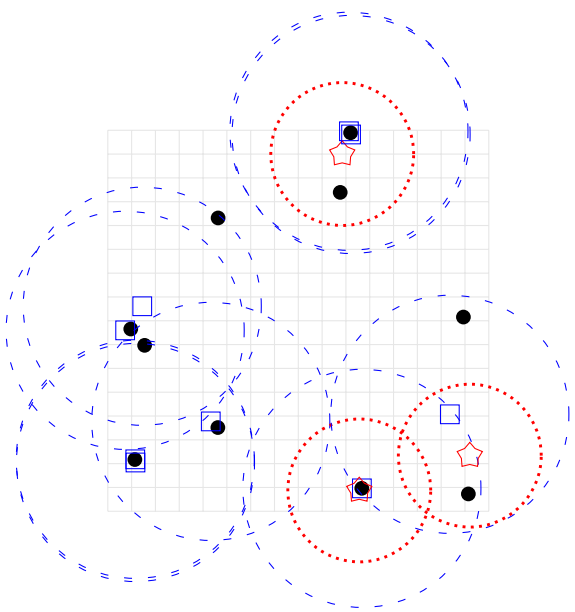
(a) Design for CR

(b) Design for TE

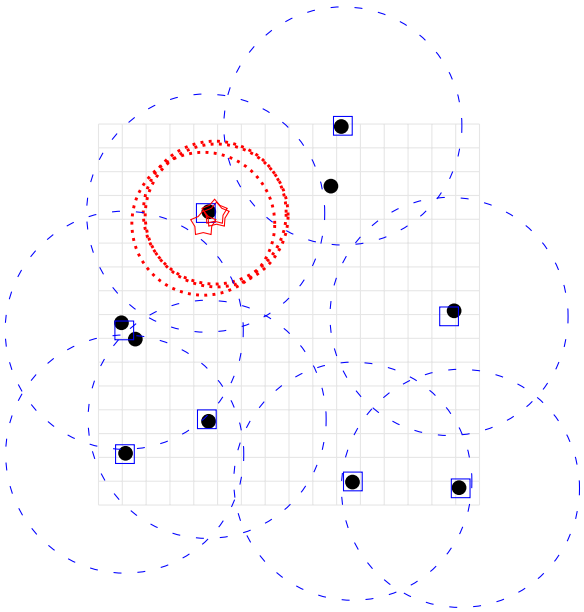
(c) Design for two-term. rel.

(d) Design for all-term. rel.

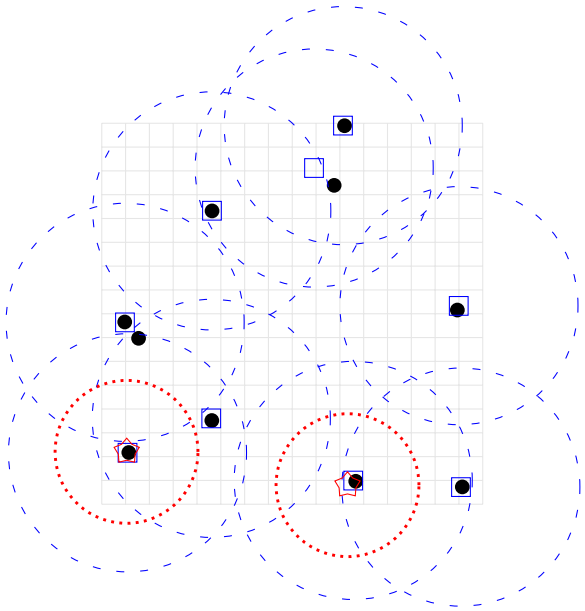
Figure 6.17: Summary of the designs of the problem instance 1 of the 10 user scenario, budget=500



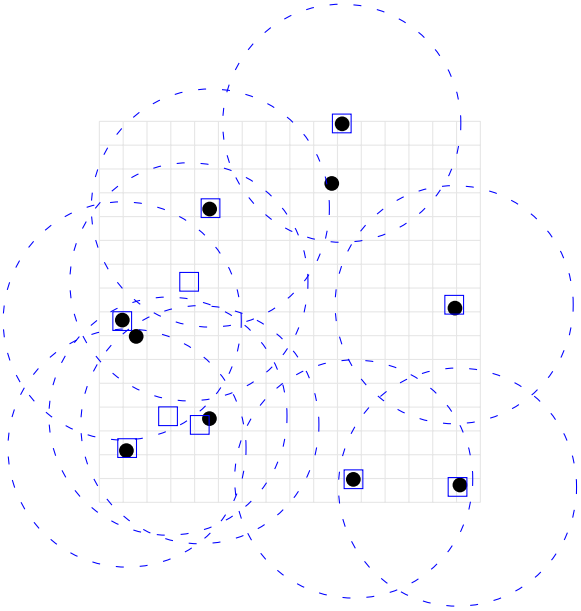
(a) Design for CR



(b) Design for TE



(c) Design for two-term. rel.



(d) Design for all-term. rel.

Figure 6.18: Summary of the designs of the problem instance 1 of the 10 user scenario, budget=600

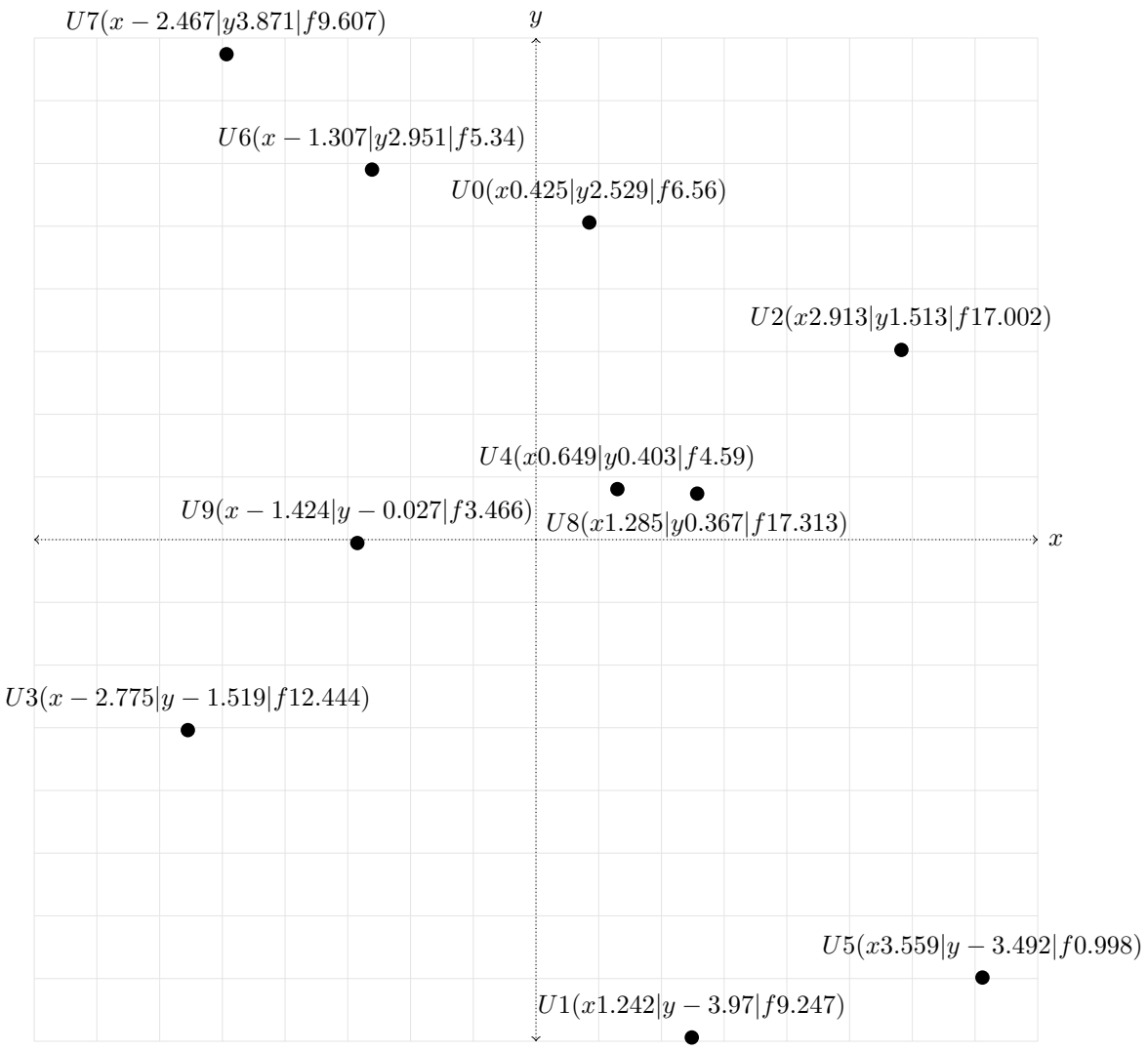


Figure 6.19: Inputs of the 10 user scenario (problem instance 2): User locations (x and y are the coordinates, f is the traffic flow requirement)

Figure 6.20 shows the network optimized for maximum capacitated resilience. Similar to the network design found for problem instance 1 (Section 6.3.1) this network emphasises strong connections for high flow nodes and redundancies around those nodes. Similarly, the constrained capacity case of the same problem instance (solved for capacitated resilience) yields a similar network design (Figure 6.21) with slightly less capacitated resilience value (0.6753 instead of 0.7461) due to limited capacities. On the other hand, the network structure of TE (Figure 6.22) focuses on strong connections for most of the nodes. Redundancy is not as important as for capacitated resilience.

The networks found for two-terminal and all-terminal reliabilities (Figures 6.23 and 6.24) provide strong connections for high flow nodes, but also some level of redundancy are allocated by RPs. In capacitated resilience, the redundancies for high flow nodes are mostly ensured by APs but the two-terminal and all-terminal reliability designs use RPs to provide redundancies in this problem instance because APs are mainly used to create strong connections (between users and APs) for two and all-terminal reliability designs and RPs provide alternative paths at a lower cost.

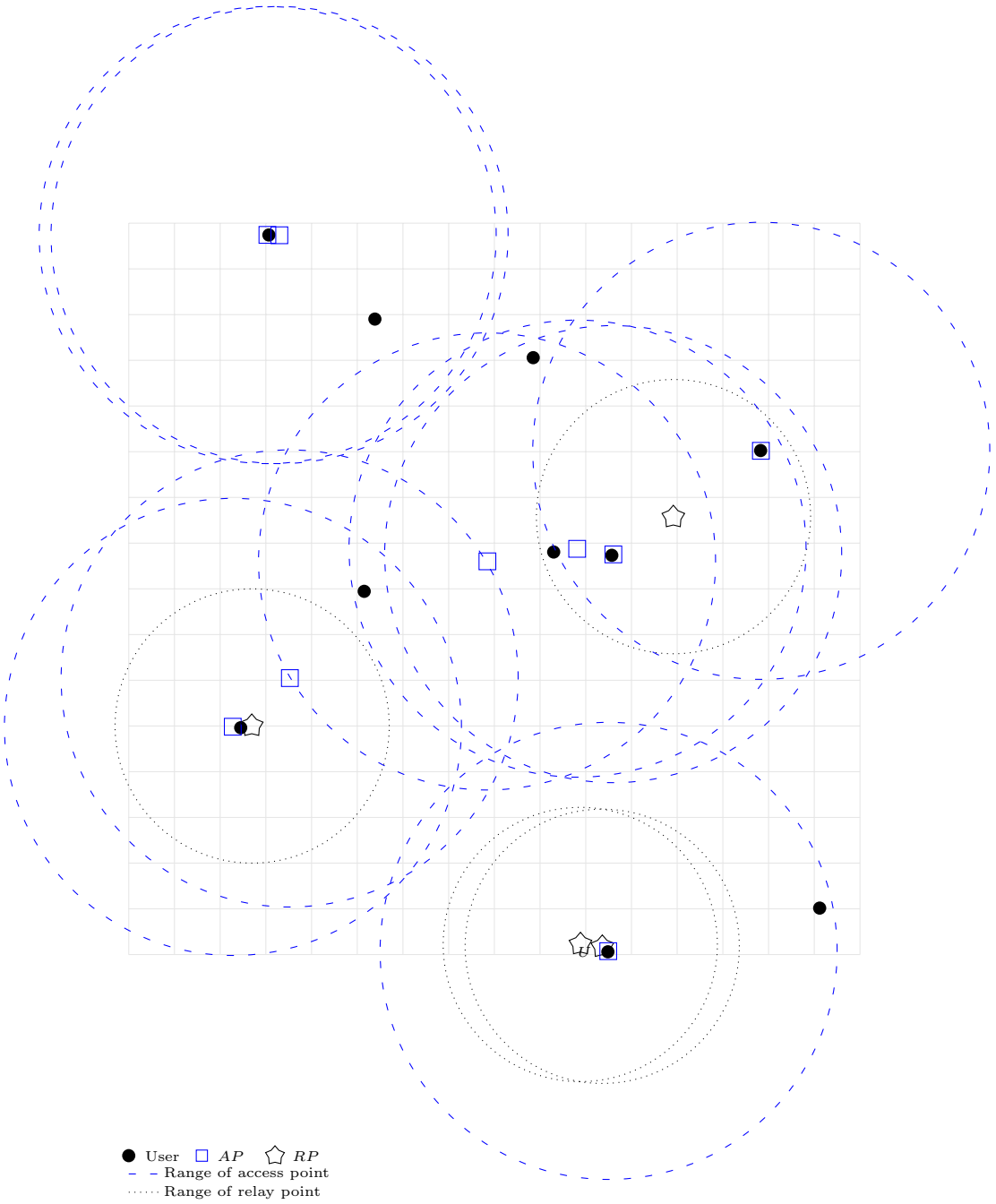
The summary of the network designs is given in Table 6.3. The designs presented in this section are compared in Figure 6.25.

Table 6.3: Summary of all metrics for the 10 user scenario, problem instance 2

		Results			
Budget	Optimized by	CR	TE	2-term	All-term
600	CR (unconstrained capacities)	0.7461	0.6308	0.8169	0.8363
	CR	0.6753	0.4523	0.7830	0.8306
	TE	0.3655	0.9101	0.9036	0.9412
	Two-terminal rel. (2-term)	0.5248	0.5361	0.9787	0.9844
	All-terminal rel. (all-term)	0.5210	0.8247	0.9764	0.9845

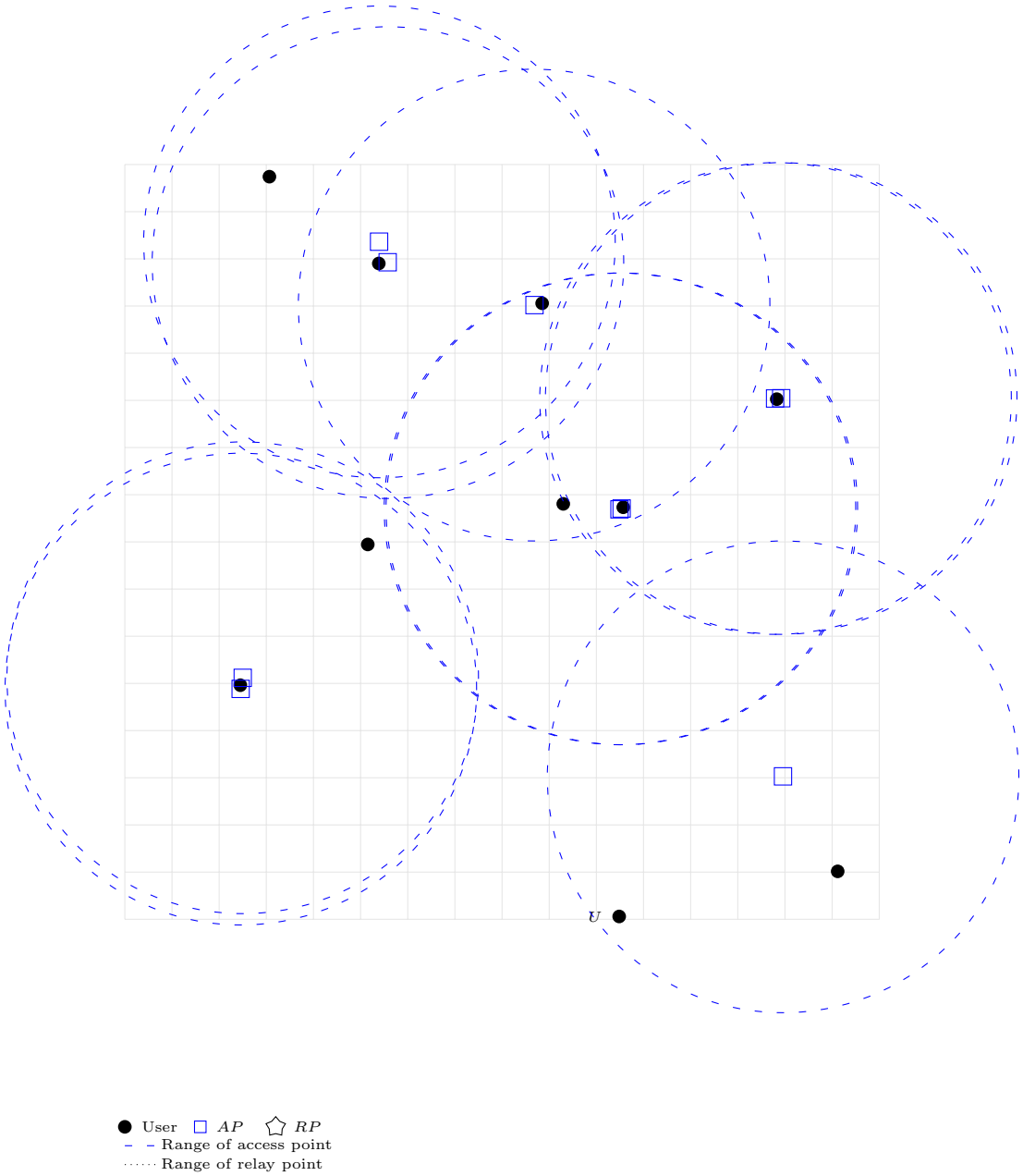
6.3.3 A problem instance of the 25 user scenario

As another example of differences the network design, a larger problem is presented in this section. Figure 6.26 shows the user locations and traffic requirements of the 25 user



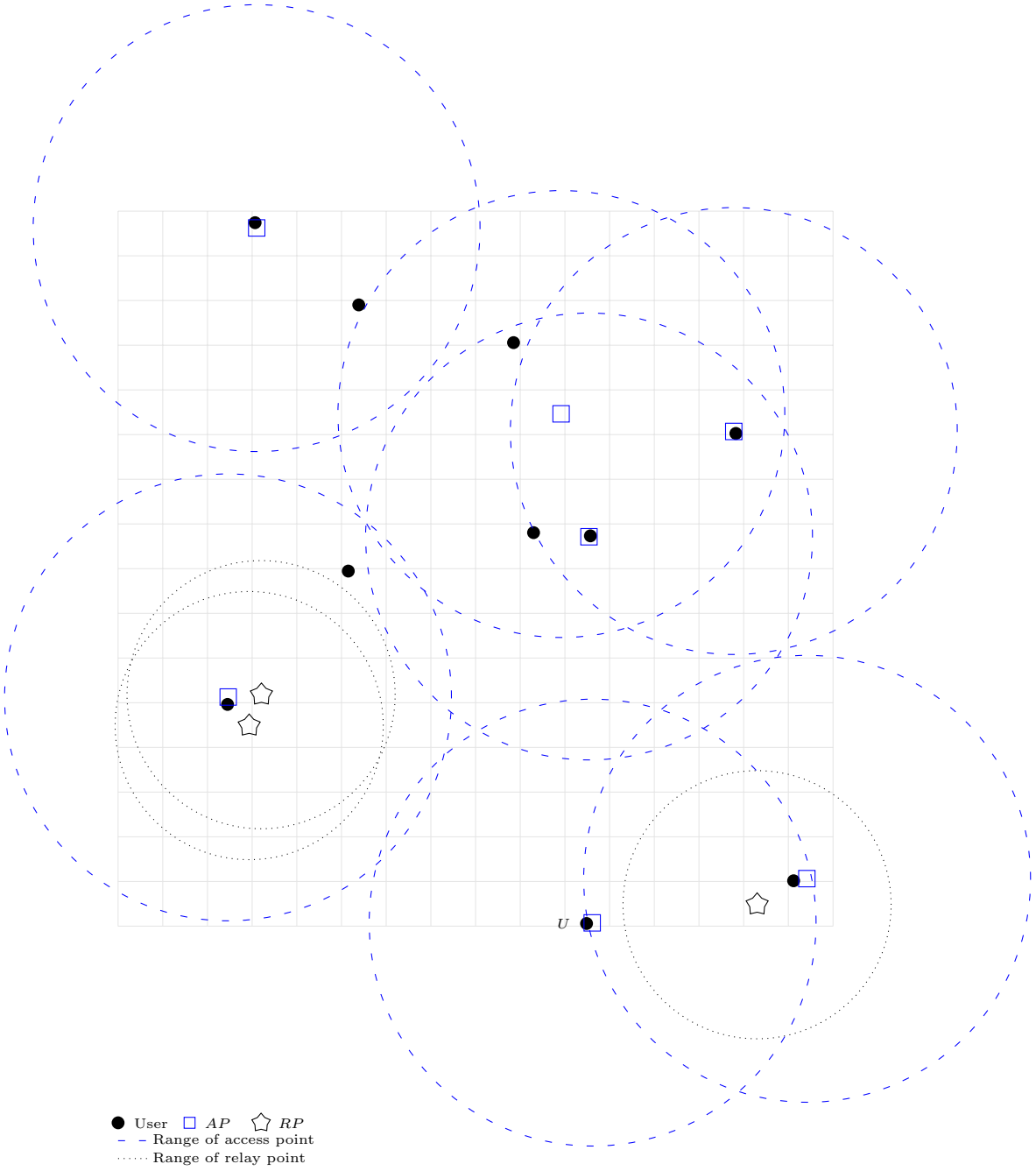
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
580, 0.7461, 0.6308, 0.8169, 0.8363
# of APs = 9, # of RPs = 4

Figure 6.20: Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for Capacitated Resilience with unconstrained capacities



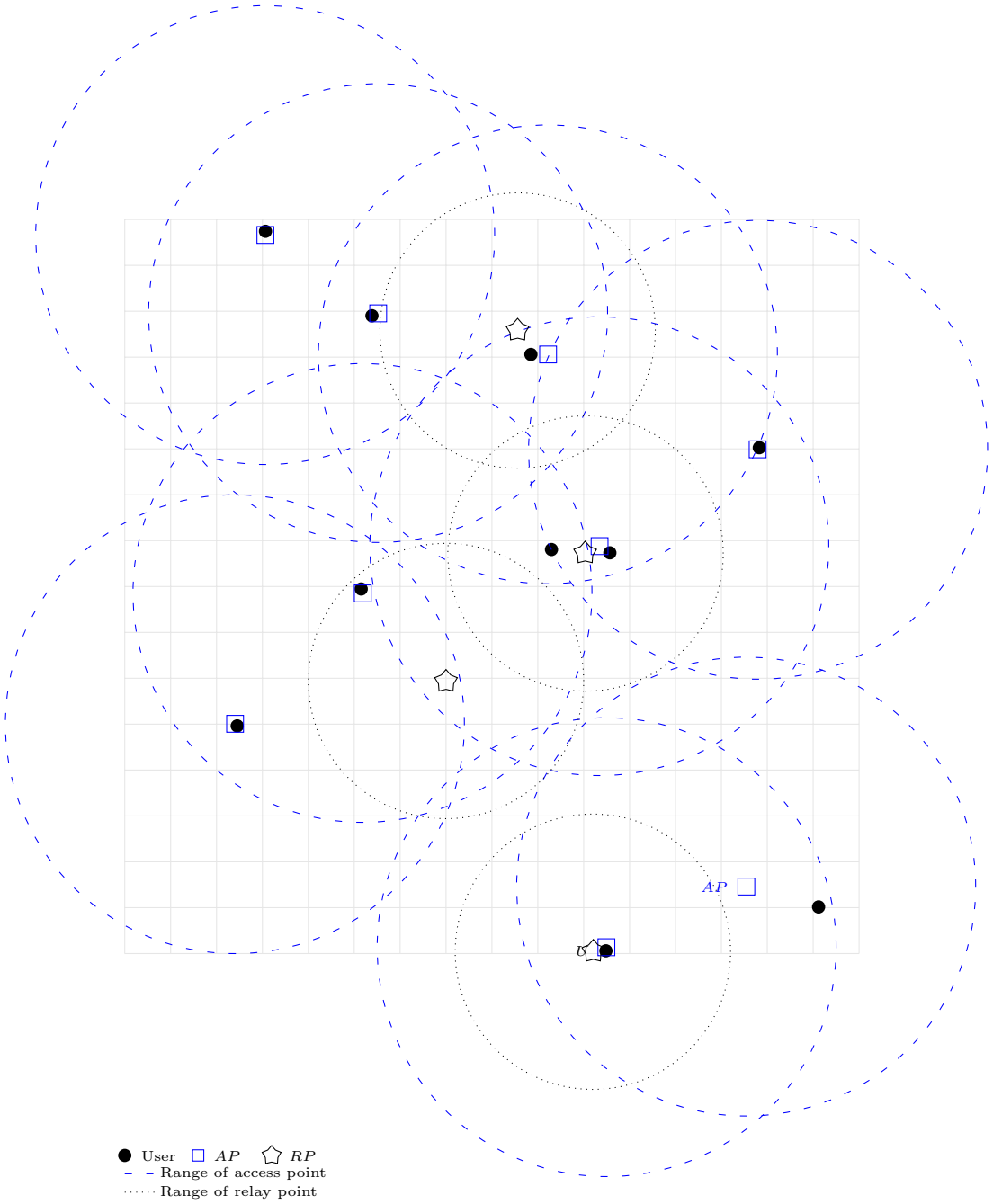
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
600, 0.6753, 0.4523, 0.7830, 0.8306
# of APs = 10, # of RPs = 0

Figure 6.21: Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for Capacitated Resilience with constrained capacities



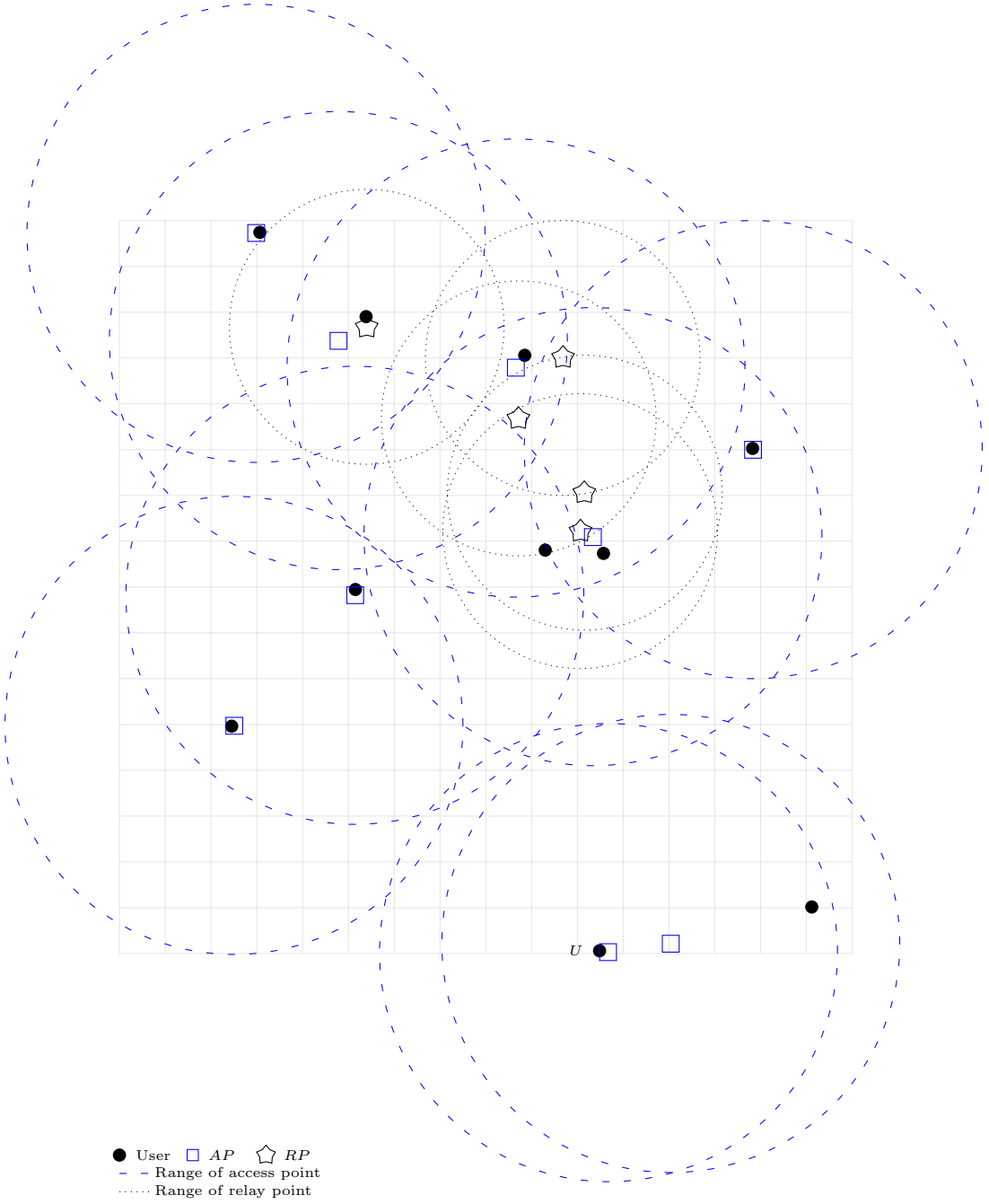
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
570, 0.3655, 0.9101, 0.9036, 0.9412
# of APs = 9, # of RPs = 3

Figure 6.22: Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for TE



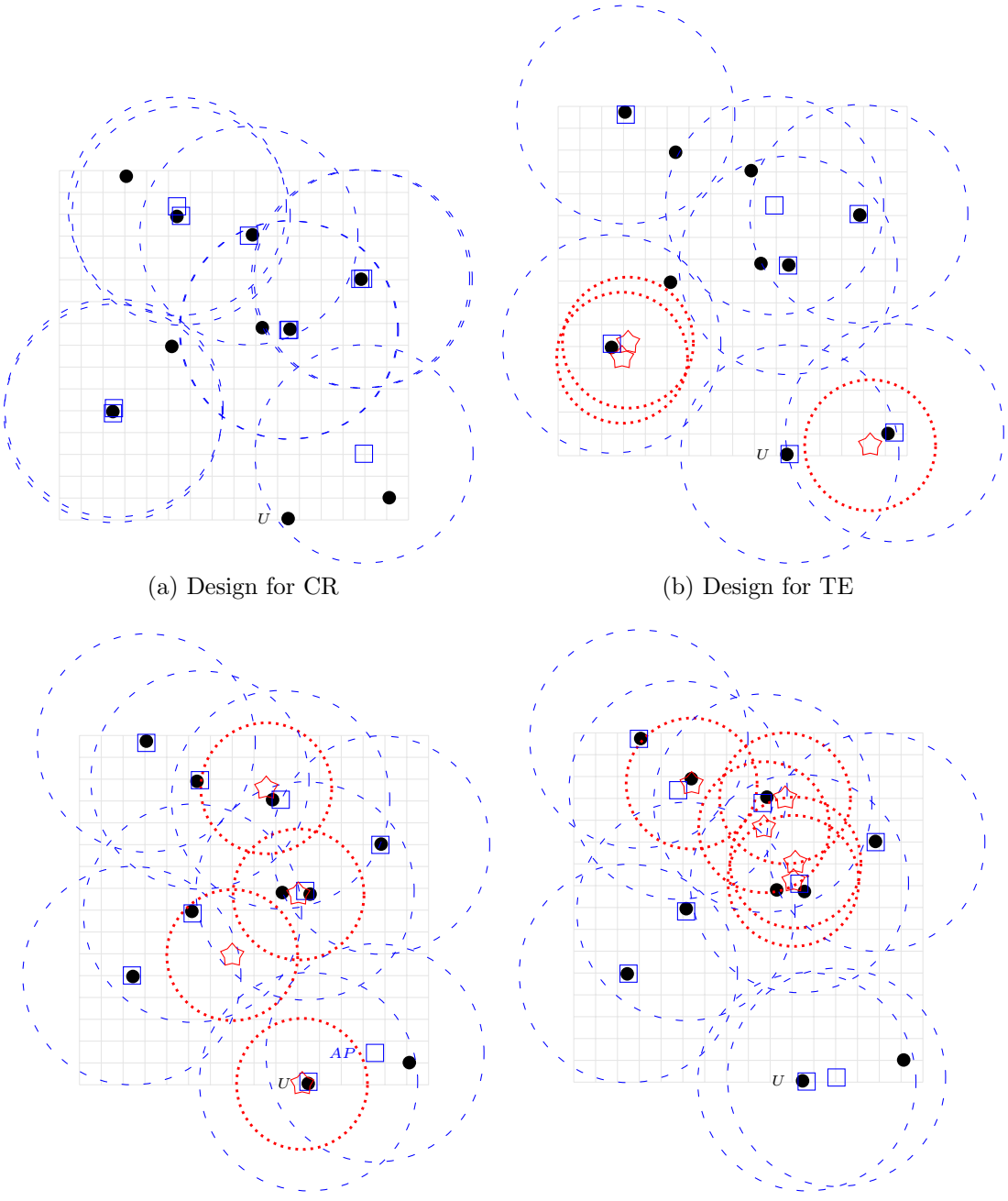
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
580, 0.5248, 0.5361, 0.9787, 0.9844
# of APs = 9, # of RPs = 4

Figure 6.23: Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for two-terminal reliability



Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
590, 0.521, 0.8247, 0.9764, 0.9845
# of APs = 9, # of RPs = 5

Figure 6.24: Network structure of the 10 user problem (problem instance 2, budget=600) found by optimization for all-terminal reliability



(a) Design for CR

(b) Design for TE

(c) Design for two-term. rel.

(d) Design for all-term. rel.

Figure 6.25: Summary of the designs of the problem instance 2 of the 10 user scenario, budget=600

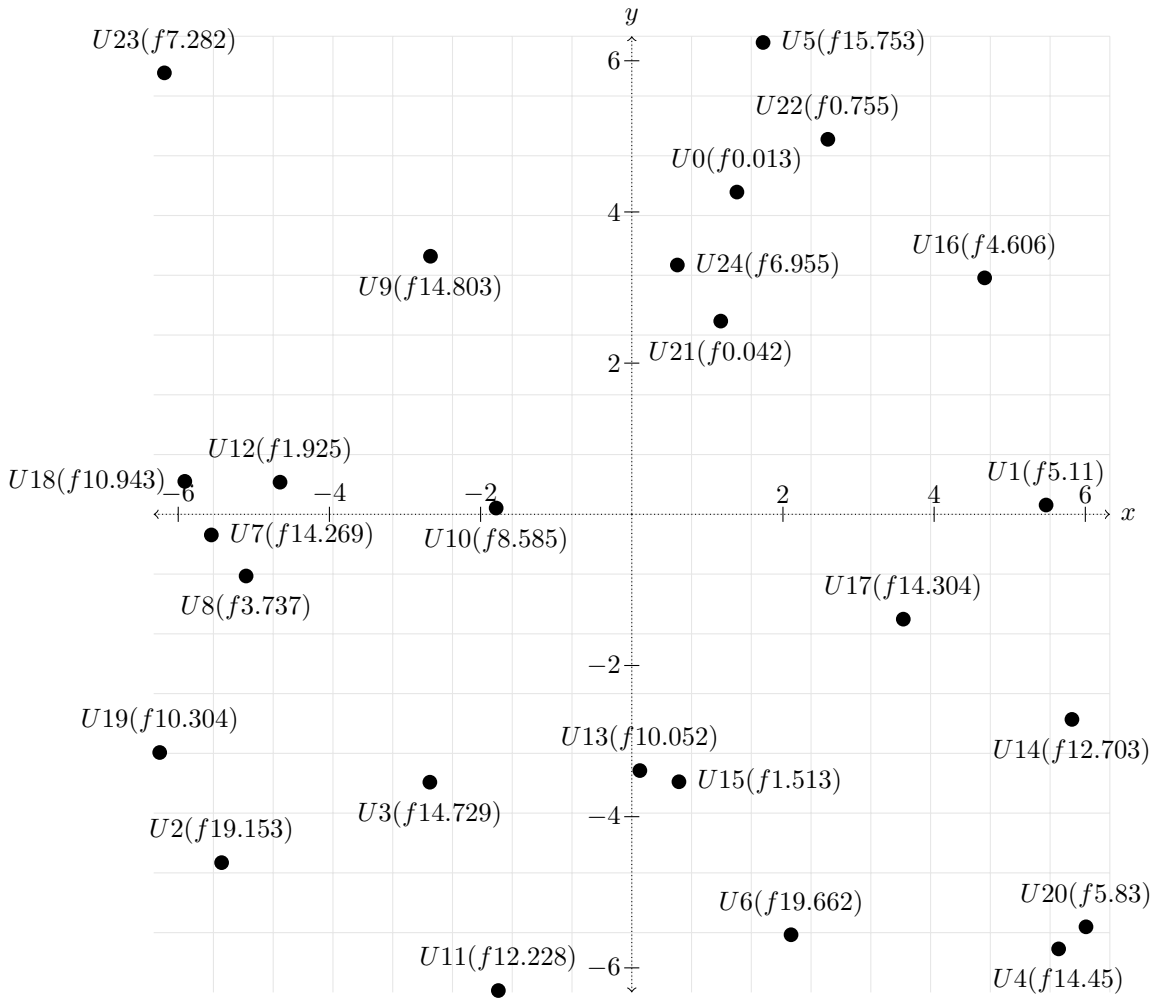


Figure 6.26: Inputs of the 25 user scenario (problem instance 1): User locations (f is the traffic flow requirement)

scenario. Similar to the previous examples, this problem is solved for maximum capacitated resilience, TE, two-terminal reliability and all-terminal reliability and differences in the network designs are discussed.

Solving the 25 user problem for maximum capacitated resilience (Figures 6.27 and 6.28) yields a similar network design found for the previous examples (Sections 6.3.1 and 6.3.2). The high traffic nodes are connected with the devices located close to them and also some redundancy is allocated as the budget allows. Low traffic nodes, for example U15 and U16, are connected with devices which are located further from these users to connect high

traffic nodes or provide redundancy for high traffic nodes. This trend was observed in the previous examples as well. The design with a larger budget (Figure 6.28) has a higher level of redundancy in the network. Similarly, constrained capacity case (Figures 6.29 and 6.30) has similar designs.

Solving the same problem for maximum TE yields a network design with high reliability connections for high traffic nodes. Redundancies are allocated as the budget constraint allows. Unlike with resilience the primary objective is to connect as many high traffic nodes with high reliability devices as possible. Both the 1000 (Figure 6.31) and the 1200 (Figure 6.32) budget scenarios result in similar structures.

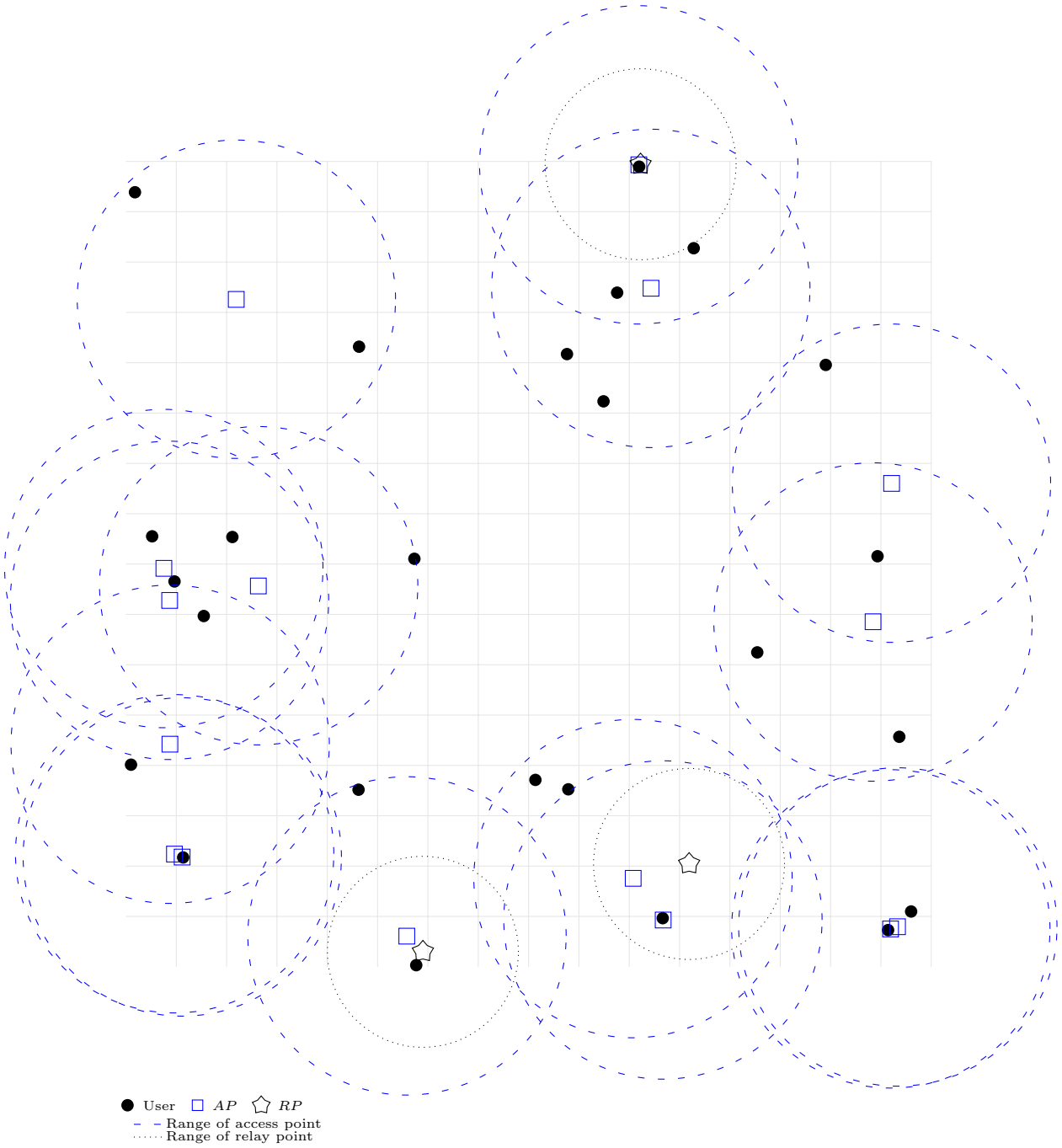
For two-terminal (Figures 6.33 and 6.34) and all-terminal (Figures 6.35 and 6.36) reliability design, the network is different than for capacitated resilience. As discussed in the previous sections, these yield a design which is similar to the one obtained by TE maximization. Network designs obtained by maximization of two-terminal and all-terminal reliabilities emphasize high reliability connections for high traffic nodes and add redundancies as the budget permits. Redundancy allocation for two-terminal and all-terminal reliability is different than for capacitated resilience because the redundancies are located near the nodes that do not have a high reliability connection. This increases the chance for those nodes to keep connected to the network.

Table 6.4 summarizes the values of all metrics for all network designs presented in this section. Figures 6.37 and 6.38 compare the designs that are presented in this section.

6.3.4 Summary of the differences of the network designs

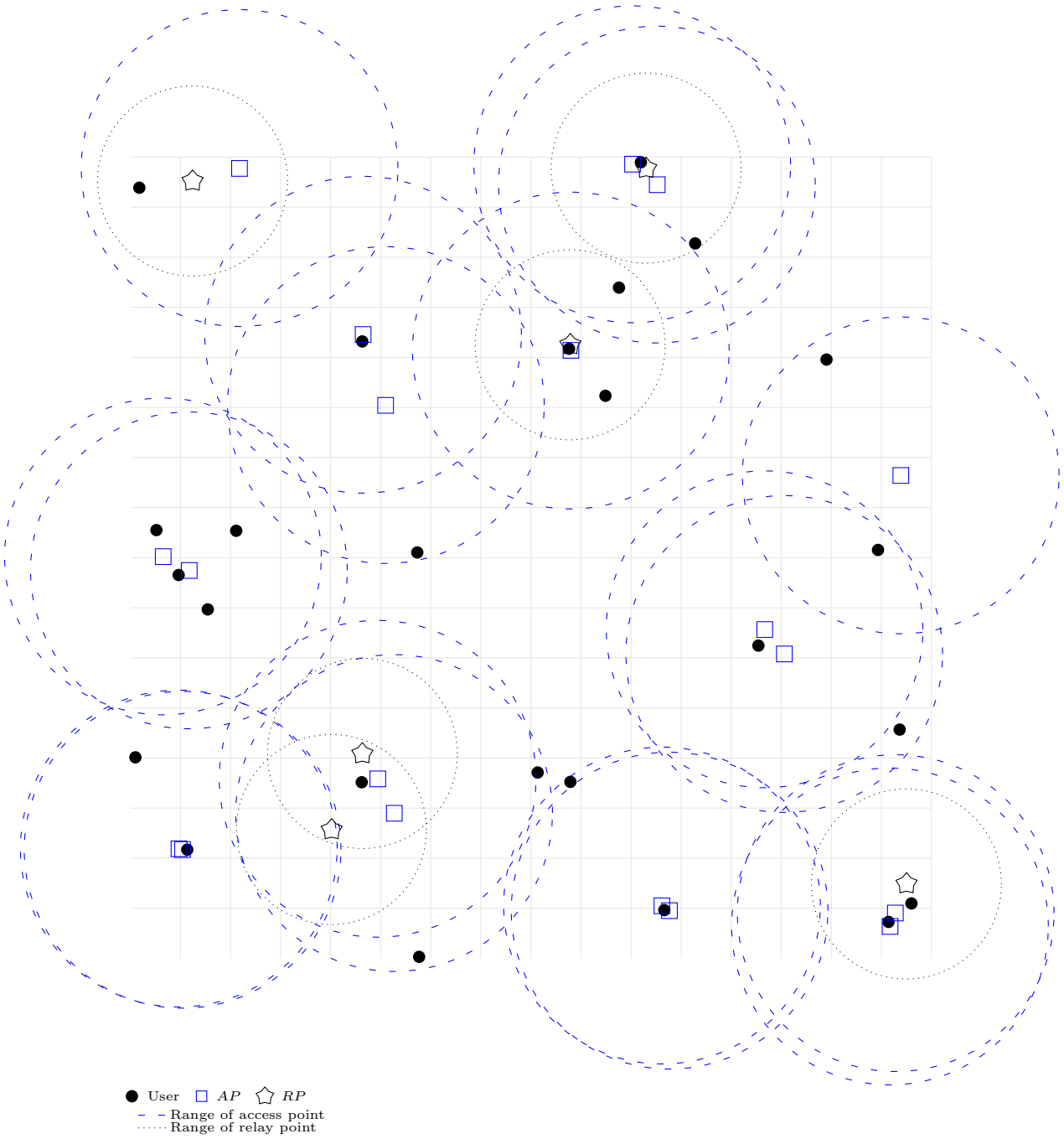
According to the examples given in the previous sections, there are some differences in the network designs obtained by optimization for different reliability/survivability metrics.

The most distinct network design is obtained by capacitated resilience maximization. Users are connected to the network with a nearby device to ensure a high reliability connection in capacitated resilience optimization. As seen in the provided examples of the previous



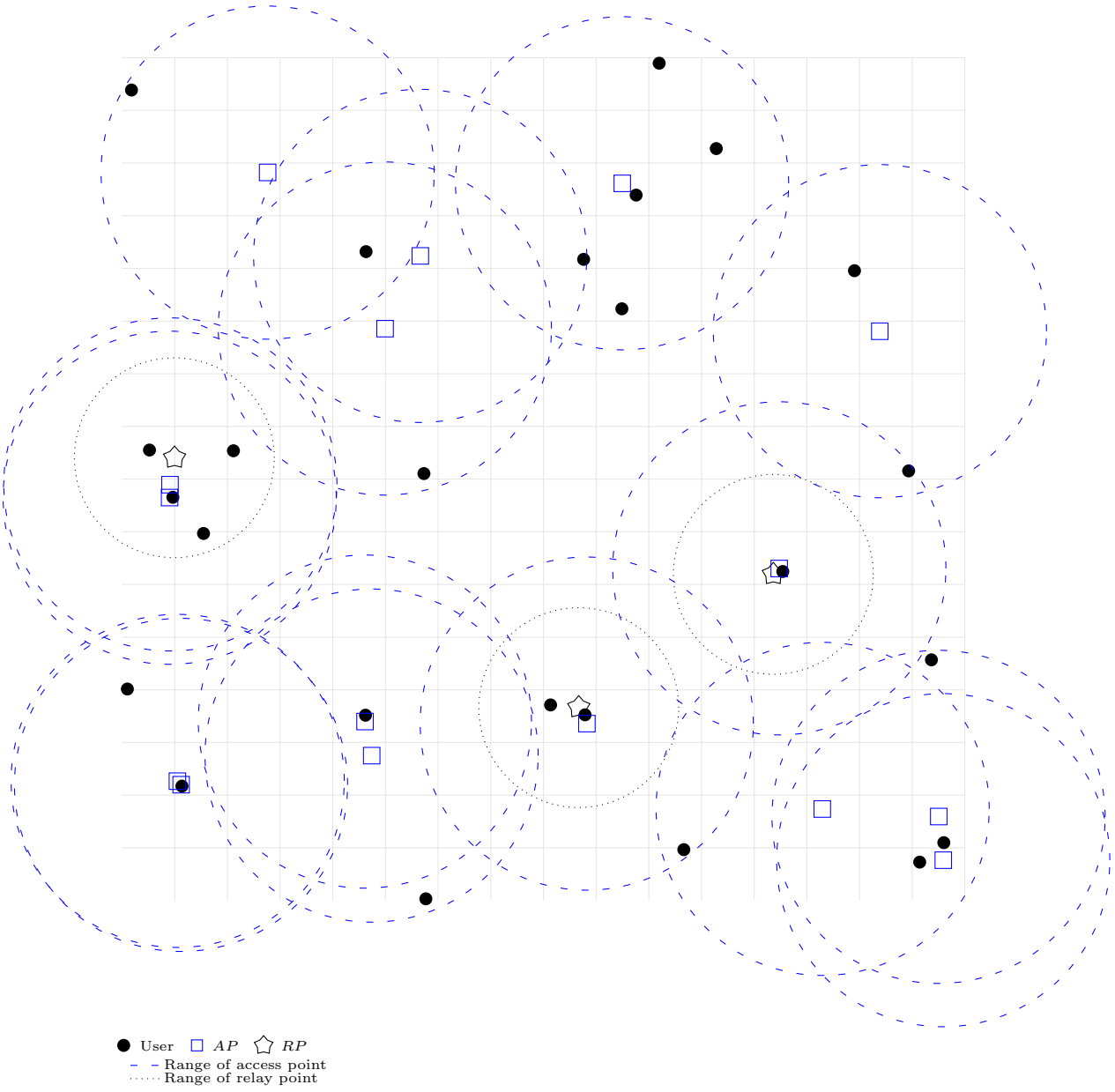
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
990, 0.4667, 0.3904, 0.5942, 0.6305
# of APs = 16, # of RPs = 3

Figure 6.27: Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for Capacitated Resilience with unconstrained capacities



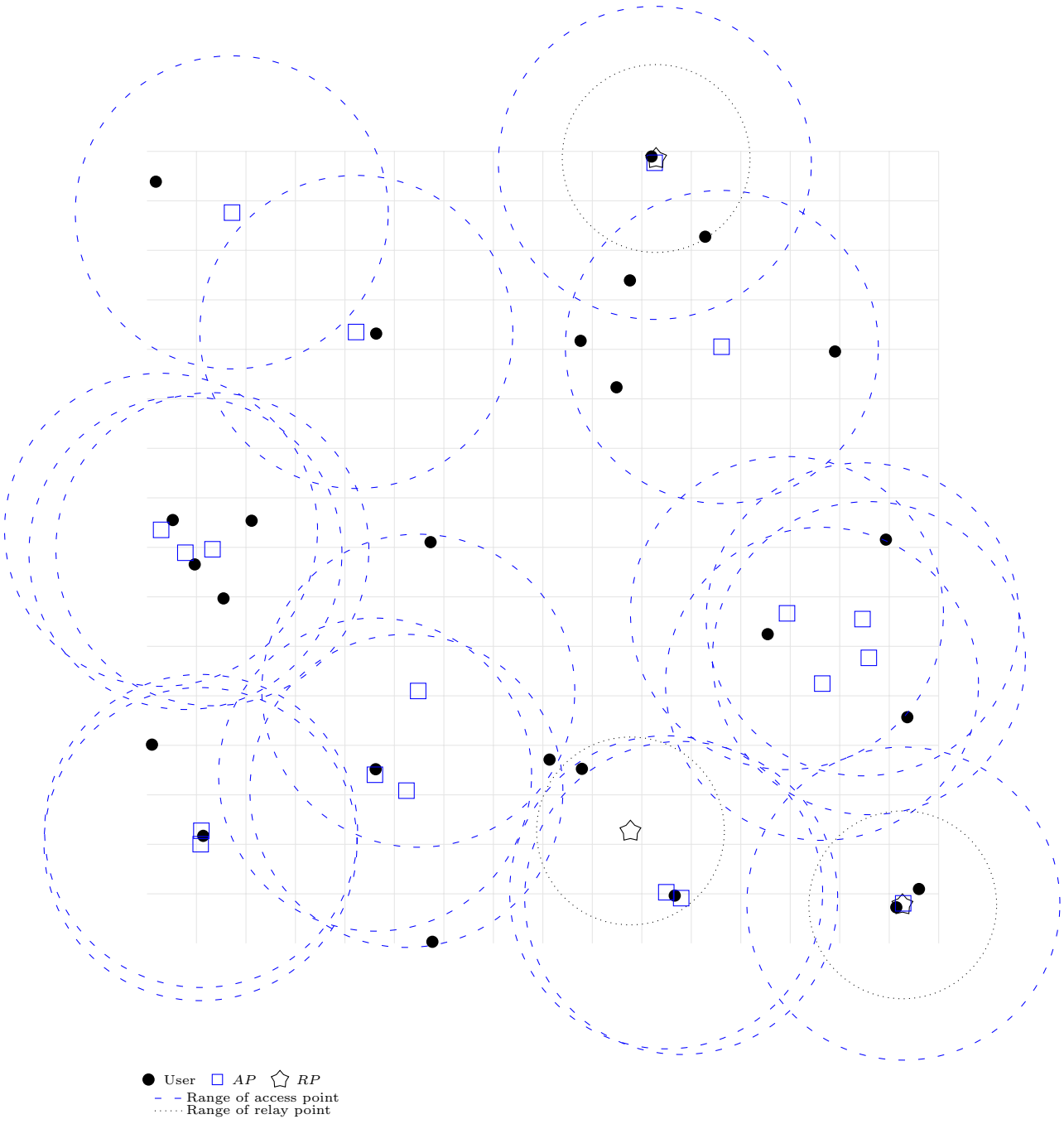
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
1200, 0.5559, 0.4717, 0.6992, 0.7434
# of APs = 19, # of RPs = 6

Figure 6.28: Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for Capacitated Resilience with unconstrained capacities



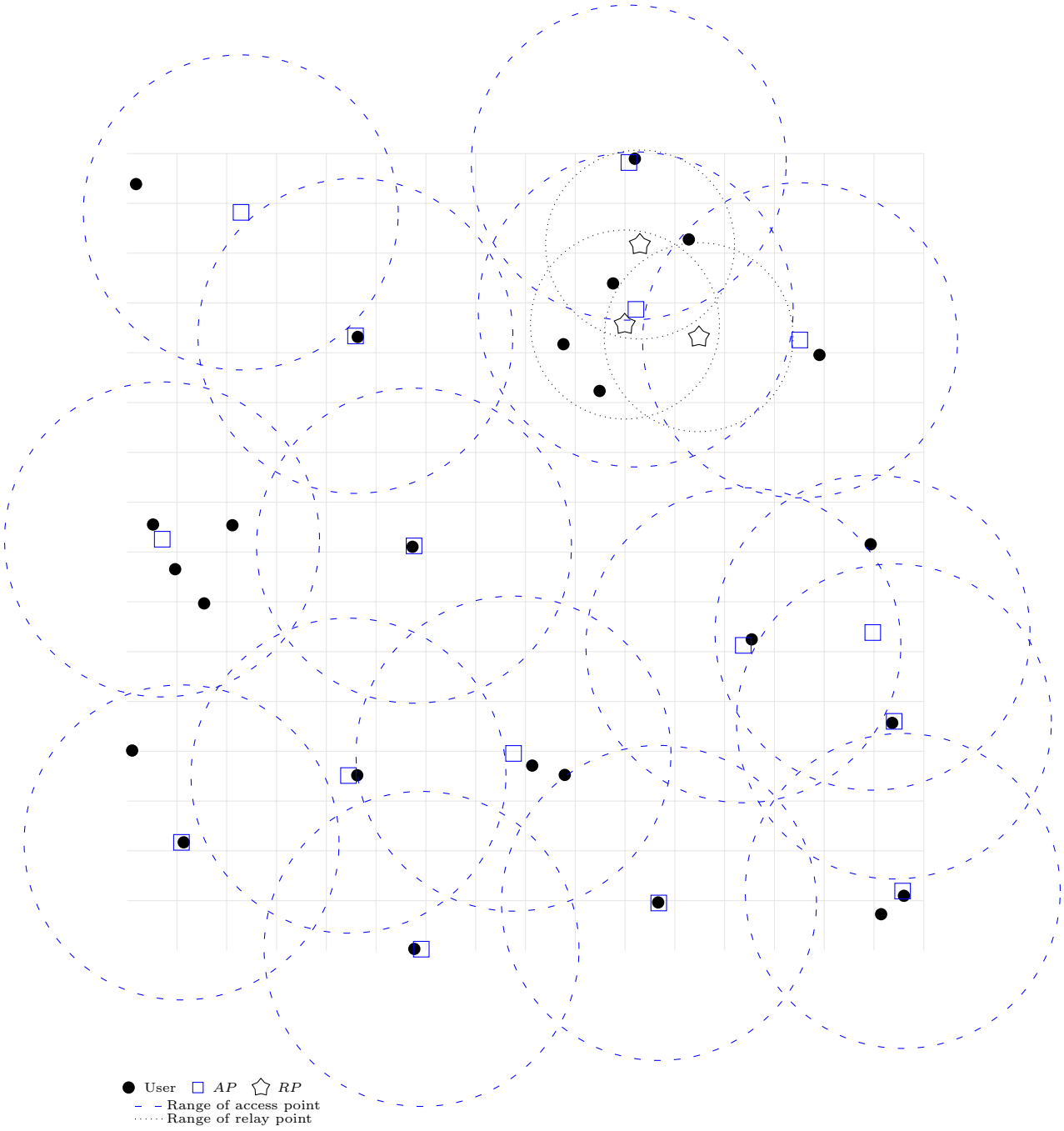
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
990, 0.3958, 0.4795, 0.5751, 0.6199
# of APs = 16, # of RPs = 3

Figure 6.29: Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for Capacitated Resilience with constrained capacities



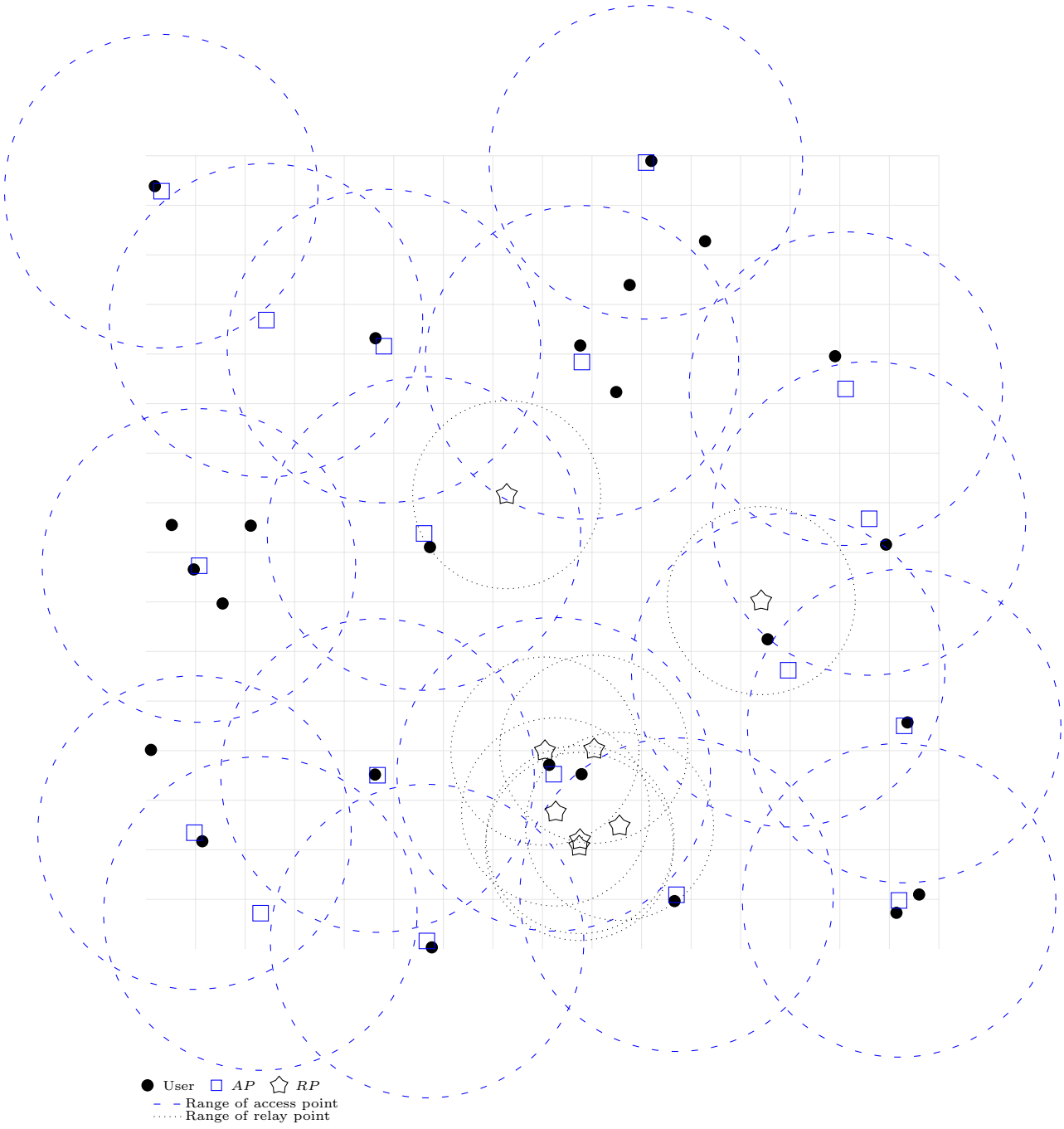
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
1170, 0.5165, 0.4823, 0.6909, 0.7549
# of APs = 19, # of RPs = 3

Figure 6.30: Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for Capacitated Resilience with constrained capacities



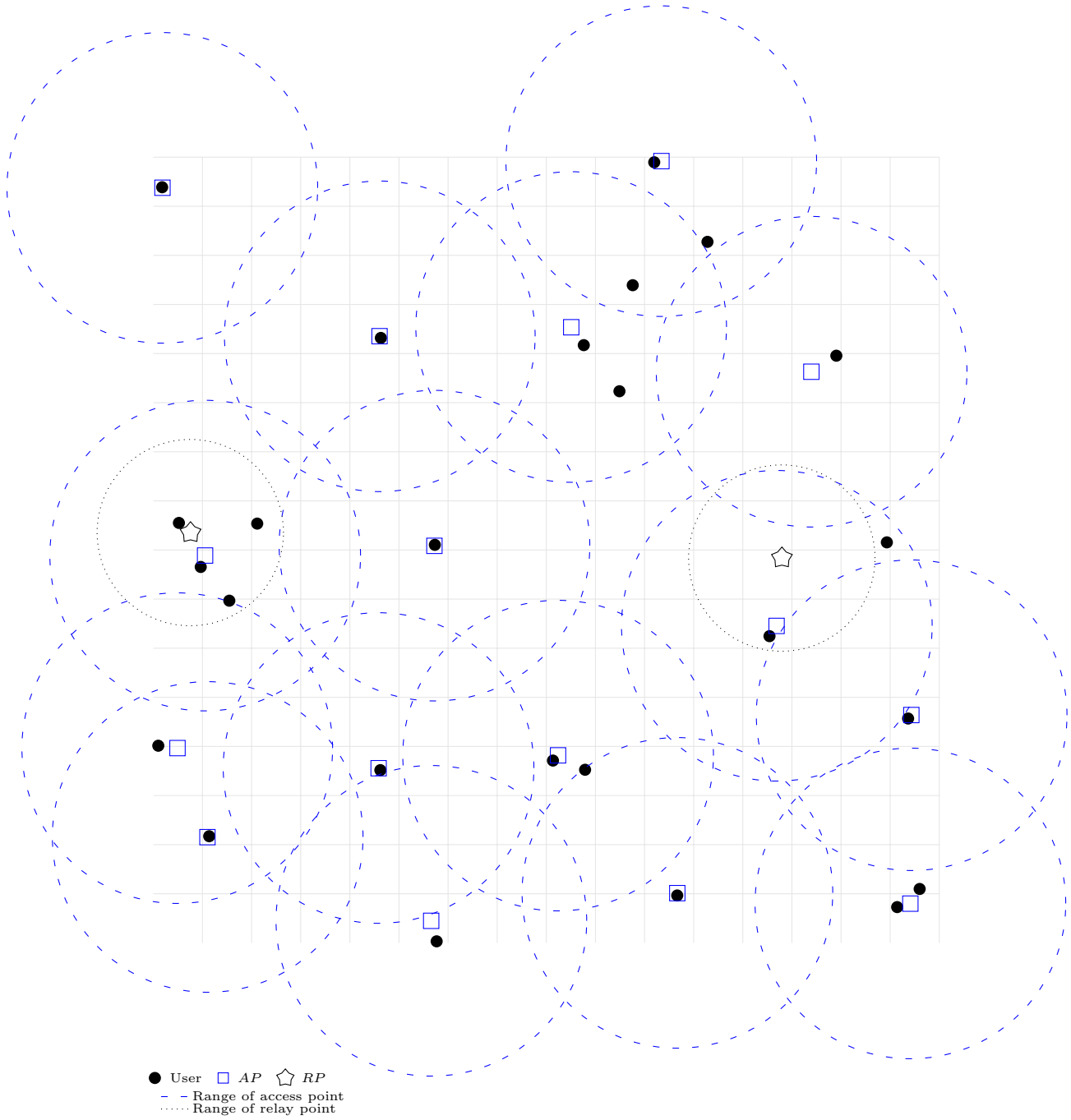
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
990, 0.0501, 0.8422, 0.8515, 0.8538
# of APs = 16, # of RPs = 3

Figure 6.31: Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for TE



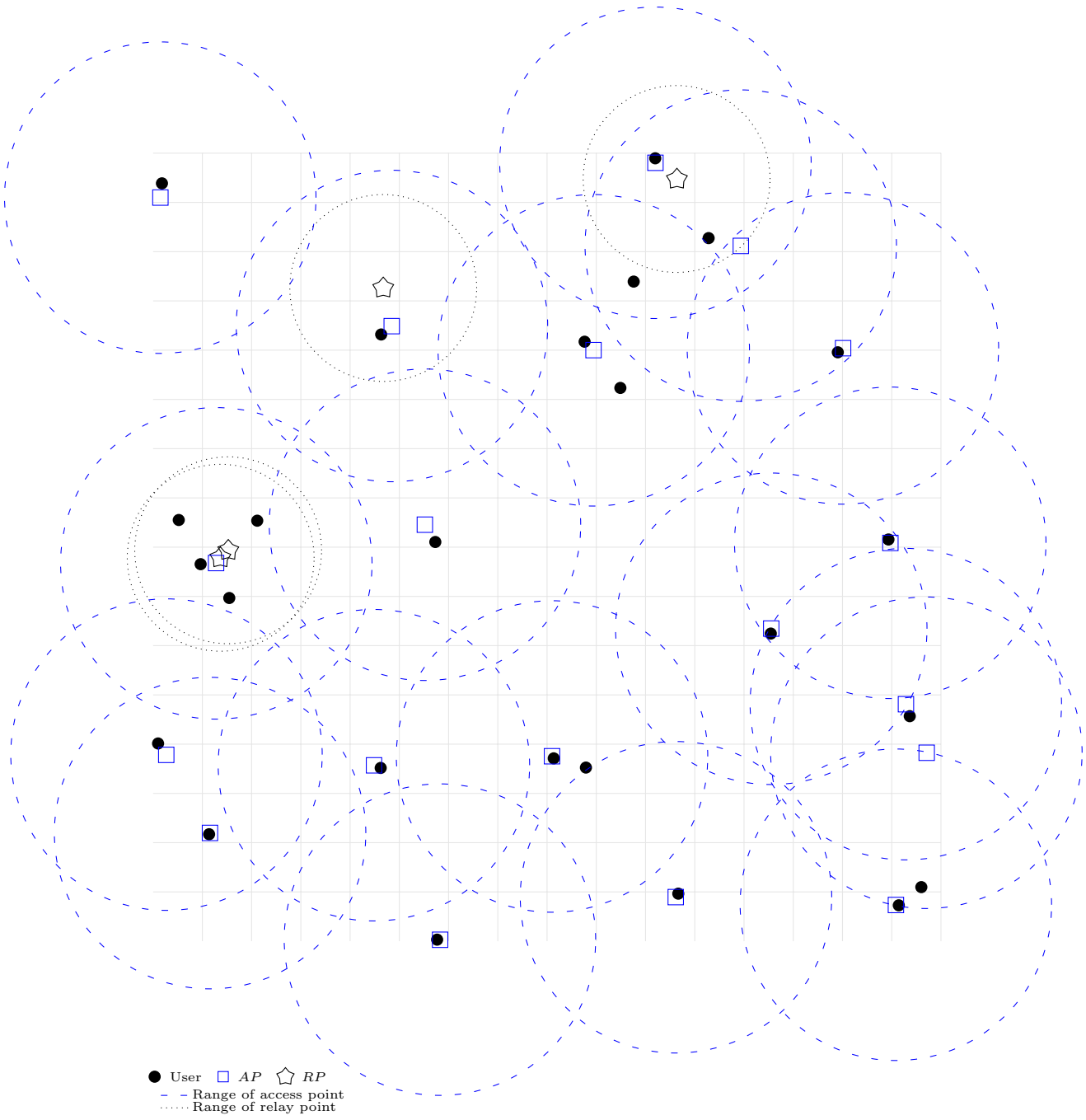
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
1160, 0.0712, 0.9084, 0.8805, 0.8848
# of APs = 18, # of RPs = 8

Figure 6.32: Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for TE



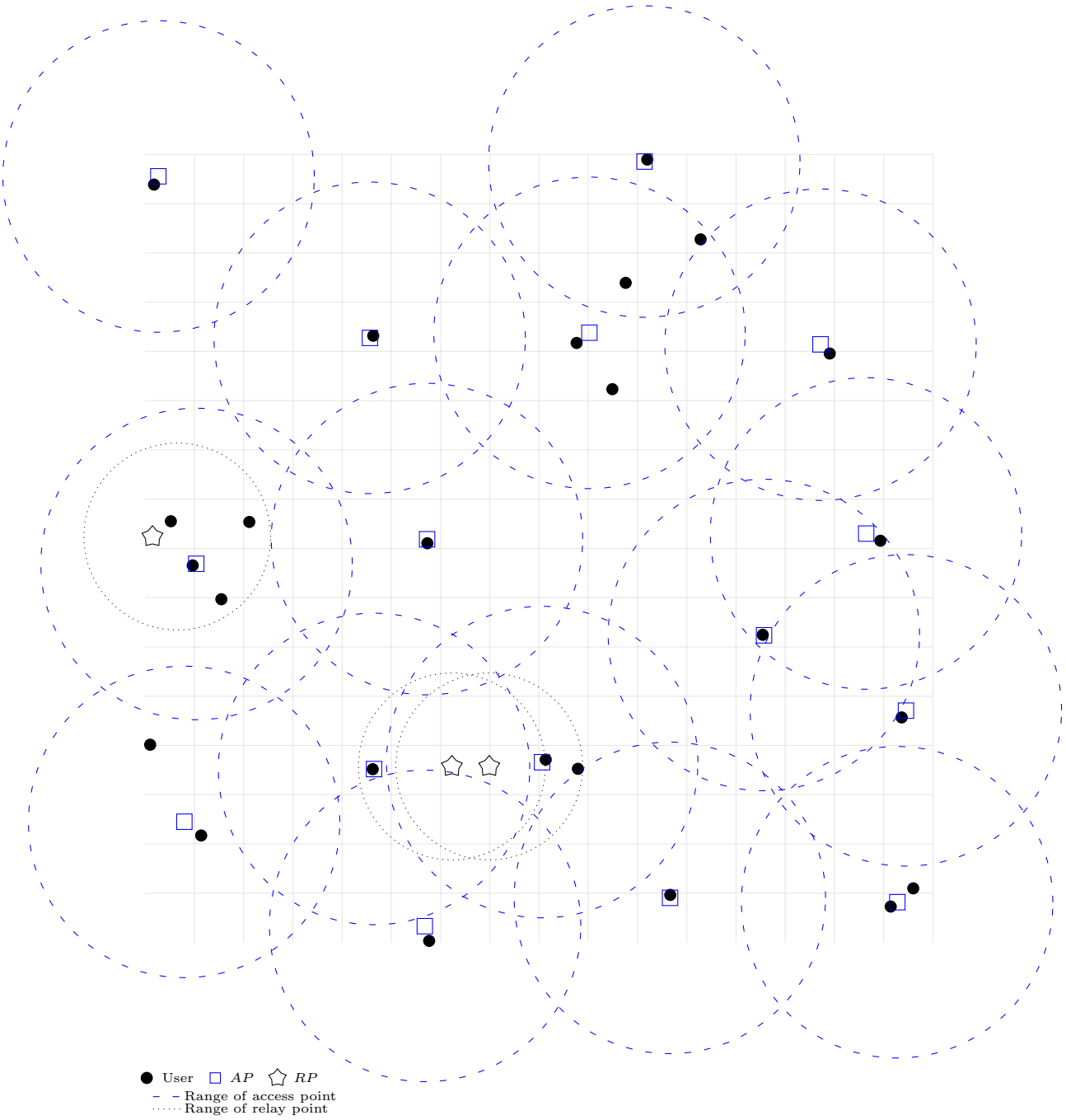
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
980, 0.0959, 0.6682, 0.9157, 0.9180
# of APs = 16, # of RPs = 2

Figure 6.33: Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for two-terminal reliability



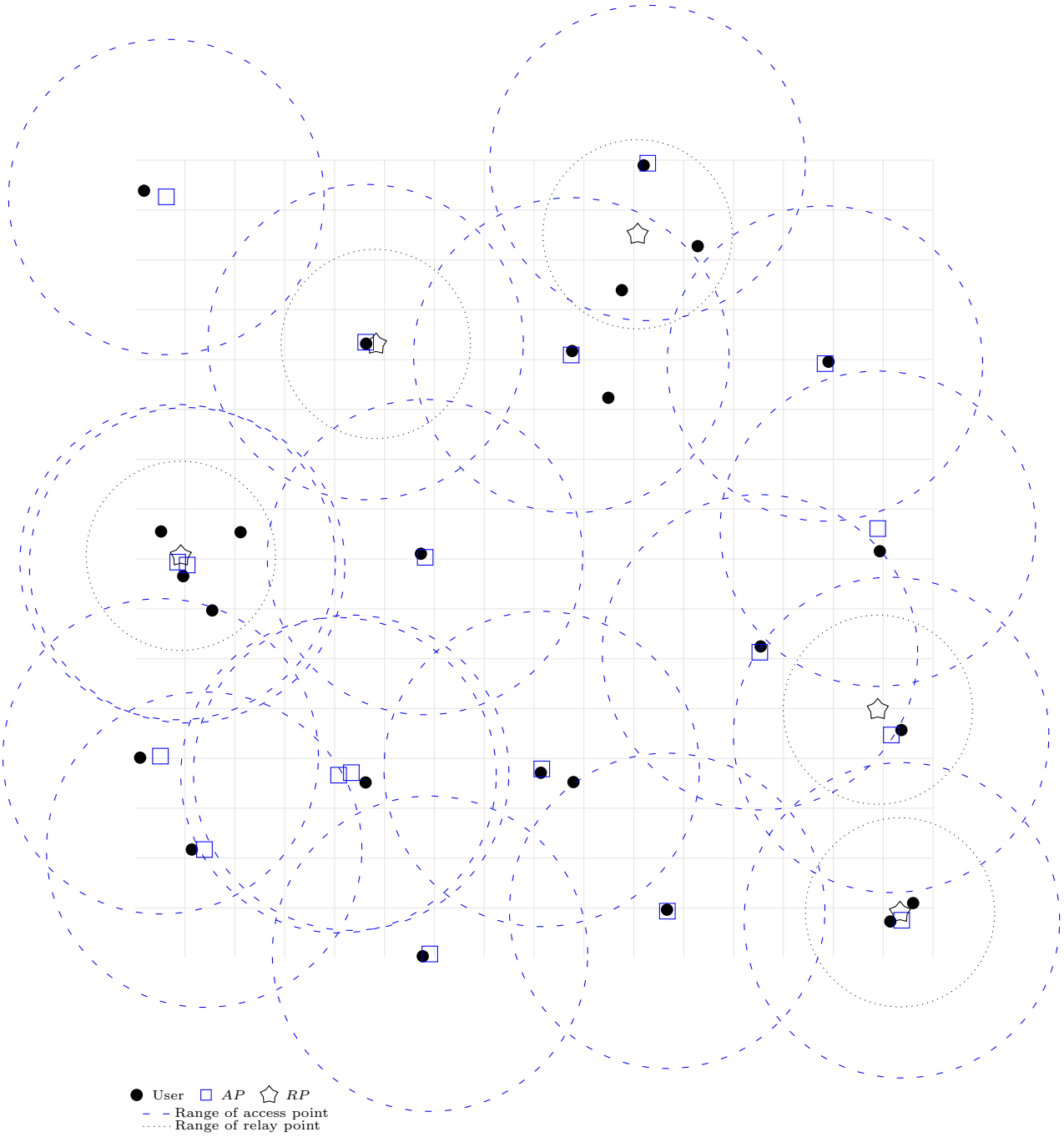
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
1180, 0.2349, 0.6378, 0.9488, 0.9554
# of APs = 19, # of RPs = 4

Figure 6.34: Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for two-terminal reliability



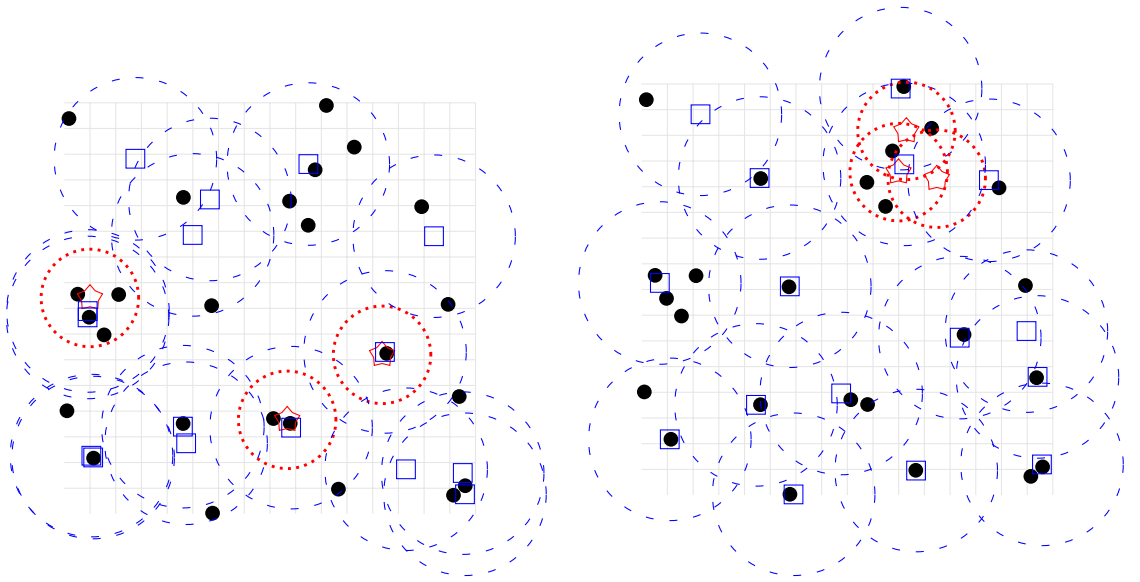
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
990, 0.1072, 0.7007, 0.9143, 0.9192
# of APs = 16, # of RPs = 3

Figure 6.35: Network structure of the 25 user problem (problem instance 1, budget=1000) found by optimization for all-terminal reliability



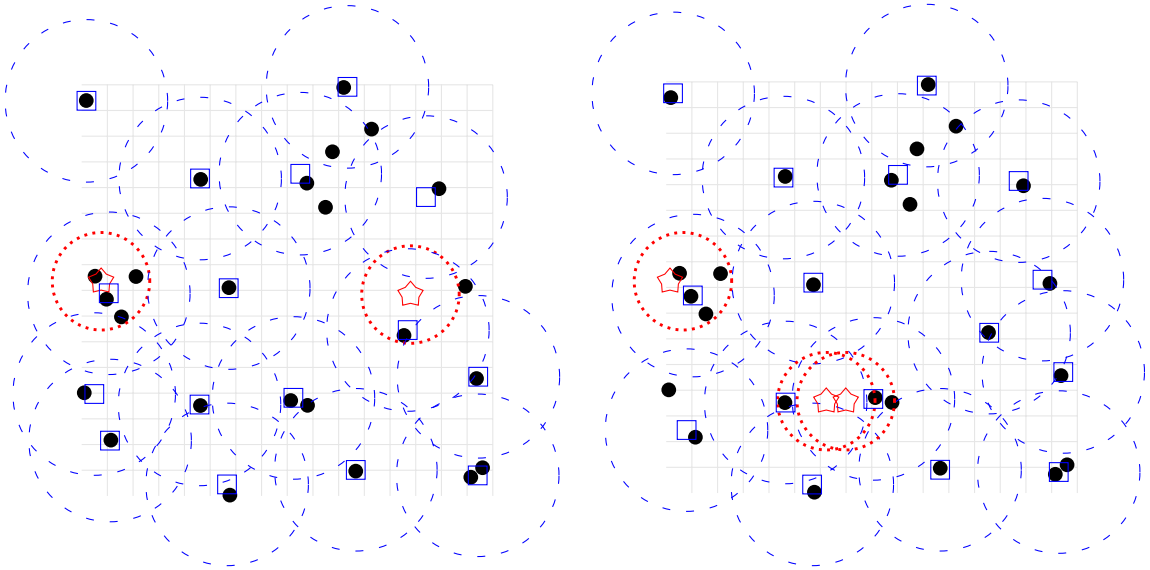
Cost, Capacitated Resilience, TE, Two-terminal Reliability, All-terminal Reliability:
1190, 0.3301, 0.7674, 0.9425, 0.9619
# of APs = 19, # of RPs = 5

Figure 6.36: Network structure of the 25 user problem (problem instance 1, budget=1200) found by optimization for all-terminal reliability



(a) Design for CR

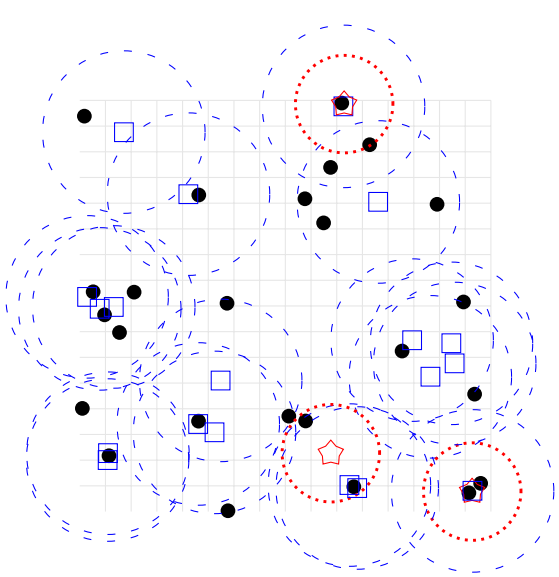
(b) Design for TE



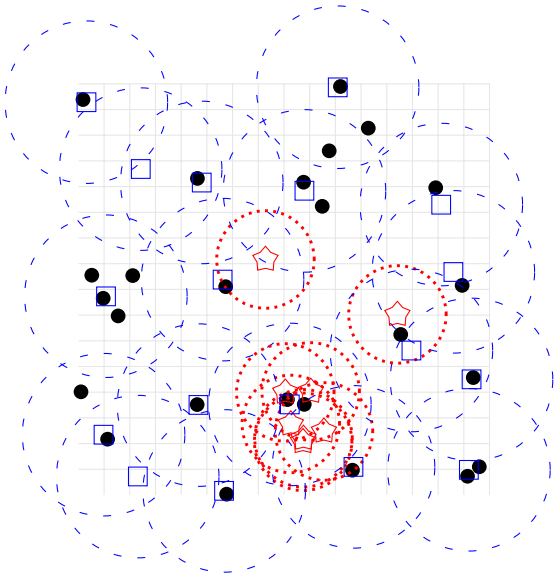
(c) Design for two-term. rel.

(d) Design for all-term. rel.

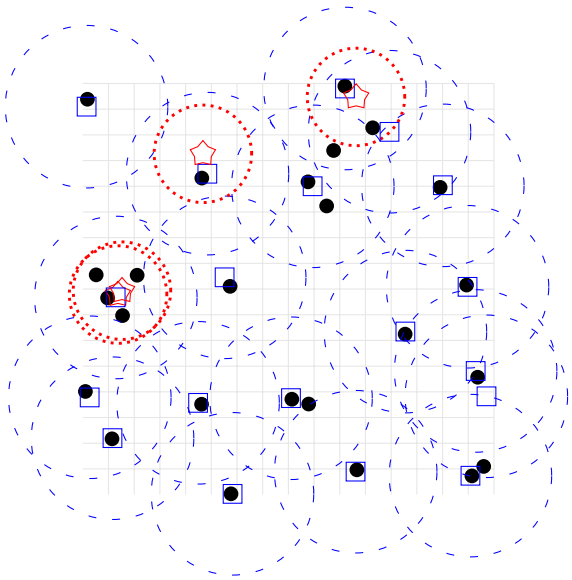
Figure 6.37: Summary of the designs of the problem instance 1 of the 25 user scenario, budget=1000



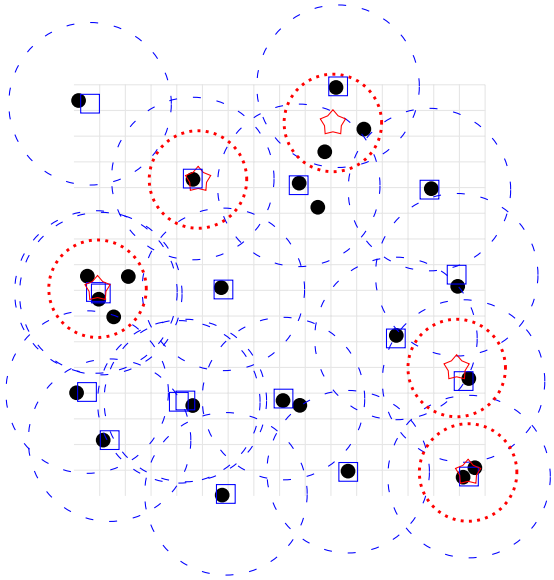
(a) Design for CR



(b) Design for TE



(c) Design for two-term. rel.



(d) Design for all-term. rel.

Figure 6.38: Summary of the designs of the problem instance 1 of the 25 user scenario, budget=1200

Table 6.4: Summary of all metrics for the 25 user scenario, problem instance 1

Budget	Optimized by	Results			
		CR	TE	2-term	All-term
1000	CR (unconstrained capacities)	0.4667	0.3904	0.5942	0.6305
	CR	0.3958	0.4795	0.5751	0.6199
	TE	0.0501	0.8422	0.8515	0.8538
	Two-terminal rel. (2-term)	0.0959	0.6682	0.9157	0.9180
	All-terminal rel. (all-term)	0.1072	0.7007	0.9143	0.9192
1200	CR (unconstrained capacities)	0.5559	0.4717	0.6992	0.7434
	CR	0.5165	0.4823	0.6909	0.7549
	TE	0.0712	0.9084	0.8805	0.8848
	Two-terminal rel. (2-term)	0.2349	0.6378	0.9488	0.9554
	All-terminal rel. (all-term)	0.3301	0.7674	0.9425	0.9619

sections, this is also a very common design rule for other metrics. The main difference of capacitated resilience is with the redundancy allocation. Redundancy is achieved by locating an additional device nearby a user. Capacitated resilience optimization locates the additional device(s) near a user to increase its "resilience". This approach, providing alternative paths, is at the core of the capacitated resilience calculation (Equation 3.2 in Section 3.2). However, budget constraint limits the number of additional devices. Therefore, redundancies are allocated near high traffic nodes or crowded regions to maximize capacitated resilience as calculated by the weighted average of user level capacitated resiliences in terms of user traffic requirements (Equation 3.1 in Section 3.2). Because of this prioritization some low traffic nodes are connected to the network without redundancies. This is a reasonable strategy because satisfying a larger number of users (or larger traffic requirements) maximizes the overall satisfaction of the network. Also, low traffic users are still connected to the network and some level of redundancy is provided to them within the given budget constraint. The ideal location for a low traffic user would be near a high traffic node so that it can benefit from redundancy of the high traffic node. If the low traffic node is isolated from other users, that user is assigned with a distant device with limited or no redundancy.

TE, two-terminal and all-terminal reliabilities have some common design properties. For example, they try to assign users a high reliability device to connect the network. Although

it seems to be similar to the design of capacitated resilience, this is actually their main difference from the capacitated resilience. Capacitated resilience adds redundancies near high traffic nodes, whereas the other metrics try to connect more nodes to a high reliability device then redundancies are allocated if budget allows. In most cases, the redundancies in the network generated by TE, two-terminal and all-terminal reliabilities are very limited. There is a slight difference between the designs of two-terminal and all-terminal reliability. The design for two-terminal reliability has more emphasis on strong connections, however, the all-terminal design uses more redundancy in network design. For example, in Figure 6.16 the redundancies are allocated in a decentralized way to serve many users instead of a limited number of users. On the other hand, two-terminal reliability allocates redundancies nearby users as observed in Figure 6.14.

The network design of capacitated resilience can provide better connectivity if there are catastrophic failures or attacks in the network because of its higher level of redundancy in the network. Any potential attack in the network most likely aims for high traffic areas. These high traffic areas can be more crowded or consist of users with high traffic requirements. This dissertation solves network design problem for wireless networks. For other networks, disconnection may mean transportation breakdowns, telephone breakdowns or interdiction in military supply service. From the rich literature of the network interdiction area, an earlier study shows that an attacker's goal might be removing n edges to reduce the maximum amount of flow (Wollmer, 1964). This is based on max-flow min-cut network problems. Since the calculation of capacitated resilience actually takes the "min-cut"s into account, it provides a more resilient design than the others. The main goal is to keep users connected to the network. Obviously, a planned network interdiction causes more harm than random failures or attacks, therefore capacitated resilience provides even more resilient network for random interdiction (or failures). As further study, network designs obtained by these metrics will be compared for planned interdictions on the network.

6.4 Bi-objective (TE and capacitated resilience) ES

In previous sections, the bi-objective ES was solved for cost and capacitated resilience. In this section, the bi-objective ES is solved for traffic efficiency and capacitated resilience. As seen from Section 6.3, different network designs are obtained by optimization for TE and for capacitated resilience.

As Section 6.2 summarizes, the correlation between TE and capacitated resilience is weak when optimizing for TE. Therefore, high TE values may not result in high capacitated resiliences. Similarly, high capacitated resilience values may not guarantee high TE values. Figures 6.39 and 6.40 present the Pareto fronts of the bi-objective ES optimized for traffic efficiency and capacitated resilience for the 10 user scenario with budget constraints of 500 and 600, respectively. The search space of the bi-objective ES of capacitated resilience and TE is larger than the one of capacitated resilience and cost due to the continuous values of TE. Therefore, instead of 3000 generations with population size of 30 and children size of 30, ES is run for 5000 generations with population size of 40 and children size of 40. Five random number seeds are used for each scenario. Traffic efficiency is calculated by the simulation method presented in Section 4.3.2 and capacitated resilience is calculated by setting the values of number of alternative paths and cut-set size to 10 and 4, respectively.

The first thing to notice is the number of solutions in the Pareto fronts. Compared to the Pareto front solutions of cost and capacitated resilience (Section 5.1.3), there are more solutions for traffic efficiency and capacitated resilience optimization. The reason is that the search space is larger as both metrics (TE and capacitated resilience) are continuous, whereas cost is discrete.

As expected, a budget increase (from 500 to 600) increases the values of TE and capacitated resilience of most solutions. Specifically, with a budget increase, the highest capacitated resilience increased from 0.5889 (random number seed 2) to 0.6537 (random number seed 4). Similarly, the highest TE increased dramatically (0.8670 for random number seed 4 of

budget=500 case and 0.9665 for random number seed 4 of budget=600). The lowest capacitated resilience remained almost same (0.0345 for random number seed 4 of budget=500 case and 0.0483 for random number seed 2 of budget=600). The lowest TE decreased from 0.5089 (random number seed 5 of budget=500 case) to 0.4238 (random number seed 1 of budget=600 case), however, this lowest value seems to be an extreme case. Without considering the random number seed 1 of budget=600 case, the lowest TE increases to 0.5561.

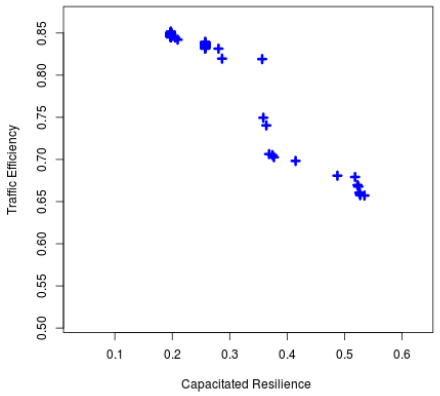
The two ends of the Pareto front (low TE/high capacitated resilience and high TE/low capacitated resilience) suggest that the structure of the networks are different in these areas. Specifically, the network structures of the solutions in the high TE/low capacitated resilience section have limited redundancies but more reliable connections for most nodes (Figure 6.41). The solutions in the other extreme section (low TE/high capacitated resilience) have more redundancies but some of the low or moderate traffic nodes have less reliable connections (Figure 6.42). However, the solutions in the middle sections of the Pareto fronts have more balanced network designs with reliable connections and some level of redundancy (Figure 6.43). Figure 6.44 provides a comparison of these three different designs.

6.5 Solution time comparison of capacitated resilience and other metrics

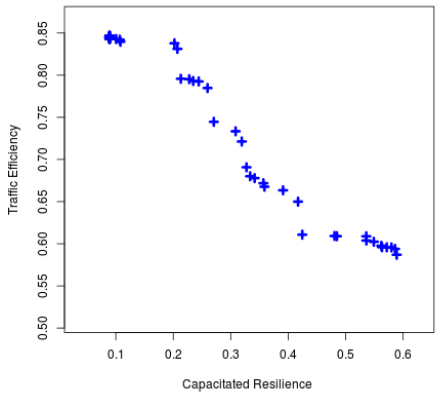
In this section, computational experience is presented. The solution methods were coded in Java and the problems were solved on a Linux computer with 2.66 Ghz Intel Quad Core Xeon W3520 CPU and 8 GB memory.

6.5.1 Solution time comparison of capacitated resilience: Effect of number of paths and cut-set size

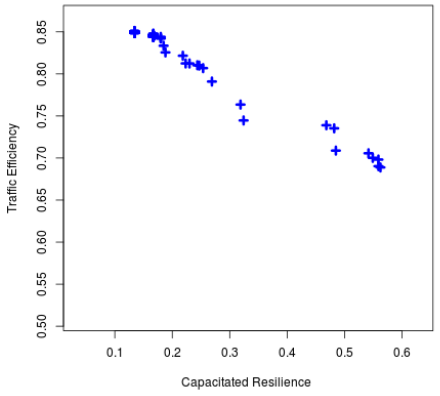
This section summarizes the solution times of the 10 user scenario (10 problem instances with 10 random number seed for each problem instance) with budget of 500 and 600 for single (capacitated resilience) and bi-objective (cost and capacitated resilience) optimization. The effects of number of alternative paths and cut set sizes are investigated.



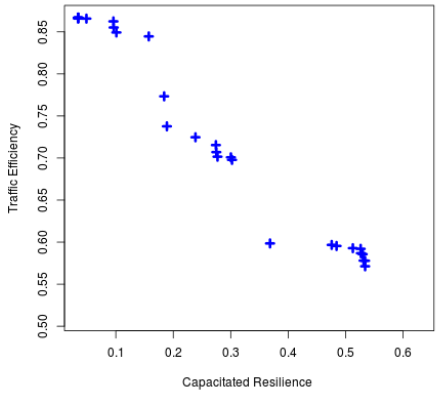
(a) Random Number Seed 1



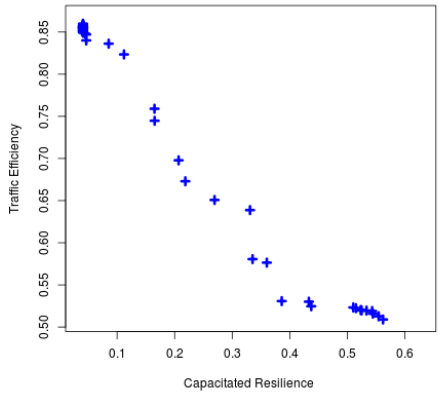
(b) Random Number Seed 2



(c) Random Number Seed 3

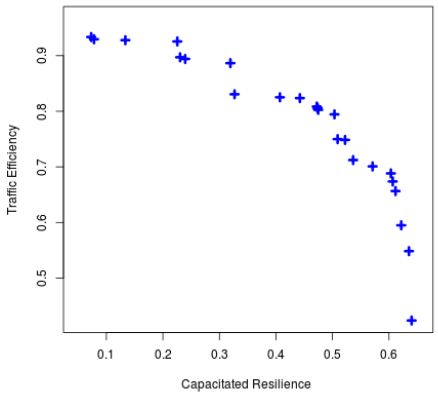


(d) Random Number Seed 4

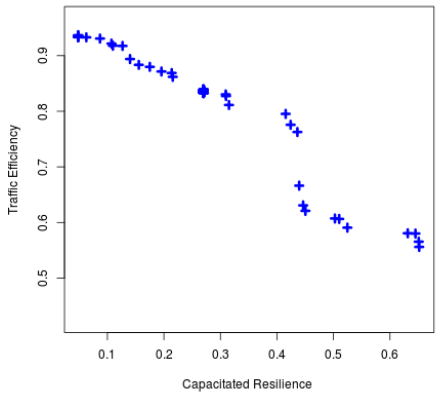


(e) Random Number Seed 5

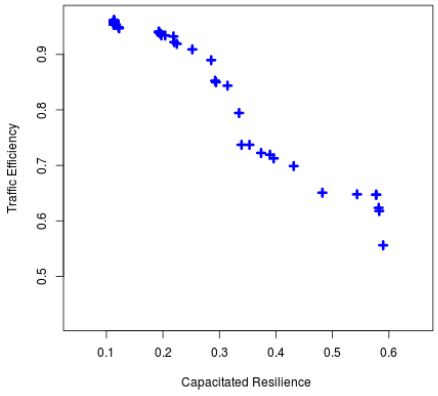
Figure 6.39: Pareto fronts for the 10 user scenario problem instance 1 with budget constraint of 500: Bi-objective optimization of TE and capacitated resilience



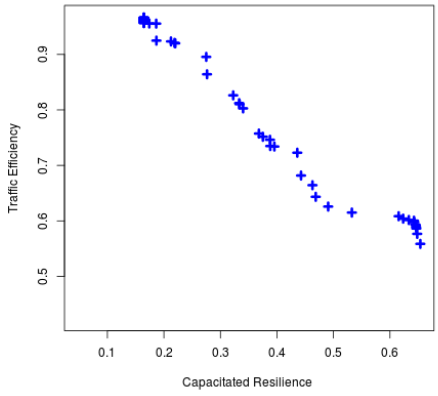
(a) Random Number Seed 1



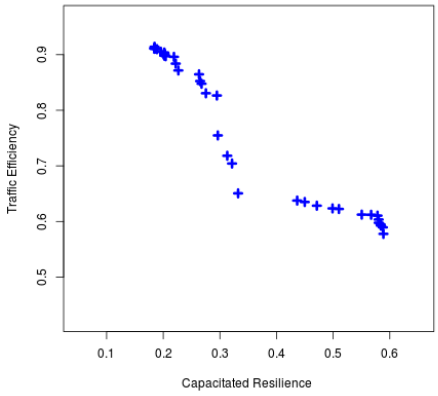
(b) Random Number Seed 2



(c) Random Number Seed 3

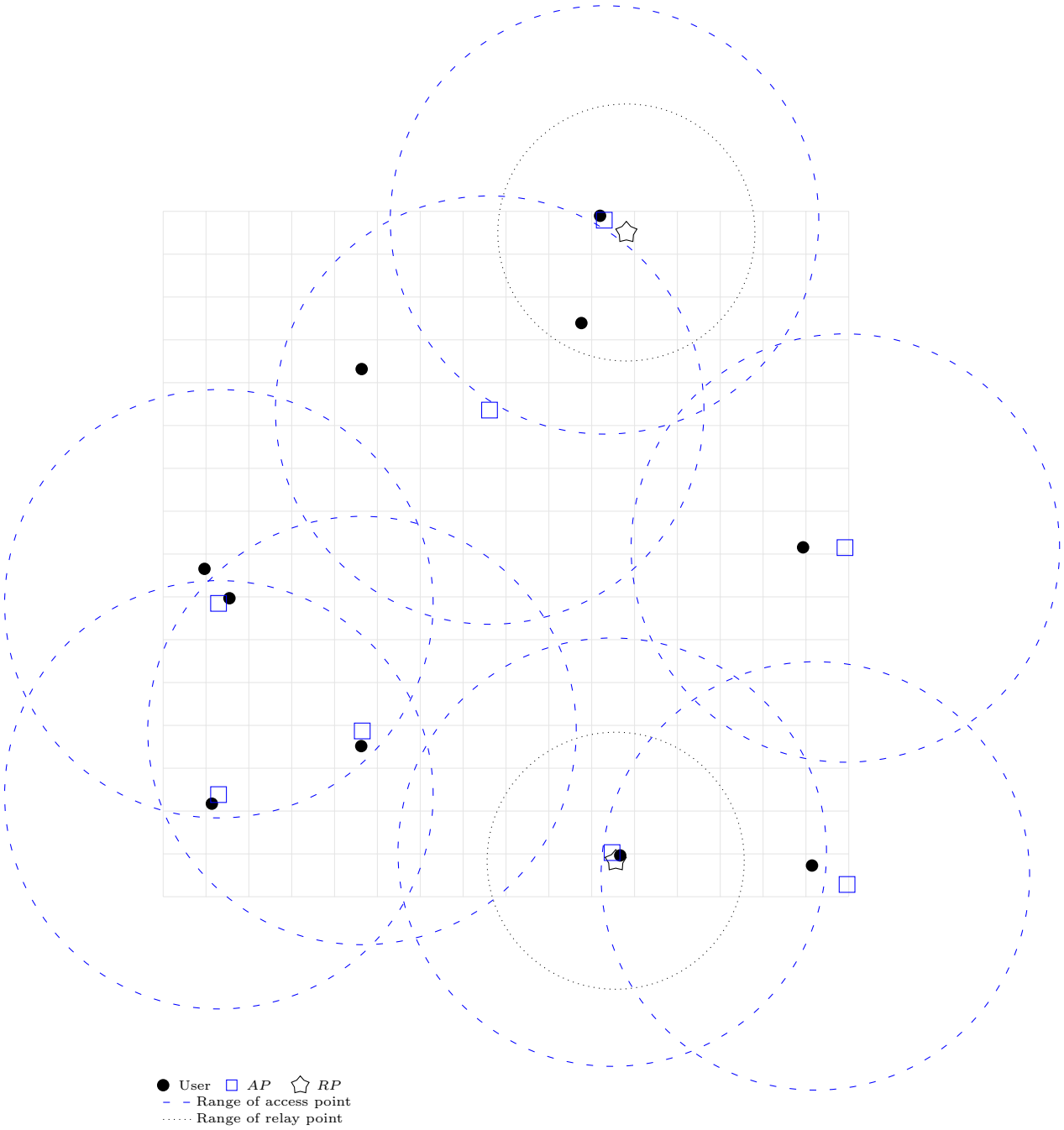


(d) Random Number Seed 4



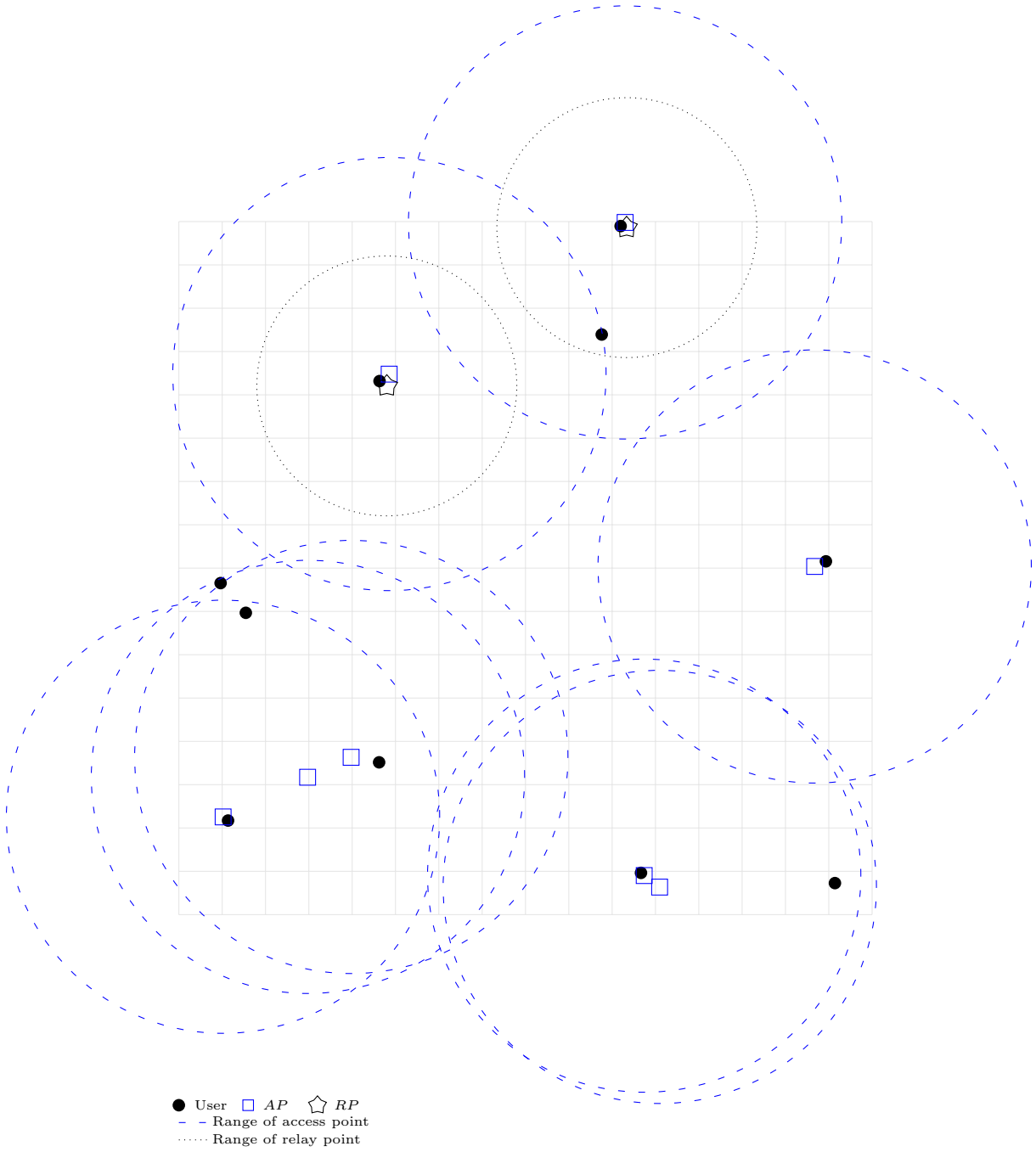
(e) Random Number Seed 5

Figure 6.40: Pareto fronts for the 10 user scenario problem instance 1 with budget constraint of 600: Bi-objective optimization of TE and capacitated resilience



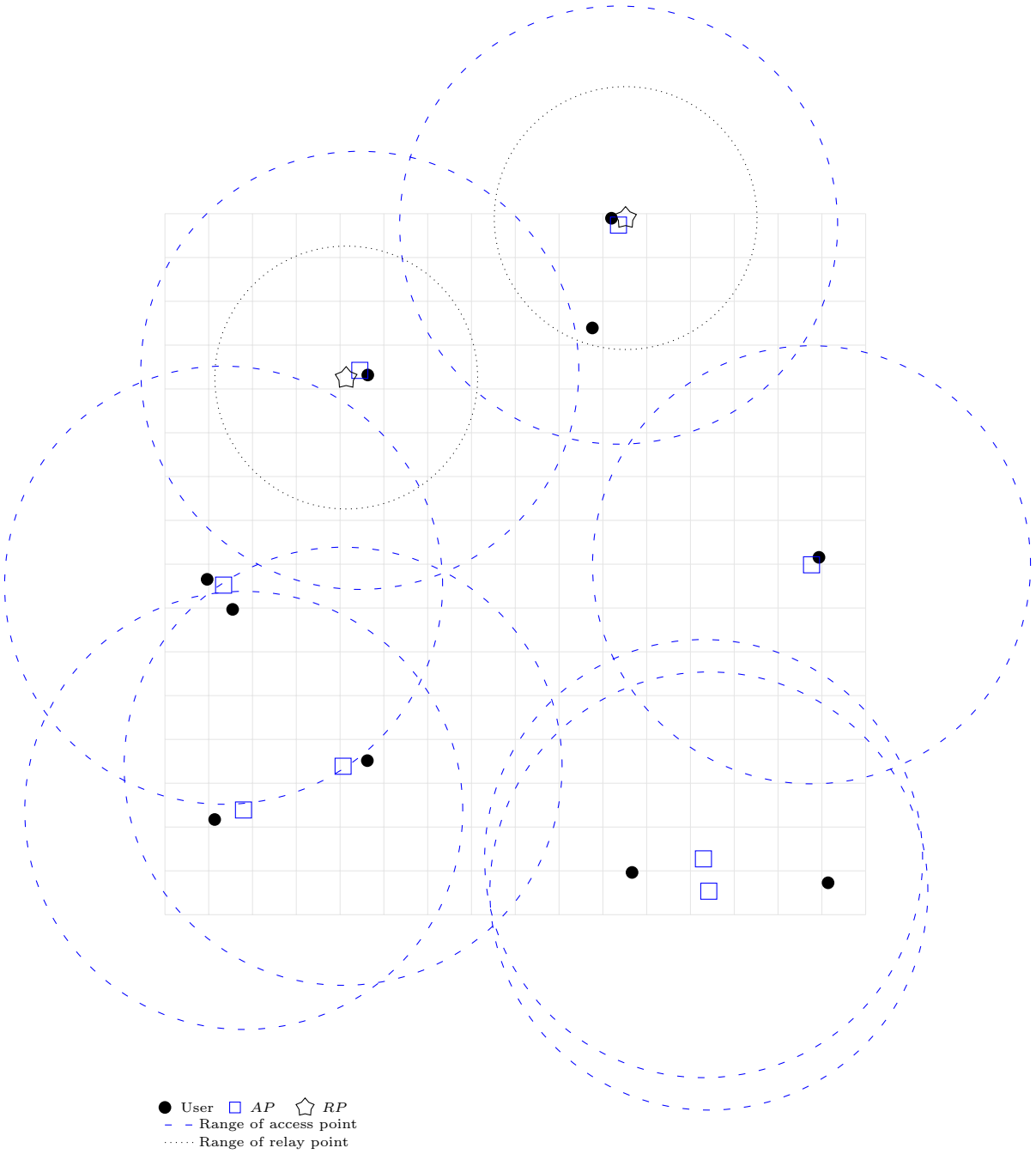
Cost, Capacitated Resilience, TE:
500.0, 0.2278, 0.7953
# of APs = 8, # of RPs = 2

Figure 6.41: Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (high TE and low capacitated resilience case)



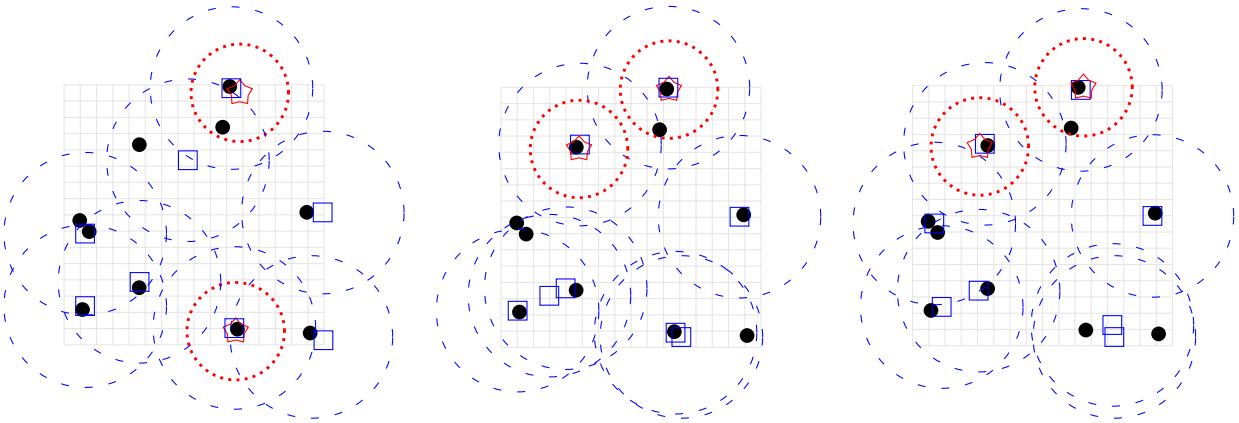
Cost, Capacitated Resilience, TE:
500.0, 0.5633, 0.5959
# of APs = 8, # of RPs = 2

Figure 6.42: Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (low TE and high capacitated resilience case)



Cost, Capacitated Resilience, TE:
500.0, 0.3769, 0.6818
# of APs = 8, # of RPs = 2

Figure 6.43: Network structure of the 10 user problem (problem instance 1, budget=500) found by bi-objective optimization for capacitated resilience and TE (medium TE and medium capacitated resilience case)



(a) high TE and low CR (b) low TE and high CR (c) medium TE and medium CR

Figure 6.44: Comparison of the designs presented in Figures 6.41 through 6.43

Figures 6.45 and 6.46 summarize the solution time per iteration (one ES generation). The reason of using time per iteration instead of total time is that the early termination criterion (250 and 500 non-improving generations for single and bi-objective ES) affects the total solution time. Therefore, solution time per iteration provides a more accurate comparison. As seen from these figures, the bi-objective ES takes significantly more time than the single objective because it runs for 2000 generations with early termination criterion of 500 non-improved generations whereas the single objective runs for 1000 generations with early termination criterion of 250 non-improved generations. The solution times of the bi-objective are about the double of the time required for single objective ES due to the extra operations to maintain Pareto front. As also expected, the solution times of the problems with a budget constraint of 600 are higher than the ones with a budget constraint of 500 due to larger number of available devices.

More importantly, the time per iteration increases as the number of alternative paths increases from 1 to ten (Figures 6.45 and 6.46). Cut set size does not seem to affect the solution time per iteration significantly for the 10 user scenario because the cut-set size does not have a significant effect on the capacitated resilience calculation due to the small number of users (and devices). To support this, Figures 6.47 and 6.48 show that capacitated resilience does not change significantly for different cut-set sizes. Interestingly, the difference

in capacitated resilience between the number of paths of two and 10 is insignificant and it validates that the estimation of capacitated resilience using a smaller number of alternative paths and cut-set size performs well, i.e., finds comparable capacitated resilience values (near-exact values) faster.

At each ES generation, some number of alternative paths are evaluated. This number varies for each generation and finding alternative paths requires solving a k-shortest path problem, therefore it affects the solution time directly. Figures 6.49 and 6.50 compares the number of evaluated alternative paths. Clearly, these figures have the same trend with the solution time (per generation) figures (Figures 6.45 and 6.46). The number of evaluated paths increases as the parameter value of number of alternative paths increases from one to 10. As expected, cut-set size does not have any effect on the number of evaluated alternative paths.

The effect of the problem size on the total solution time is investigated in detail in Section 6.5.2.

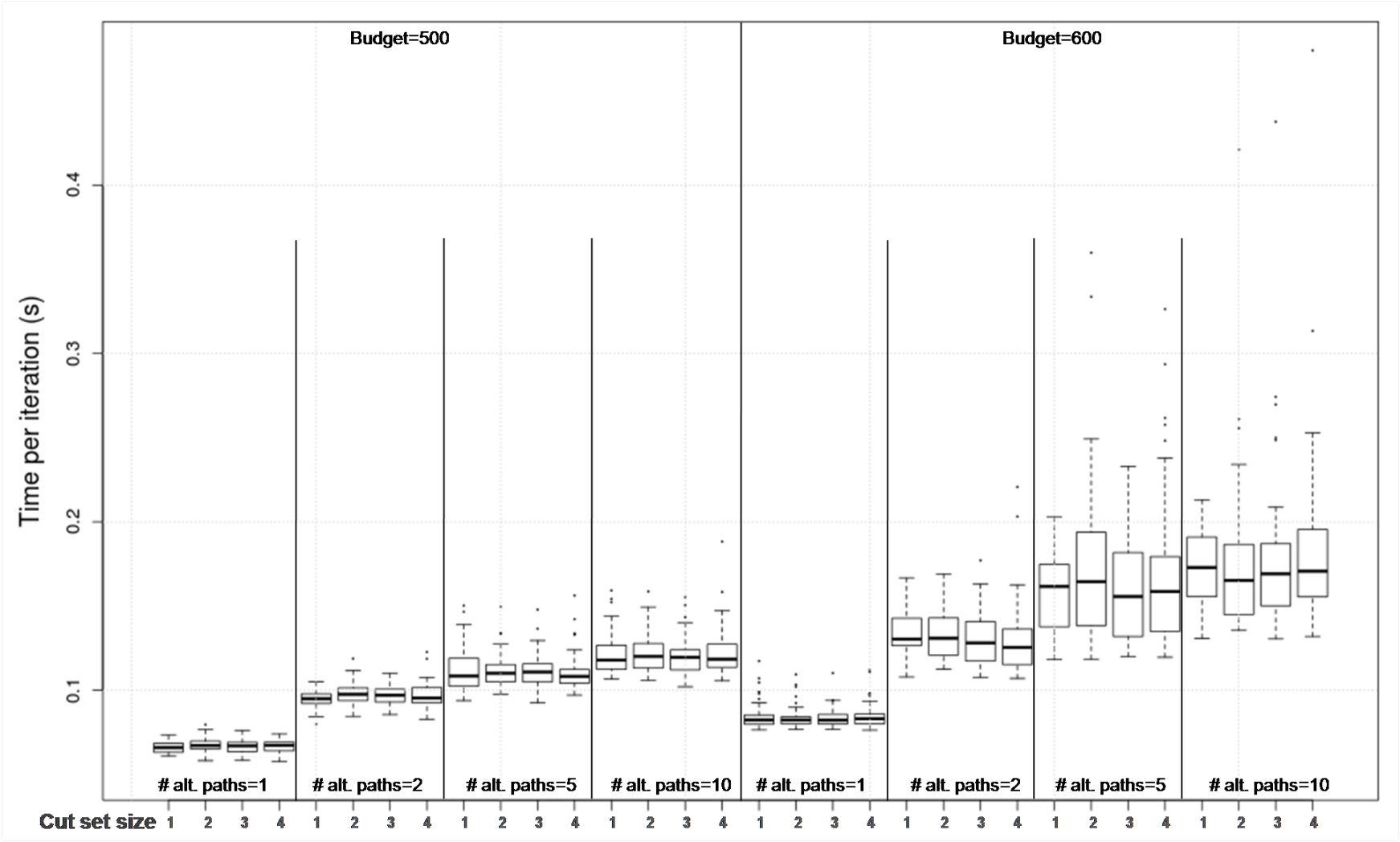


Figure 6.45: Solution time per iteration (one ES generation) comparison of single objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

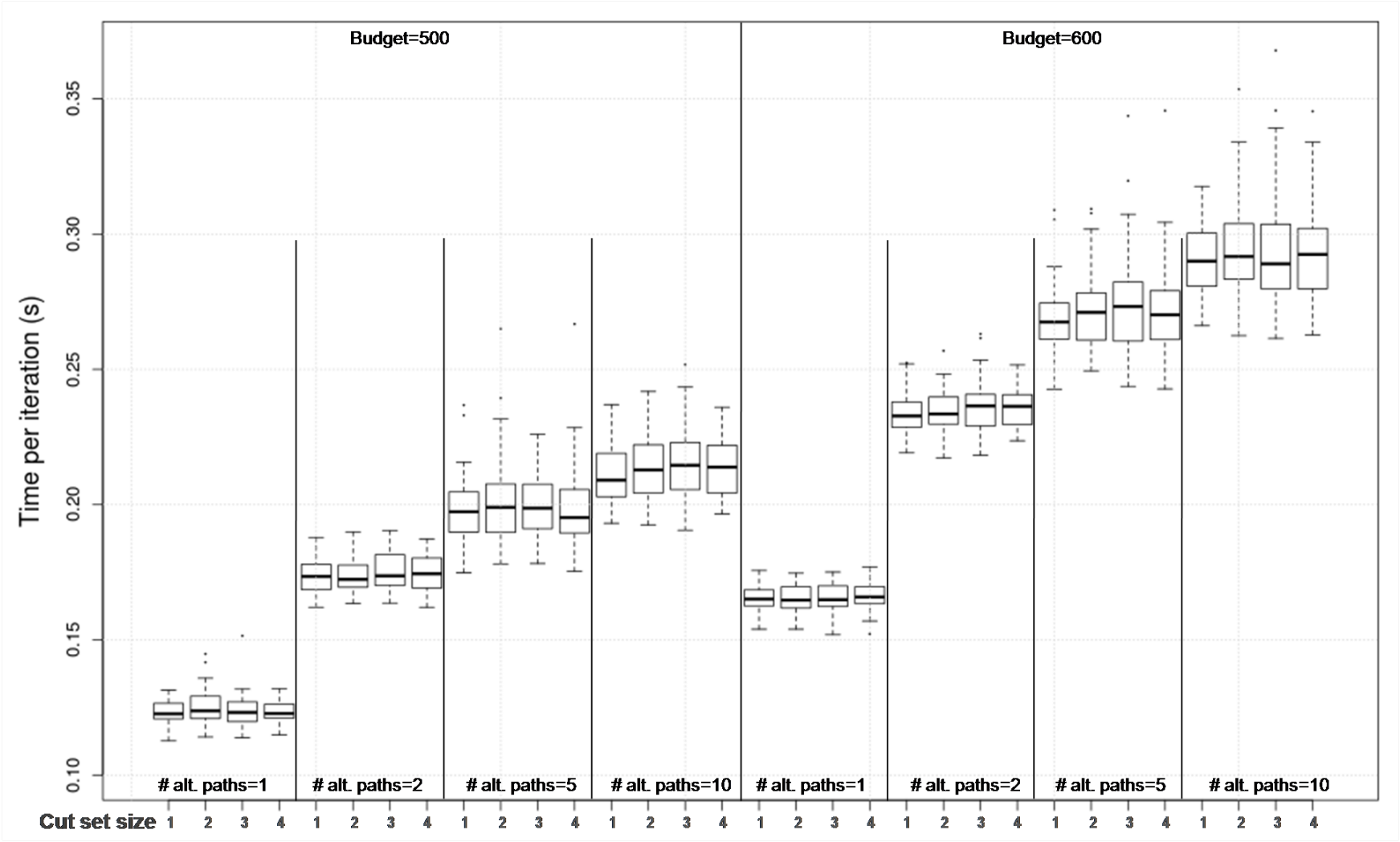


Figure 6.46: Solution time per iteration (one ES generation) comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

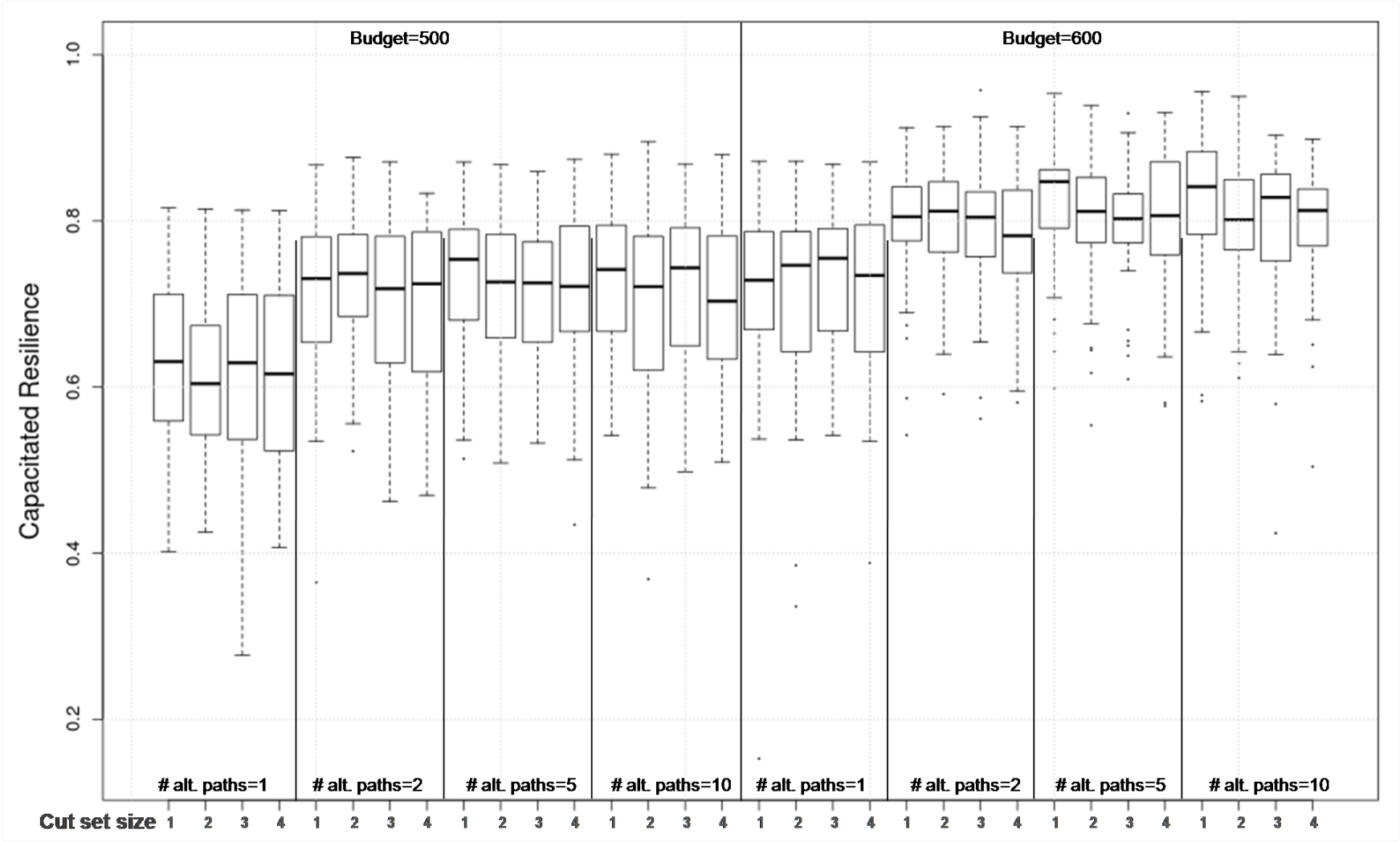


Figure 6.47: Capacitated resilience comparison of single objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

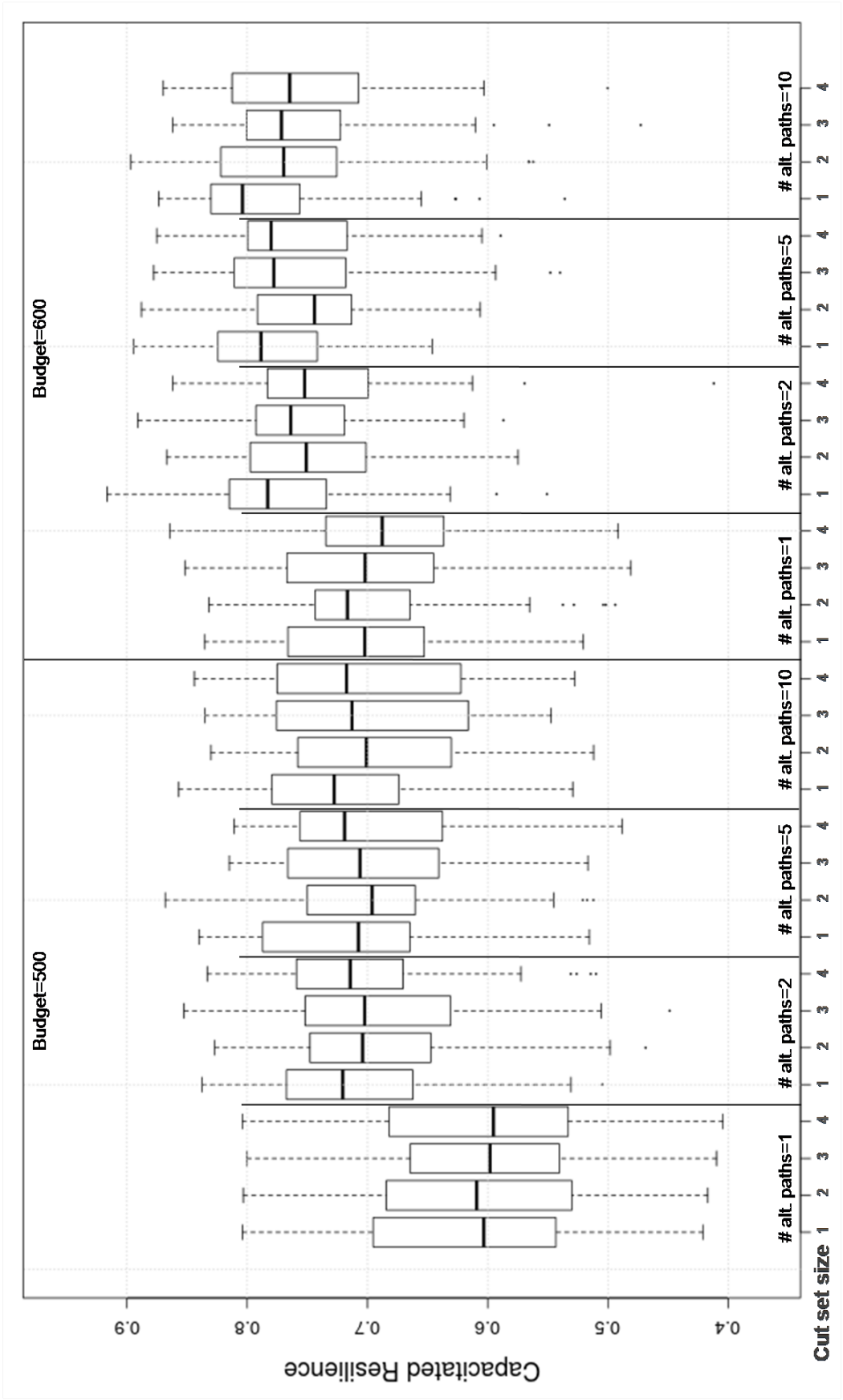


Figure 6.48: Capacitated resilience comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

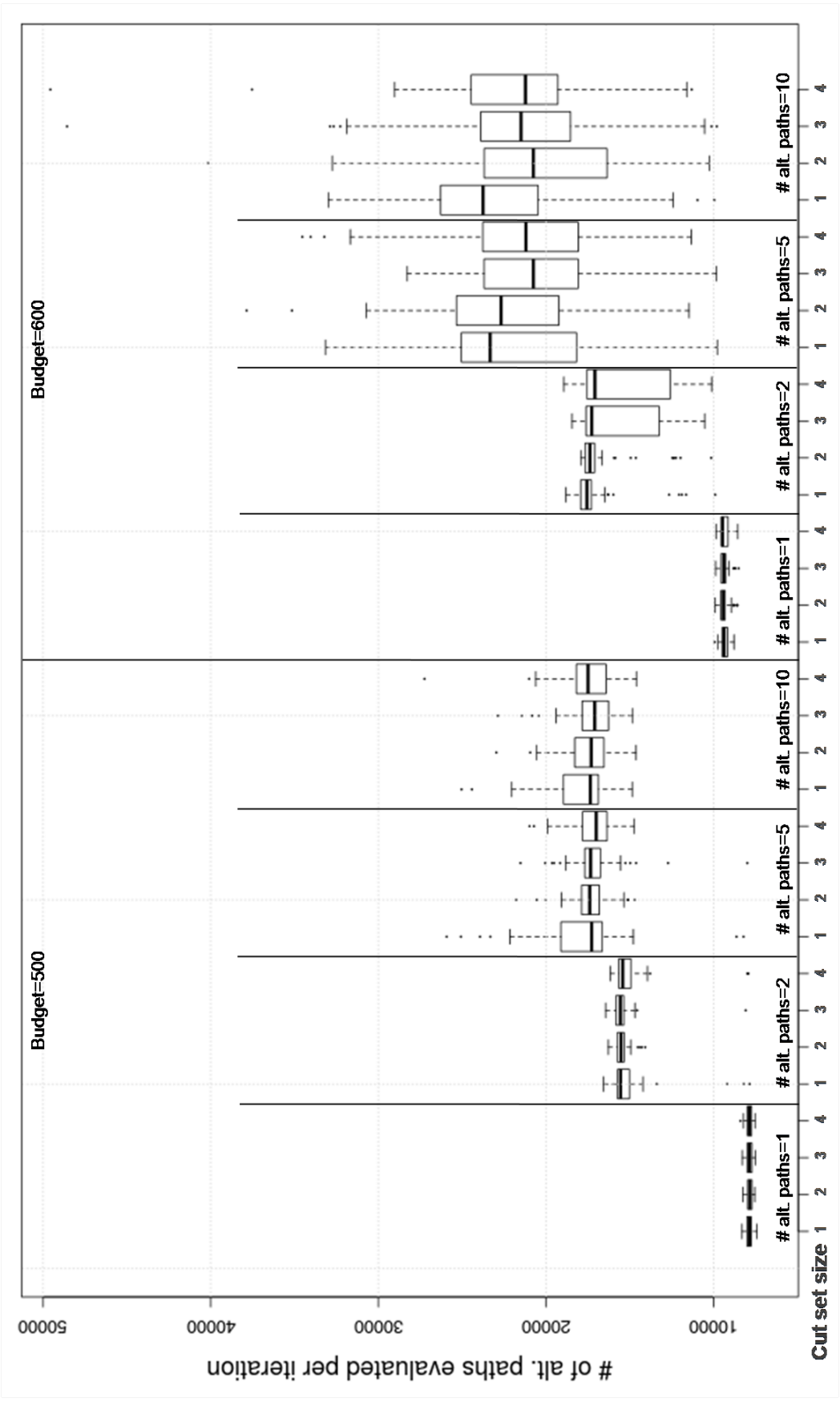


Figure 6.49: Number of evaluated alternative paths per iteration (one ES generation) comparison of single-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

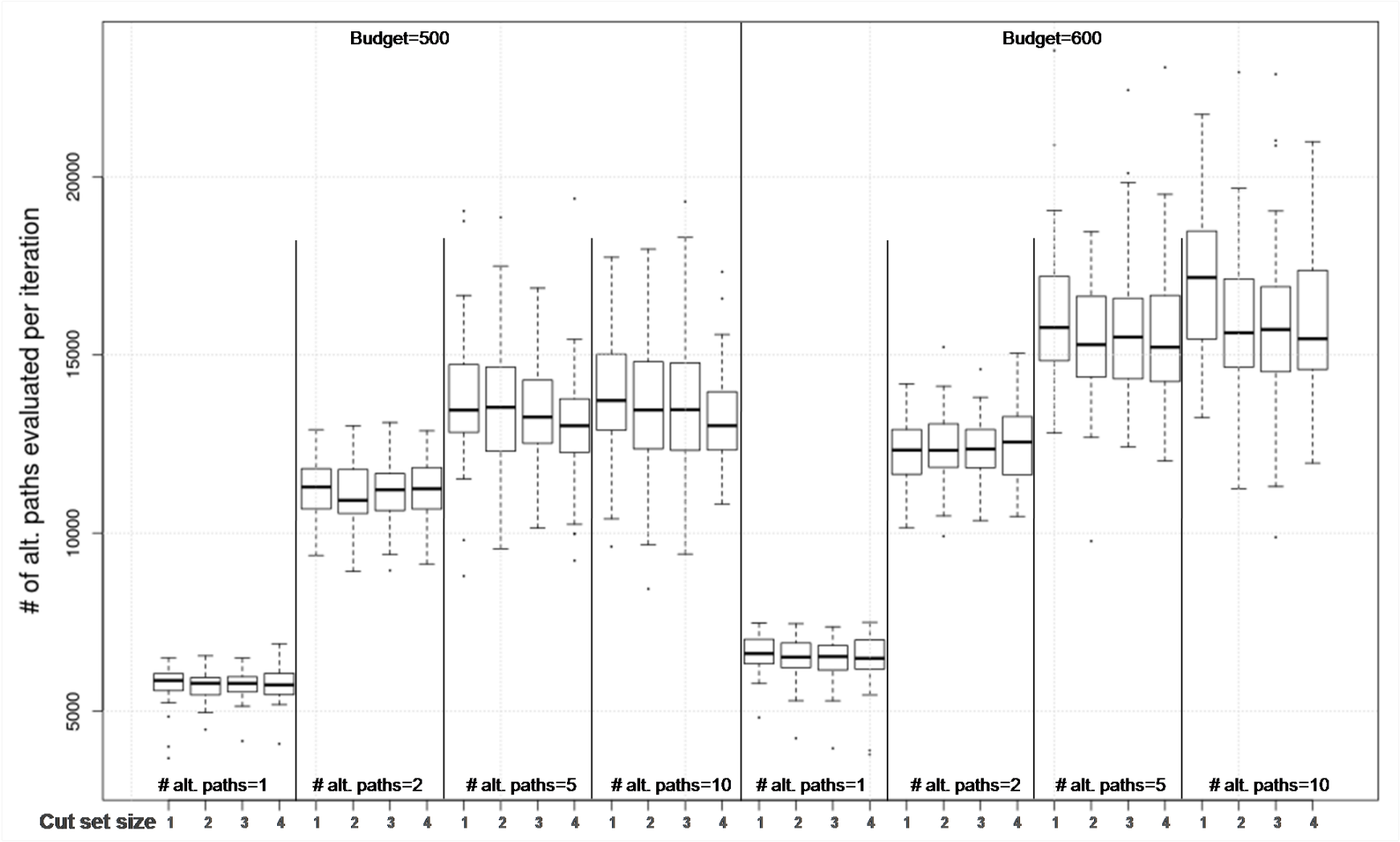


Figure 6.50: Number of evaluated alternative paths per iteration (one ES generation) comparison of bi-objective ES optimized according to different budgets, number of alternative paths and cut set sizes for capacitated resilience for the 10 user problem (10 problem instances, 10 random number seeds each)

6.5.2 Solution time comparison of capacitated resilience: Problem size

Table 6.5 summarizes the solution times of single (capacitated resilience) and bi-objective (cost and capacitated resilience) ES. The smallest problem size is the 10 user and the largest test problem is the 150 user scenario. The solution time difference between single and bi-objective ES increases as the problem size increases. For the 10 user scenario the bi-objective takes 2.89 times longer than the single objective, whereas it takes 4.45 times longer for the 150 user scenario. The reason is that the single objective terminates earlier (1000 generations with early termination of 250 non-improved generations) as the problem size increases, whereas the bi-objective does not terminate (2000 generations with early termination of 500 non-improved generations) as early as the single objective because finding an improved non-dominated solution is easier. According to these results, solving problem instances having more than 200 users is not practical.

Table 6.5: Solution time comparison of single (capacitated resilience) and bi-objective (cost and capacitated resilience) ES for different problem sizes

		Solution Time (s)			
Objective	Scenario	Average	Std. Dev.	Min	Max
Single obj	u10_600_10_4*	170.009	58.589	68.252	480.021
	u25_1200_10_4	858.131	135.665	499.680	1,503.129
	u50_1700_10_4	2,157.812	136.623	1,992.201	2,656.511
	u75_2700_10_4	7,885.627	462.430	7,137.655	8,810.718
	u100_3000_5_4	10,978.277	412.083	10,350.390	12,057.203
	u150_4700_5_4	45,656.592	5,853.616	25,814.137	53,271.754
Bi-obj	u10_600_10_4	492.209	127.283	190.815	667.567
	u25_1200_10_4	2,688.267	454.826	1,589.506	3,309.185
	u50_1700_10_4	7,174.958	1,519.856	3,754.245	9,124.685
	u75_2700_10_4	30,748.788	3,836.392	19,615.756	34,733.680
	u100_3000_5_4	46,205.246	3,294.315	30,808.370	49,085.168
	u150_4700_5_4	203,254.431	13,536.241	154,795.730	220,643.620

* a_b_c_d: a=number of users, b=budget, c=number of alternative paths and d=cut set size

In summary, the solution time of bi-objective is higher than single objective. Solution times of both single and bi-objective ES are sensitive to the parameter value of the number of alternative paths due to total number of evaluated paths. Cut-set size seems to have an

insignificant effect on the solution time for the 10 user scenario. An important result is that the estimation of capacitated resilience with lower number or alternative paths and cut-set sizes (e.g., 2 and 2 instead of 10 and 4, respectively) performs comparably with the exact capacitated resilience calculation and runs faster.

6.5.3 Solution time comparison of all metrics

In this section, the solution times for optimization of all metrics for different problem sizes are summarized. Table 6.6 presents the solution times of the 10 and the 25 user scenarios for single objective (capacitated resilience, TE, two-terminal and all-terminal reliability) and bi-objective (cost and capacitated resilience, and TE and capacitated resilience) ES.

According to these results, two-terminal and all-terminal reliability have the lowest solution times, however, capacitated resilience optimization performs very closely. On the other hand, traffic efficiency is the worst in terms of solution time. Similarly, the bi-objective ES of cost and capacitated resilience is significantly faster than the bi-objective ES of TE and capacitated resilience. Solving bi-objective ES of TE and capacitated resilience is not practical beyond problem sizes larger than 30 users, whereas bi-objective of cost and capacitated resilience can solve for the 100 user scenario within the same time of solving the 25 user scenario for TE and capacitated resilience.

The largest solvable problem sizes within 24 hours for all metrics are given in Table 6.7. All scenarios are for the medium budget for their size (number of users). According to these results, using capacitated resilience can solve larger problems than using TE and its largest solvable problem size is very close to using all-terminal or two-terminal reliabilities.

Table 6.6: Solution time comparison of all metrics for the 10 and the 25 user scenarios

Scenario	Budget	Objective	Method	Solution Time (s) Average	Std. Dev.	Min	Max
10 users	500	Single obj	Capacitated Resilience	108.330	22.588	64.379	188.263
			Traffic Efficiency	16,151.041	3,360.070	9,058.007	22,496.754
			Two-terminal Reliability	96.009	20.574	53.801	145.500
			All-terminal Reliability	96.096	18.203	61.903	153.425
		Bi-obj	Cost and Capacitated Res.	305.470	76.779	141.138	454.692
			TE and Capacitated Res.	30,680.334	8,955.493	14,522.584	48,832.880
	600	Single obj	Capacitated Resilience	170.009	58.590	68.252	480.021
			Traffic Efficiency	21,648.773	4,280.702	10,326.142	30,497.717
			Two-terminal Reliability	138.195	34.506	74.350	211.100
			All-terminal Reliability	139.868	38.629	52.555	231.609
		Bi-obj	Cost and Capacitated Res.	492.209	127.283	190.815	667.567
			TE and Capacitated Res.	38,763.822	3,626.932	32,148.512	46,172.710
25 users	1000	Single obj	Capacitated Resilience	571.159	70.104	388.844	826.490
			Traffic Efficiency	42,517.725	6,287.600	32,546.073	52,489.889
			Two-terminal	538.086	82.644	374.393	682.823
			All-terminal reliability	534.854	59.740	431.624	728.873
		Bi-obj	Cost and Capacitated Res.	1,747.886	345.188	943.294	2,141.489
			TE and Capacitated Res.	141,753.159	13,036.430	119,028.622	150,931.612

Table 6.7: The largest solvable scenarios in 24 hours, given as number of users for a medium budget

Method	# of users in problem
Capacitated resilience*	200
Traffic efficiency	50
Two-terminal reliability	240
All-terminal reliability	220
Bi-objective cost vs. capacitated resilience	125
Bi-objective TE vs. capacitated resilience	20

* Number of alternative paths and cut set size are 10 and 4, respectively.

Chapter 7

Conclusions and Further Research

7.1 Conclusions

In network design, survivability is a challenging but important problem. In the network reliability/survivability literature, many metrics have been proposed. Among them all-terminal reliability, k-terminal reliability, traffic efficiency, k edge-disjoint paths and k node-disjoint paths are commonly used. However, a new survivability metric has been proposed in this dissertation to overcome limitations of the previous metrics and provide a new aspect to survivability.

Capacitated resilience, the proposed metric in this dissertation, considers capacity, reliability and rerouting options simultaneously. This is the main difference of capacitated resilience compared with the other metrics. If a node or an edge does not have enough capacity then a path using that node or edge is not feasible. The availability of rerouting options in case of a failure helps to maintain connectivity and session continuity. Capacitated resilience becomes zero when rerouting is not available. In a wireless network, redundant devices create alternative paths which are the rerouting options. Capacitated resilience uses reliability and scales it with rerouting options to find the true resilience of the network under capacity constraints. Therefore, it allows a comparison of different network designs whereas connectivity based metrics do not allow comparison of different designs. The connectivity based metrics, k edge-disjoint or node-disjoint paths, consider neither reliability nor capacity in the network. They focus on the redundant paths in the network design assuming perfectly reliable nodes and edges. Terminal reliability (k-terminal or all-terminal) focuses on reliability but does not consider capacity. Traffic efficiency considers rerouting options but it does not consider capacity.

In the literature, exact methods to calculate the reliability/survivability metrics have been presented. However, due to intractability of exact methods for even moderately sized problems, many approximation methods or simulation (mostly Monte Carlo simulation) approaches have also been developed. For example, Konak and Bartolacci (2007) proposed a simulation based method to estimate TE. In this dissertation, an exact method to calculate capacitated resilience is given as well as an approximation method. The exact calculation is based on the k-shortest path calculation and cut-set identification. However, a fast and effective estimation of capacitated resilience is obtained by changing the values of k and the size of the cut-set.

In this dissertation both single and bi-objective optimization are considered. The Evolutionary Strategies method has been used to solve the network design problem because of its success on nonlinear problems for which traditional optimization methods (e.g., mixed integer programming) do not perform satisfactorily. Also, ES works best with continuous problems and the network design problem in this dissertation has device coordinates as continuous decision variables. Besides, ES can easily be extended to multiobjective optimization, which is one of the contributions of this dissertation. Single objective optimization solves for maximum capacitated resilience, traffic efficiency, two-terminal reliability and all-terminal reliability. Bi-objective optimization simultaneously optimizes cost and capacitated resilience because a main constraint of real life projects is the budget. Pareto optimality is used for bi-objective optimization. A well known multiobjective method, NSGA-II (Deb et al., 2002), has been adapted for the proposed bi-objective ES model in this dissertation.

Different problem instances were solved for capacitated resilience as well as the other metrics and the resulting network designs were compared. The network designs obtained by optimization for capacitated resilience have structures such that the high traffic users are prioritized. Redundancies are allocated as the budget allows but the redundant devices are mostly allocated around the high traffic users to ensure maximum capacitated resilience. The other metrics prioritized reliable connections for all users (starting from the high traffic

ones) and redundancies are considered as a secondary objective. Therefore, it can be concluded that capacitated resilience prioritizes the allocation of redundancies (survivability) considering the capacities.

From the network survivability perspective, it is interesting to consider the performance of the different designs in case of failures and attacks. Obviously, a planned attack has a more severe effect on the network than a random attack (or failure). An attack on the network may aim for removal (or elimination) of some edges (or nodes) to disconnect users (the max-flow min-cut problem). Capacitated resilience considers cut-sets in its calculation and its primary design goal is to create redundancies to maintain connectivity in case of a failure. Since capacitated resilience maximizes survivability by taking cut-sets into account, it provides a resilient design that minimizes the adverse effects of planned or unplanned attack.

In this dissertation, heterogeneous wireless networks are considered. Design of survivable heterogeneous wireless networks is a new area in optimization which has gained popularity because of the growing use of new telecommunication technologies such as 4G and wireless hotspots. Heterogeneity is defined as the differences in both offered services and device properties in a wireless network. However, capacitated resilience and its design rules can be applied to any network including homogeneous ones (such as military networks, transportation networks, communication networks or electrical networks) to ensure resilience and survivability.

7.2 Further research

For future research, node failures can be added to capacitated resilience calculation. In the current model presented in this dissertation the nodes are assumed to be perfectly reliable. Including node failures requires the independent subgroup identification process (Section 3.2.3) to be redesigned because common nodes among different independent subgroups create dependencies. To solve this, two independent subgroups having one (or more)

common node should be merged. This may increase computational effort by reducing the number but increasing the size of independent subgroups.

As an interesting extension, the effect of interference can be included in the calculation of capacitated resilience and other metrics. Currently, wireless interference is not considered as a parameter. Different wireless channels are assumed to be assigned for each wireless link within the same communication range to eliminate interference. Although interference has been mostly considered negligible in the network reliability/survivability literature, as discussed in Section 2.5, it may affect the reliabilities of wireless links in dense networks. Addition of the wireless channel assignment as a decision variable (this also requires the routing algorithm to be changed accordingly) and including interference in the wireless link reliability calculation would make the model presented in this dissertation more realistic.

As discussed in Section 4.1.4, the current assignment of users to devices is done in a randomized order. Although this is not an issue for most cases, for a few instances it can lead to suboptimal user-device assignments if the capacity of the network is restricted. The obvious, but not ideal, solution for this user-device assignment issue would be considering all possible combinations of user to device assignments (enumeration) and choosing the best one. However, this approach would make the proposed model intractable for moderate to large size problem instances. An alternative heuristic method could be devised as future work.

As another change in the proposed method, the use of Monte Carlo simulation will be investigated for calculation of the reliability of independent subgroups. In the proposed model, the reliability is calculated by identification of minimal cut-sets which provides a lower bound of reliability (worst case). Simulation may provide the expected value of the reliability of the subgroups, however, it is computationally more expensive.

Network failures and attacks are two important topics in the survivable network design literature. As future work, the network designs obtained by optimization for capacitated resilience and the other metrics could be compared in terms of network attacks (that is,

planned interdictions) or different failure scenarios. The proposed metric, capacitated resilience, could be applied to other networks, such as transportation networks or military supply networks.

In this dissertation, the locations of users are fixed. However, in real life the users are mobile for wireless network applications. Therefore, to be more realistic, mobility of the users could be included in the proposed model.

One of the most popular topics in today's computational optimization area is to use parallel computing. It gained popularity as devices with multiple core processors became widespread after 2000. The solution approach presented in this dissertation and calculation methods of the other metrics are computationally expensive. Effective use of parallel computing could reduce solution times dramatically. Parallel computing will be even more important in the future as the parallel use of the GPUs (Graphics Processing Unit) and the CPUs (Central Processing Unit) are becoming popular. Therefore, adaptation of the solution method of this dissertation to parallel computing would be an interesting extension.

Appendices

Appendix A

Input data of problems

All problem data (10, 25, 50, 75, 100, 150 user scenarios) are given. Each scenario consists of 10 problem instances. Coordinates and traffic requirement of each user are given as an array of size three, [x, y, traffic requirement].

A.1 The 10 user scenario

Problem instance 1: [0.88, 2.696, 0.013], [3.467, 0.078, 5.11], [-3.432, -2.914, 19.153], [-1.689, -2.242, 14.729], [3.571, -3.636, 14.45], [1.098, 3.948, 15.753], [1.333, -3.518, 19.662], [-3.518, -0.173, 14.269], [-3.227, -0.516, 3.737], [-1.684, 2.159, 14.803]

Problem instance 2: [0.425, 2.529, 6.56], [1.242, -3.97, 9.247], [2.913, 1.513, 17.002], [-2.775, -1.519, 12.444], [0.649, 0.403, 4.59], [3.559, -3.492, 0.998], [-1.307, 2.951, 5.34], [-2.467, 3.871, 9.607], [1.285, 0.367, 17.313], [-1.424, -0.027, 3.466]

Problem instance 3: [-3.447, 2.797, 0.317], [-2.521, 2.604, 8.453], [1.481, -3.948, 9.813], [-0.72, 0.644, 13.425], [0.864, 3.667, 3.374], [0.85, 2.552, 3.359], [3.91, 0.577, 14.871], [-3.499, 2.782, 18.653], [-3.675, 0.586, 2.583], [-0.111, -2.855, 18.295]

Problem instance 4: [2.065, -3.872, 4.379], [-1.016, -0.351, 0.317], [1.165, 3.918, 5.042], [0.292, 0.695, 13.085], [3.838, -3.714, 1.632], [2.008, -0.472, 3.663], [0.525, -2.181, 0.488], [2.914, 2.585, 12.139], [1.778, -2.903, 18.476], [0.906, -2.608, 13.341]

Problem instance 5: [-2.095, -1.775, 5.265], [2.815, -0.196, 16.87], [1.497, -0.779, 15.504], [-1.886, -1.92, 2.493], [2.775, -2.977, 9.111], [-1.996, 3.514, 7.032], [1.765, 2.797, 0.603], [3.243, -1.44, 11.039], [-2.868, -1.304, 2.82], [2.098, 1.215, 5.066]

Problem instance 6: [-2.764, -3.462, 1.966], [1.331, 2.051, 4.234], [2.643, 3.312, 12.872], [-0.611, -2.715, 19.712], [3.23, 1.423, 14.972], [-3.822, -1.639, 5.411], [-3.382, 3.901, 16.055], [0.416, 1.244, 14.181], [-1.236, -3.566, 9.328], [0.882, 1.563, 2.772]

Problem instance 7: [-0.941, -3.71, 4.635], [1.681, 0.351, 18.569], [3.508, 2.687, 19.517], [3.035, 0.02, 19.447], [-2.124, -1.193, 2.273], [3.853, -2.038, 8.534], [2.469, 2.661, 3.19], [-3.177, -3.079, 15.768], [-3.33, -2.657, 18.658], [-0.477, -2.643, 8.68]

Problem instance 8: [-3.041, 2.415, 12.086], [-1.889, 3.811, 11.329], [1.091, 2.325, 18.516], [2.073, -0.832, 4.929], [1.031, -0.61, 14.808], [0.086, 1.011, 0.342], [-0.998, 0.798, 10.204], [1.432, -2.797, 1.415], [0.207, -3.128, 12.247], [-0.151, -0.136, 17.919]

Problem instance 9: [-1.034, -3.91, 0.69], [-1.066, 1.384, 19.162], [-2.36, -3.127, 4.057], [0.787, -1.283, 5.559], [2.27, -3.082, 0.4], [-2.979, -1.388, 7.635], [-1.676, -2.389, 0.179], [-0.238, 1.017, 9.586], [-2.458, -3.617, 19.424], [-2.329, 3.876, 4.893]

Problem instance 10: [0.931, 2.503, 2.432], [-0.405, -0.237, 8.911], [2.67, 2.788, 4.89], [1.548, -1.768, 19.765], [0.077, 0.535, 11.362], [1.731, 0.163, 13.669], [-3.519, 2.104, 16.373], [1.195, 2.416, 11.378], [-1.397, -3.387, 8.271], [3.657, 1.635, 19.93]

A.2 The 25 user scenario

Problem instance 1: [1.391, 4.263, 0.013], [5.481, 0.123, 5.11], [-5.426, -4.608, 19.153], [-2.671, -3.544, 14.729], [5.647, -5.749, 14.45], [1.737, 6.242, 15.753], [2.107, -5.562, 19.662], [-5.562, -0.273, 14.269], [-5.102, -0.816, 3.737], [-2.663, 3.414, 14.803], [-1.794, 0.083, 8.585], [-1.765, -6.3, 12.228], [-4.653, 0.425, 1.925], [0.107, -3.39, 10.052], [5.823, -2.713, 12.703], [0.624, -3.538, 1.513], [4.668, 3.128, 4.606], [3.592, -1.387, 14.304], [-5.913, 0.436, 10.943], [-6.245, -3.151, 10.304], [6.009, -5.457, 5.83], [1.177, 2.556, 0.042], [2.594, 4.961, 0.755], [-6.183, 5.84, 7.282], [0.603, 3.298, 6.955]

Problem instance 2: [-5.566, 5.313, 16.264], [-3.916, -3.112, 1.571], [-2.464, 2.949, 16.276], [4.19, -1.131, 16.322], [0.502, 4.679, 11.183], [3.039, -1.857, 10.679], [-0.306, 3.989, 9.016], [0.757, 6.284, 13.796], [-4.933, -1.147, 5.947], [5.54, 0.711, 12.313], [-2.606, 1.93, 9.247], [-4.703, 1.202, 10.161], [2.93, -4.495, 2.67], [4.17, 2.936, 0.845], [2.111, 5.758, 18.317], [4.327, 0.98, 10.013], [-0.483, 5.198, 16.846], [-3.782, 3.959, 8.267], [5.978, 0.944, 0.845], [-5.419, 3.253, 11.699], [4.47, -0.273, 2.33], [-4.352, -2.492, 19.088], [-5.81, 4.619, 9.598], [2.951, 4.91, 9.552], [4.872, 4.065, 8.333]

Problem instance 3: [-5.45, 4.422, 0.317], [-3.986, 4.118, 8.453], [2.342, -6.243, 9.813], [-1.139, 1.018, 13.425], [1.365, 5.798, 3.374], [1.344, 4.035, 3.359], [6.183, 0.912, 14.871], [-5.533, 4.399, 18.653], [-5.811, 0.926, 2.583], [-0.175, -4.515, 18.295], [-0.375, -1.563, 6.404], [2.462, 1.265, 19.06], [5.013, 3.641, 7.024], [0.755, -3.928, 18.024], [1.791, 3.54, 5.174], [0.11, -2.224, 17.889], [-3.362, -2.509, 17.724], [-0.892, -1.127, 10.177], [-3.769, 5.502, 0.987], [1.736, 0.437, 15.58], [-2.44, -0.885, 17.143], [1.644, -2.805, 9.306], [4.615, -3.753, 18.278], [2.696, 5.115, 5.954], [-2.939, -0.918, 14.029]

Problem instance 4: [3.265, -6.121, 4.379], [-1.606, -0.556, 0.317], [1.841, 6.195, 5.042], [0.461, 1.099, 13.085], [6.068, -5.872, 1.632], [3.174, -0.746, 3.663], [0.83, -3.448, 0.488], [4.608,

4.087, 12.139], [2.812, -4.591, 18.476], [1.433, -4.124, 13.341], [1.393, 3.296, 16.015], [-0.014, -1.685, 4.304], [-2.287, 5.521, 0.392], [-2.888, 5.582, 0.617], [-4.652, 3.927, 18.274], [3.082, -3.432, 17.171], [5.19, 6.276, 7.79], [-6.02, 4.863, 11.239], [-3.875, 1.856, 18.628], [-2.823, -5.527, 8.739], [-4.983, 2.961, 3.949], [3.221, 3.574, 8.156], [-3.066, 1.617, 4.196], [-1.044, -2.5, 4.303], [-0.53, 1.551, 2.976]

Problem instance 5: [-3.312, -2.807, 5.265], [4.451, -0.309, 16.87], [2.367, -1.232, 15.504], [-2.982, -3.035, 2.493], [4.387, -4.707, 9.111], [-3.157, 5.556, 7.032], [2.791, 4.422, 0.603], [5.128, -2.277, 11.039], [-4.535, -2.062, 2.82], [3.317, 1.922, 5.066], [-1.397, 3.963, 13.456], [-2.088, -3.56, 0.458], [-4.108, -6.32, 9.832], [-3.632, 5.126, 12.081], [-0.112, -4.312, 13.064], [4.601, -1.721, 17.17], [-3.599, 4.186, 6.903], [2.646, 5.441, 2.404], [4.423, 6.196, 9.041], [4.859, 2.084, 6.417], [0.037, -0.233, 18.423], [-4.411, 2.11, 15.584], [-0.403, 2.886, 10.464], [3.847, 1.455, 5.727], [4.88, -3.506, 7.859]

Problem instance 6: [-4.371, -5.474, 1.966], [2.105, 3.242, 4.234], [4.179, 5.236, 12.872], [-0.967, -4.293, 19.712], [5.107, 2.25, 14.972], [-6.043, -2.592, 5.411], [-5.348, 6.168, 16.055], [0.659, 1.967, 14.181], [-1.955, -5.638, 9.328], [1.394, 2.471, 2.772], [5.51, 4.801, 10.97], [-6.127, 5.125, 3.239], [4.912, 5.017, 19.512], [-2.181, -0.594, 9.299], [2.377, 5.962, 3.182], [-3.123, -2.471, 13.902], [5.171, 0.651, 10.818], [5.858, -2.852, 5.917], [0.691, 6.203, 6.802], [-1.6, 5.693, 4.677], [3.335, -6.136, 16.245], [-6.044, 5.252, 9.539], [-1.857, -6.269, 12.376], [-0.159, -1.381, 6.3], [2.656, -1.048, 3.789]

Problem instance 7: [-1.488, -5.866, 4.635], [2.657, 0.555, 18.569], [5.547, 4.248, 19.517], [4.799, 0.032, 19.447], [-3.358, -1.886, 2.273], [6.092, -3.222, 8.534], [3.904, 4.208, 3.19], [-5.023, -4.869, 15.768], [-5.265, -4.2, 18.658], [-0.755, -4.179, 8.68], [4.321, -4.504, 5.384], [-0.63, -3.094, 14.327], [-4.734, 5.981, 15.098], [1.274, -4.947, 19.668], [-5.521, 5.805, 17.34], [5.828, 0.364, 0.627], [-0.755, 0.014, 9.008], [0.303, -4.944, 17.036], [-5.534, 3.857,

14.227], [-5.343, 2.404, 4.151], [3.73, 4.34, 11.503], [1.338, -0.659, 12.834], [-1.142, -0.483, 5.835], [-2.187, -3.158, 6.496], [-5.752, -2.686, 15.109]

Problem instance 8: [-4.809, 3.818, 12.086], [-2.987, 6.026, 11.329], [1.725, 3.676, 18.516], [3.277, -1.315, 4.929], [1.631, -0.965, 14.808], [0.136, 1.598, 0.342], [-1.578, 1.261, 10.204], [2.265, -4.423, 1.415], [0.327, -4.946, 12.247], [-0.239, -0.214, 17.919], [-4.47, -1.628, 17.803], [-1.982, -3.592, 9.652], [-2.152, 6.195, 3.955], [-0.285, 4.425, 0.322], [1.308, 4.117, 2.191], [1.24, 2.583, 18.785], [-4.832, 5.997, 2.62], [5.004, -2.876, 4.381], [-1.251, 1.814, 14.071], [-0.995, -2.462, 13.04], [2.772, 2.814, 7.315], [1.45, -1.239, 9.32], [2.168, 4.116, 19.273], [4.811, 5.321, 19.853], [-4.593, 5.806, 0.627]

Problem instance 9: [-1.635, -6.182, 0.69], [-1.686, 2.188, 19.162], [-3.731, -4.945, 4.057], [1.244, -2.029, 5.559], [3.59, -4.873, 0.4], [-4.711, -2.195, 7.635], [-2.651, -3.778, 0.179], [-0.377, 1.607, 9.586], [-3.887, -5.719, 19.424], [-3.682, 6.128, 4.893], [-2.285, -2.343, 5.256], [1.849, -1.127, 8.19], [1.761, 1.058, 4.807], [2.575, 2.063, 4.96], [-5.249, -5.235, 14.013], [6.193, 2.75, 13.133], [-5.673, 5.592, 1.832], [5.16, 1.674, 15.116], [-6.005, 3.459, 0.088], [-0.004, -3.038, 1.035], [-1.479, -0.532, 1.893], [-4.764, 3.13, 19.831], [-0.283, -2.142, 15.42], [0.246, -2.758, 18.417], [2.146, -3.154, 15.359]

Problem instance 10: [1.472, 3.958, 2.432], [-0.641, -0.375, 8.911], [4.222, 4.409, 4.89], [2.448, -2.796, 19.765], [0.122, 0.846, 11.362], [2.736, 0.258, 13.669], [-5.563, 3.326, 16.373], [1.889, 3.821, 11.378], [-2.208, -5.355, 8.271], [5.783, 2.585, 19.93], [-3.586, 5.174, 0.389], [-0.618, -2.202, 15.35], [0.968, -4.059, 7.338], [-0.584, 5.487, 7.232], [-1.206, 3.216, 18.935], [-5.609, -6.124, 16.754], [-2.32, 1.538, 13.13], [-2.393, -2.823, 4.36], [3.066, -0.899, 16.088], [-5.305, -4.712, 16.385], [-5.085, -2.181, 8.741], [1.482, -0.739, 5.719], [-3.308, 4.405, 17.658], [0.803, 0.9, 6.896], [5.483, -0.384, 6.717]

A.3 The 50 user scenario

Problem instance 1: [1.967, 6.028, 0.013], [7.752, 0.173, 5.11], [-7.673, -6.517, 19.153], [-3.777, -5.012, 14.729], [7.986, -8.131, 14.45], [2.456, 8.827, 15.753], [2.98, -7.866, 19.662], [-7.866, -0.386, 14.269], [-7.215, -1.155, 3.737], [-3.766, 4.828, 14.803], [-2.536, 0.117, 8.585], [-2.496, -8.909, 12.228], [-6.58, 0.601, 1.925], [0.151, -4.794, 10.052], [8.235, -3.837, 12.703], [0.882, -5.004, 1.513], [6.601, 4.424, 4.606], [5.08, -1.961, 14.304], [-8.362, 0.616, 10.943], [-8.832, -4.456, 10.304], [8.498, -7.717, 5.83], [1.665, 3.615, 0.042], [3.669, 7.016, 0.755], [-8.744, 8.259, 7.282], [0.853, 4.664, 6.955], [-5.683, 7.133, 17.293], [5.72, 2.821, 9.802], [-8.359, -3.888, 7.194], [6.145, -8.099, 18.491], [0.071, -8.477, 4.166], [-6.133, -8.925, 9.775], [-8.537, 6.21, 14.335], [-0.172, -2.864, 13.144], [-2.63, 0.086, 17.368], [-7.151, 1.85, 1.009], [0.984, -6.537, 19.711], [-4.739, 7.82, 7.03], [5.741, -6.644, 19.099], [3.277, -4.065, 8.193], [8.564, 7.878, 0.284], [3.931, -0.811, 10.279], [-0.602, -4.814, 19.315], [-3.122, -7.569, 13.917], [-5.749, -2.906, 12.623], [5.612, -1.037, 18.098], [-3.058, -7.471, 16.108], [4.954, 2.657, 18.198], [-2.154, 7.527, 12.932], [-3.58, -4.181, 1.301], [-2.822, 6.504, 12.931]

Problem instance 2: [-7.872, 7.514, 16.264], [-5.538, -4.401, 1.571], [-3.485, 4.17, 16.276], [5.925, -1.599, 16.322], [0.71, 6.618, 11.183], [4.297, -2.626, 10.679], [-0.433, 5.641, 9.016], [1.071, 8.887, 13.796], [-6.976, -1.622, 5.947], [7.835, 1.006, 12.313], [-3.685, 2.729, 9.247], [-6.651, 1.7, 10.161], [4.144, -6.356, 2.67], [5.898, 4.152, 0.845], [2.985, 8.143, 18.317], [6.119, 1.386, 10.013], [-0.683, 7.351, 16.846], [-5.349, 5.599, 8.267], [8.454, 1.336, 0.845], [-7.664, 4.6, 11.699], [6.321, -0.386, 2.33], [-6.154, -3.524, 19.088], [-8.216, 6.532, 9.598], [4.173, 6.943, 9.552], [6.89, 5.748, 8.333], [4.707, 1.239, 19.653], [-5.624, 1.501, 9.7], [-1.807, 8.596, 0.65], [-0.064, -0.291, 16.267], [-2.304, 2.736, 1.652], [-0.458, 7.4, 6.83], [6.017, 6.694, 12.227], [3.208, 3.439, 11.02], [4.516, -3.939, 1.113], [0.853, 1.728, 9.313], [-2.23, -6.862, 11.642], [-2.95, 2.502, 0.817], [4.774, -0.301, 0.667], [4.099, -8.193, 7.482], [7.399, 7.657, 2.596], [-5.54, 3.527, 8.587], [-5.854, 1.03, 5.198], [-5.356, 7.693, 17.39], [-1.275, -7.606, 7.763], [8.217, -2.825, 13.669], [8.022, -7.272, 18.338], [2.967, 7.497, 7.284], [3.305, -4.26, 18.8], [-7.35, 3.19, 11.776],

[-2.737, -2.266, 3.255]

Problem instance 3: [-7.708, 6.254, 0.317], [-5.637, 5.823, 8.453], [3.313, -8.828, 9.813], [-1.611, 1.439, 13.425], [1.931, 8.199, 3.374], [1.9, 5.706, 3.359], [8.744, 1.29, 14.871], [-7.825, 6.221, 18.653], [-8.218, 1.31, 2.583], [-0.247, -6.385, 18.295], [-0.531, -2.21, 6.404], [3.481, 1.789, 19.06], [7.089, 5.149, 7.024], [1.068, -5.555, 18.024], [2.533, 5.006, 5.174], [0.156, -3.145, 17.889], [-4.754, -3.549, 17.724], [-1.262, -1.594, 10.177], [-5.331, 7.781, 0.987], [2.455, 0.618, 15.58], [-3.45, -1.252, 17.143], [2.325, -3.967, 9.306], [6.527, -5.307, 18.278], [3.813, 7.233, 5.954], [-4.156, -1.298, 14.029], [-4.834, 5.622, 0.981], [-0.569, -6.238, 3.029], [5.733, -8.013, 14.799], [5.123, -7.712, 1.984], [1.087, -5.509, 2.361], [-7.378, 6.138, 13.116], [-5.294, -3.467, 2.613], [-0.41, 4.894, 6.345], [-2.528, -4.958, 7.044], [5.87, -8.054, 8.951], [-6.493, -7.845, 18.027], [-1.895, -8.677, 11.516], [6.33, 0.578, 18.171], [5.075, -1.556, 7.65], [3.887, -7.349, 7.521], [3.671, 3.462, 19.24], [8.672, 8.32, 8.083], [-1.701, 5.357, 15.402], [-7.216, -0.743, 11.301], [4.885, 8.199, 4.504], [-1.903, 3.86, 18.638], [-4.729, -2.005, 11.642], [6.873, 3.528, 17.364], [-6.35, 2.27, 1.108], [-7.854, -5.837, 13.617]

Problem instance 4: [4.618, -8.657, 4.379], [-2.271, -0.786, 0.317], [2.604, 8.761, 5.042], [0.653, 1.555, 13.085], [8.582, -8.305, 1.632], [4.489, -1.055, 3.663], [1.174, -4.876, 0.488], [6.517, 5.78, 12.139], [3.977, -6.492, 18.476], [2.027, -5.832, 13.341], [1.97, 4.661, 16.015], [-0.02, -2.383, 4.304], [-3.234, 7.808, 0.392], [-4.084, 7.894, 0.617], [-6.579, 5.553, 18.274], [4.359, -4.853, 17.171], [7.34, 8.876, 7.79], [-8.514, 6.877, 11.239], [-5.48, 2.625, 18.628], [-3.993, -7.816, 8.739], [-7.047, 4.187, 3.949], [4.555, 5.054, 8.156], [-4.336, 2.287, 4.196], [-1.476, -3.535, 4.303], [-0.75, 2.194, 2.976], [3.053, -8.532, 14.047], [2.013, 8.305, 7.218], [5.133, 1.917, 14.707], [2.096, -3.251, 11.635], [0.841, -7.685, 12.252], [4.627, -0.525, 16.254], [6.964, -1.883, 15.11], [-8.234, -1.653, 2.702], [-6.237, -1.709, 2.029], [-1.878, 6.714, 19.324], [-0.986, -3.293, 11.953], [7.185, -5.307, 7.675], [-2.363, -1.602, 12.774], [2.46, -6.027, 7.67], [0.885, -2.879, 0.047], [-8.478, 4.651, 10.272], [-7.095, -6.568, 7.34], [-3.179, 8.696, 16.591], [-1.958, 0.993,

15.162], [1.617, 0.789, 10.452], [-3.84, -3.383, 9.934], [-5.715, -4.648, 10.825], [5.703, 2.275, 0.946], [7.316, -5.501, 10.487], [1.434, -7.758, 8.439]

Problem instance 5: [-4.684, -3.969, 5.265], [6.295, -0.437, 16.87], [3.348, -1.743, 15.504], [-4.217, -4.292, 2.493], [6.205, -6.656, 9.111], [-4.464, 7.857, 7.032], [3.947, 6.253, 0.603], [7.252, -3.221, 11.039], [-6.413, -2.917, 2.82], [4.691, 2.718, 5.066], [-1.975, 5.605, 13.456], [-2.953, -5.034, 0.458], [-5.81, -8.937, 9.832], [-5.137, 7.249, 12.081], [-0.158, -6.099, 13.064], [6.507, -2.434, 17.17], [-5.089, 5.919, 6.903], [3.742, 7.695, 2.404], [6.256, 8.762, 9.041], [6.872, 2.947, 6.417], [0.052, -0.33, 18.423], [-6.238, 2.984, 15.584], [-0.57, 4.081, 10.464], [5.44, 2.058, 5.727], [6.901, -4.958, 7.859], [-1.092, -4.252, 11.065], [-5.405, 7.442, 12.474], [0.478, -1.797, 2.861], [-2.722, 8.343, 2.661], [4.818, 6.089, 11.968], [1.895, -4.747, 8.615], [-6.239, 7.166, 10.234], [-1.863, -6.647, 12.609], [2.76, 8.864, 4.975], [-2.446, -6.652, 5.043], [0.851, -0.323, 1.589], [2.698, -8.921, 7.352], [4.437, 0.886, 1.715], [-1.365, 1.375, 7.809], [-5.697, 1.62, 10.048], [6.568, -4.313, 5.129], [6.562, 8.646, 5.473], [-0.543, 4.53, 2.858], [3.465, -8.344, 19.529], [-5.885, 3.2, 2.279], [-2.542, -7.456, 6.143], [8.518, 1.808, 11.395], [7.08, 7.618, 0.966], [-6.254, 6.745, 1.138], [-7.571, 4.247, 5.406]

Problem instance 6: [-6.181, -7.741, 1.966], [2.977, 4.585, 4.234], [5.91, 7.405, 12.872], [-1.367, -6.071, 19.712], [7.222, 3.181, 14.972], [-8.546, -3.665, 5.411], [-7.563, 8.722, 16.055], [0.931, 2.781, 14.181], [-2.764, -7.973, 9.328], [1.972, 3.495, 2.772], [7.792, 6.789, 10.97], [-8.665, 7.248, 3.239], [6.946, 7.095, 19.512], [-3.084, -0.84, 9.299], [3.362, 8.431, 3.182], [-4.417, -3.495, 13.902], [7.313, 0.92, 10.818], [8.284, -4.033, 5.917], [0.978, 8.772, 6.802], [-2.263, 8.051, 4.677], [4.716, -8.678, 16.245], [-8.547, 7.428, 9.539], [-2.627, -8.866, 12.376], [-0.225, -1.953, 6.3], [3.756, -1.482, 3.789], [-6.064, -5.289, 17.436], [-5.991, 4.627, 10.374], [-4.721, 1.921, 0.395], [7.207, -6.194, 2.688], [6.116, -2.253, 0.645], [-2.659, 5.401, 16.618], [0.295, 4.274, 10.454], [8.064, 4.843, 3.913], [-4.847, -0.224, 19.209], [7.963, 2.433, 3.14], [6.766, 4.78, 5.8], [-0.813, 7.542, 7.531], [-8.842, -8.467, 5.591], [-0.487, 6.725, 16.624], [1.122, -4.89, 18.796],

[-5.072, -2.134, 11.865], [1.988, 4.89, 9.848], [5.353, -8.124, 1.76], [7.351, 4.513, 9.799], [-1.452, 2.976, 18.906], [8.599, -1.751, 14.223], [-8.306, 5.713, 12.091], [1.303, -7.996, 16.316], [-4.034, -6.84, 15.376], [-2.337, -7.788, 6.761]

Problem instance 7: [-2.105, -8.295, 4.635], [3.758, 0.785, 18.569], [7.844, 6.008, 19.517], [6.787, 0.046, 19.447], [-4.749, -2.667, 2.273], [8.615, -4.557, 8.534], [5.521, 5.951, 3.19], [-7.103, -6.885, 15.768], [-7.445, -5.94, 18.658], [-1.068, -5.91, 8.68], [6.111, -6.369, 5.384], [-0.89, -4.376, 14.327], [-6.694, 8.458, 15.098], [1.802, -6.997, 19.668], [-7.808, 8.209, 17.34], [8.243, 0.515, 0.627], [-1.068, 0.02, 9.008], [0.428, -6.992, 17.036], [-7.826, 5.455, 14.227], [-7.557, 3.399, 4.151], [5.275, 6.138, 11.503], [1.892, -0.932, 12.834], [-1.615, -0.684, 5.835], [-3.093, -4.466, 6.496], [-8.134, -3.799, 15.109], [-1.441, 3.134, 11.053], [-7.829, -2.281, 11.29], [-7.285, 2.943, 2.679], [1.098, 2.846, 12.451], [-7.163, 0.238, 2.564], [-3.987, -6.721, 7.561], [-6.817, -6.935, 1.66], [-7.921, 2.093, 19.394], [-7.177, -2.843, 9.946], [8.025, 5.58, 9.297], [1.279, 1.027, 7.677], [-5.099, -7.356, 5.582], [-7.855, 3.011, 11.832], [6.788, 3.533, 12.035], [-3.865, 1.267, 13.452], [-7.49, 8.276, 9.623], [8.729, 7.164, 2.969], [-5.881, -1.919, 5.054], [8.094, -1.849, 1.462], [-2.728, -2.42, 13.381], [6.911, 5.327, 3.794], [3.413, -4.376, 0.614], [-0.167, -5.621, 10.23], [4.429, 7.956, 7.98], [8.064, 5.848, 11.309]

Problem instance 8: [-6.801, 5.4, 12.086], [-4.225, 8.522, 11.329], [2.439, 5.199, 18.516], [4.634, -1.86, 4.929], [2.306, -1.365, 14.808], [0.193, 2.26, 0.342], [-2.232, 1.784, 10.204], [3.203, -6.255, 1.415], [0.463, -6.995, 12.247], [-0.339, -0.303, 17.919], [-6.321, -2.303, 17.803], [-2.804, -5.08, 9.652], [-3.044, 8.761, 3.955], [-0.403, 6.258, 0.322], [1.85, 5.823, 2.191], [1.754, 3.652, 18.785], [-6.834, 8.481, 2.62], [7.077, -4.067, 4.381], [-1.77, 2.565, 14.071], [-1.407, -3.481, 13.04], [3.92, 3.979, 7.315], [2.051, -1.752, 9.32], [3.066, 5.821, 19.273], [6.803, 7.525, 19.853], [-6.495, 8.211, 0.627], [5.002, 8.653, 11.567], [2.7, 4.284, 8.756], [-6.127, -8.445, 9.424], [1.52, 3.762, 16.464], [2.542, -3.684, 5.685], [8.539, -0.981, 15.444], [1.498, -6.895, 17.785], [2.496, 4.296, 3.866], [5.432, 6.258, 5.23], [-1.226, -5.228, 0.874], [4.316, 1.29, 2.069], [2.893, -5.882,

13.46], [-6.775, 6.002, 8.279], [-5.284, 3.455, 17.283], [2.313, -4.199, 12.28], [6.501, -8.91, 7.165], [-2.611, -0.038, 4.041], [6.859, -6.106, 19.606], [-4.988, 6.607, 15.425], [-1.662, 2.299, 8.112], [3.092, 8.004, 2.301], [8.497, -7.725, 13.755], [5.113, -1.167, 6.279], [5.964, 2.205, 4.205], [4.211, 6.771, 1.349]

Problem instance 9: [-2.312, -8.743, 0.69], [-2.385, 3.094, 19.162], [-5.276, -6.993, 4.057], [1.76, -2.869, 5.559], [5.077, -6.891, 0.4], [-6.662, -3.104, 7.635], [-3.749, -5.343, 0.179], [-0.532, 2.273, 9.586], [-5.497, -8.087, 19.424], [-5.208, 8.667, 4.893], [-3.232, -3.314, 5.256], [2.615, -1.594, 8.19], [2.49, 1.496, 4.807], [3.642, 2.917, 4.96], [-7.423, -7.403, 14.013], [8.759, 3.89, 13.133], [-8.023, 7.908, 1.832], [7.297, 2.367, 15.116], [-8.492, 4.891, 0.088], [-0.005, -4.296, 1.035], [-2.092, -0.753, 1.893], [-6.737, 4.427, 19.831], [-0.4, -3.03, 15.42], [0.348, -3.901, 18.417], [3.034, -4.46, 15.359], [-2.934, -6.657, 18.732], [-2.953, -5.218, 16.716], [1.328, -3.79, 11.914], [-3.86, -4.733, 3.197], [-7.333, 8.117, 3.384], [-6.672, -7.051, 0.219], [-7.009, -1.396, 11.47], [-0.323, 7.877, 11.754], [3.497, 1.91, 10.66], [7.16, -4.961, 9.828], [-7.134, -4.168, 14.069], [-2.892, 6.713, 5.121], [3.746, 2.073, 13.972], [-1.605, -2.36, 17.73], [2.515, -6.354, 1.335], [-7.202, -5.957, 1.786], [8.046, 7.447, 16.882], [-8.384, 2.923, 2.787], [-4.225, 6.208, 11.437], [4.305, -1.345, 15.754], [-5.503, 3.864, 13.636], [4.059, -7.917, 3.785], [7.937, 2.549, 2.262], [5.435, -3.151, 5.49], [8.717, 3.839, 15.51]

Problem instance 10: [2.081, 5.597, 2.432], [-0.906, -0.53, 8.911], [5.971, 6.235, 4.89], [3.462, -3.954, 19.765], [0.173, 1.197, 11.362], [3.87, 0.365, 13.669], [-7.868, 4.704, 16.373], [2.671, 5.403, 11.378], [-3.123, -7.573, 8.271], [8.178, 3.656, 19.93], [-5.071, 7.318, 0.389], [-0.874, -3.114, 15.35], [1.369, -5.741, 7.338], [-0.826, 7.76, 7.232], [-1.705, 4.549, 18.935], [-7.933, -8.66, 16.754], [-3.281, 2.176, 13.13], [-3.384, -3.992, 4.36], [4.337, -1.271, 16.088], [-7.502, -6.664, 16.385], [-7.192, -3.084, 8.741], [2.096, -1.045, 5.719], [-4.678, 6.23, 17.658], [1.136, 1.272, 6.896], [7.755, -0.543, 6.717], [4.507, -1.399, 0.201], [1.51, 3.05, 6.454], [-1.575, -8.495, 3.289], [-4.321, -0.907, 1.998], [8.754, -5.077, 14.343], [7.608, 2.805, 7.084], [6.922,

6.011, 12.915], [-3.911, -6.862, 1.272], [3.62, -1.407, 3.344], [4.352, 0.092, 7.431], [8.179, 7.561, 1.958], [-4.536, 3.227, 9.428], [-5.245, 0.12, 3.811], [-6.31, 6.009, 4.69], [-3.482, -4.229, 8.286], [3.103, -2.261, 16.521], [-8.666, -7.364, 3.542], [2.007, -3.635, 19.301], [-4.866, -6.784, 5.814], [-1.76, 3.269, 10.979], [1.693, -5.581, 15.315], [8.221, -0.869, 16.199], [3.581, 5.833, 19.338], [6.805, 0.368, 7.279], [-7.251, 2.626, 17.83]

A.4 The 75 user scenario

Problem instance 1: [2.409, 7.383, 0.013], [9.494, 0.212, 5.11], [-9.398, -7.981, 19.153], [-4.626, -6.139, 14.729], [9.781, -9.958, 14.45], [3.008, 10.811, 15.753], [3.65, -9.634, 19.662], [-9.634, -0.472, 14.269], [-8.836, -1.414, 3.737], [-4.613, 5.913, 14.803], [-3.107, 0.144, 8.585], [-3.057, -10.912, 12.228], [-8.059, 0.736, 1.925], [0.185, -5.872, 10.052], [10.086, -4.699, 12.703], [1.08, -6.129, 1.513], [8.085, 5.419, 4.606], [6.222, -2.402, 14.304], [-10.242, 0.755, 10.943], [-10.817, -5.457, 10.304], [10.408, -9.451, 5.83], [2.039, 4.427, 0.042], [4.493, 8.593, 0.755], [-10.709, 10.115, 7.282], [1.045, 5.712, 6.955], [-6.96, 8.736, 17.293], [7.006, 3.455, 9.802], [-10.238, -4.762, 7.194], [7.526, -9.919, 18.491], [0.088, -10.382, 4.166], [-7.511, -10.93, 9.775], [-10.455, 7.605, 14.335], [-0.211, -3.508, 13.144], [-3.221, 0.105, 17.368], [-8.758, 2.266, 1.009], [1.205, -8.006, 19.711], [-5.804, 9.578, 7.03], [7.032, -8.137, 19.099], [4.014, -4.979, 8.193], [10.489, 9.648, 0.284], [4.814, -0.993, 10.279], [-0.737, -5.896, 19.315], [-3.823, -9.269, 13.917], [-7.042, -3.559, 12.623], [6.874, -1.27, 18.098], [-3.745, -9.151, 16.108], [6.068, 3.254, 18.198], [-2.639, 9.218, 12.932], [-4.385, -5.121, 1.301], [-3.456, 7.966, 12.931], [1.629, -6.449, 1.617], [5.583, 3.279, 1.716], [-7.267, -1.026, 18.109], [-4.92, 7.412, 2.69], [3.984, -1.754, 10.246], [7.302, -0.917, 4.575], [2.738, 1.414, 5.374], [-6.525, -9.809, 2.798], [7.858, 9.542, 5.496], [-2.264, -3.468, 13.783], [-9.35, -8.791, 18.704], [2.329, 3.016, 6.007], [-5.257, 10.59, 7.005], [-10.162, -3.831, 1.601], [4.63, -0.781, 7.818], [-7, 2.155, 11.707], [-3.574, -3.025, 9.733], [-4.413,

6.247, 3.81], [-10.845, 6.443, 2.828], [8.502, -2.268, 9.626], [-6.603, -2.023, 2.721], [-2.678, -3.088, 5.35], [10.824, 2.05, 14.046], [-4.414, 8.134, 14.616], [9.818, -9.004, 13.559]

Problem instance 2: [-9.641, 9.202, 16.264], [-6.783, -5.39, 1.571], [-4.269, 5.107, 16.276], [7.257, -1.958, 16.322], [0.87, 8.105, 11.183], [5.263, -3.216, 10.679], [-0.531, 6.909, 9.016], [1.311, 10.884, 13.796], [-8.544, -1.987, 5.947], [9.596, 1.232, 12.313], [-4.513, 3.343, 9.247], [-8.146, 2.082, 10.161], [5.076, -7.785, 2.67], [7.223, 5.085, 0.845], [3.656, 9.973, 18.317], [7.495, 1.697, 10.013], [-0.836, 9.003, 16.846], [-6.551, 6.857, 8.267], [10.354, 1.636, 0.845], [-9.386, 5.634, 11.699], [7.742, -0.472, 2.33], [-7.537, -4.316, 19.088], [-10.063, 8, 9.598], [5.111, 8.504, 9.552], [8.438, 7.04, 8.333], [5.765, 1.517, 19.653], [-6.888, 1.838, 9.7], [-2.213, 10.528, 0.65], [-0.079, -0.356, 16.267], [-2.822, 3.351, 1.652], [-0.561, 9.063, 6.83], [7.369, 8.198, 12.227], [3.929, 4.212, 11.02], [5.531, -4.825, 1.113], [1.045, 2.116, 9.313], [-2.731, -8.404, 11.642], [-3.613, 3.064, 0.817], [5.846, -0.369, 0.667], [5.02, -10.035, 7.482], [9.062, 9.378, 2.596], [-6.785, 4.32, 8.587], [-7.169, 1.262, 5.198], [-6.56, 9.422, 17.39], [-1.562, -9.315, 7.763], [10.064, -3.459, 13.669], [9.825, -8.906, 18.338], [3.634, 9.182, 7.284], [4.048, -5.218, 18.8], [-9.002, 3.906, 11.776], [-3.352, -2.775, 3.255], [-1.726, -2.732, 9.753], [-5.987, -8.049, 9.477], [7.21, -8.491, 17.368], [7.234, 3.782, 16.078], [0.87, -7.935, 4.173], [-9.613, -6.931, 15.493], [-2.815, -1.452, 7.789], [0.769, 9.295, 9.345], [-4.21, -1.665, 16.332], [6.639, 8.026, 3.99], [3.734, 7.571, 16.745], [-4.534, -8.891, 2.972], [0.533, 8.625, 2.667], [2.243, -10.532, 9.704], [-1.24, 6.822, 12.744], [-1.406, 0.307, 18.737], [1.06, -7.19, 12.95], [-7.736, -2.805, 0.231], [-9.872, -6.592, 19.123], [-3.598, -9.732, 3.749], [0.846, 4.612, 2.71], [-5.043, 0.679, 19.922], [-0.881, -9.866, 2.161], [4.817, -9.622, 0.188], [-3.079, 1.763, 3.338]

Problem instance 3: [-9.44, 7.659, 0.317], [-6.904, 7.132, 8.453], [4.057, -10.813, 9.813], [-1.973, 1.763, 13.425], [2.365, 10.042, 3.374], [2.327, 6.988, 3.359], [10.709, 1.58, 14.871], [-9.583, 7.619, 18.653], [-10.065, 1.605, 2.583], [-0.303, -7.82, 18.295], [-0.65, -2.707, 6.404], [4.264, 2.191, 19.06], [8.682, 6.306, 7.024], [1.308, -6.803, 18.024], [3.102, 6.131, 5.174], [0.19,

-3.852, 17.889], [-5.823, -4.346, 17.724], [-1.545, -1.952, 10.177], [-6.529, 9.53, 0.987], [3.007, 0.756, 15.58], [-4.226, -1.534, 17.143], [2.847, -4.859, 9.306], [7.994, -6.5, 18.278], [4.669, 8.859, 5.954], [-5.09, -1.59, 14.029], [-5.92, 6.885, 0.981], [-0.697, -7.64, 3.029], [7.022, -9.814, 14.799], [6.275, -9.445, 1.984], [1.331, -6.747, 2.361], [-9.037, 7.517, 13.116], [-6.484, -4.246, 2.613], [-0.502, 5.993, 6.345], [-3.096, -6.073, 7.044], [7.189, -9.864, 8.951], [-7.952, -9.608, 18.027], [-2.321, -10.627, 11.516], [7.753, 0.708, 18.171], [6.216, -1.906, 7.65], [4.761, -9.001, 7.521], [4.496, 4.24, 19.24], [10.621, 10.189, 8.083], [-2.083, 6.561, 15.402], [-8.838, -0.91, 11.301], [5.983, 10.042, 4.504], [-2.331, 4.727, 18.638], [-5.792, -2.455, 11.642], [8.418, 4.32, 17.364], [-7.777, 2.78, 1.108], [-9.619, -7.149, 13.617], [4.911, 4.092, 17.737], [-3.361, -10.487, 12.164], [7.382, -10.677, 2.338], [10.78, 4.49, 7.92], [-2.201, -7.809, 7.936], [-7.648, 9.901, 11.292], [9.961, -4.736, 10.95], [7.849, 7.203, 14.411], [10.269, -6.915, 13.165], [6.733, 9.533, 5.995], [7.078, -6.581, 19.517], [-4.42, -7.505, 9.469], [-4.334, -10.061, 9.408], [-10.235, -4.741, 2.824], [9.969, -7.457, 14.893], [-4.553, 0.623, 11.666], [-2.942, -0.263, 19.139], [-2.428, 7.746, 15.701], [-3.21, -10.169, 8.455], [9.151, 7.058, 14.294], [9.59, 9.138, 7.301], [-4.056, -0.334, 0.043], [-5.87, 4.037, 12.003], [3.967, -7.09, 13.447], [6.811, 2.733, 18.3]

Problem instance 4: [5.656, -10.603, 4.379], [-2.782, -0.962, 0.317], [3.189, 10.73, 5.042], [0.799, 1.904, 13.085], [10.511, -10.171, 1.632], [5.498, -1.292, 3.663], [1.438, -5.972, 0.488], [7.982, 7.079, 12.139], [4.871, -7.951, 18.476], [2.482, -7.143, 13.341], [2.413, 5.708, 16.015], [-0.025, -2.919, 4.304], [-3.961, 9.562, 0.392], [-5.002, 9.668, 0.617], [-8.057, 6.802, 18.274], [5.338, -5.944, 17.171], [8.989, 10.87, 7.79], [-10.428, 8.422, 11.239], [-6.712, 3.215, 18.628], [-4.89, -9.572, 8.739], [-8.63, 5.128, 3.949], [5.579, 6.19, 8.156], [-5.31, 2.8, 4.196], [-1.808, -4.33, 4.303], [-0.918, 2.687, 2.976], [3.739, -10.449, 14.047], [2.466, 10.171, 7.218], [6.287, 2.348, 14.707], [2.567, -3.981, 11.635], [1.03, -9.413, 12.252], [5.667, -0.643, 16.254], [8.529, -2.307, 15.11], [-10.084, -2.025, 2.702], [-7.639, -2.093, 2.029], [-2.3, 8.223, 19.324], [-1.208, -4.033, 11.953], [8.799, -6.5, 7.675], [-2.895, -1.962, 12.774], [3.013, -7.381, 7.67], [1.083, -3.526, 0.047], [-10.384, 5.696, 10.272], [-8.69, -8.044, 7.34], [-3.893, 10.651, 16.591], [-2.398,

1.216, 15.162], [1.981, 0.966, 10.452], [-4.704, -4.144, 9.934], [-6.999, -5.692, 10.825], [6.985, 2.786, 0.946], [8.96, -6.738, 10.487], [1.756, -9.501, 8.439], [-6.179, -4.864, 18.14], [-8.948, -1.33, 0.171], [-5.179, -5.704, 9.412], [-0.849, -5.875, 7.455], [1.723, -8.272, 17.721], [-3.239, 8.502, 18.667], [-7.53, -8.279, 15.757], [4.864, 1.996, 16.902], [-8.199, 0.849, 10.426], [1.516, 7.979, 18.491], [-4.681, 3.7, 18.265], [-3.889, 9.82, 18.962], [-6.519, 9.751, 18.402], [9.471, 6.126, 5.144], [2.671, 2.26, 17.681], [-9.554, 7.677, 3.011], [-4.355, 3.634, 6.328], [-9.852, -5.496, 8.086], [-2.174, -8.322, 2.225], [9.645, 10.387, 10.388], [2.115, -0.901, 0.032], [9.055, -2.027, 1.876], [4.032, 2.156, 6.825], [-3.782, -5.716, 18.546], [6.242, 7.995, 19.986]

Problem instance 5: [-5.737, -4.861, 5.265], [7.71, -0.536, 16.87], [4.1, -2.134, 15.504], [-5.165, -5.257, 2.493], [7.599, -8.152, 9.111], [-5.467, 9.622, 7.032], [4.834, 7.659, 0.603], [8.882, -3.944, 11.039], [-7.854, -3.572, 2.82], [5.746, 3.329, 5.066], [-2.419, 6.864, 13.456], [-3.617, -6.166, 0.458], [-7.115, -10.946, 9.832], [-6.291, 8.878, 12.081], [-0.194, -7.469, 13.064], [7.969, -2.981, 17.17], [-6.233, 7.25, 6.903], [4.583, 9.424, 2.404], [7.661, 10.731, 9.041], [8.417, 3.61, 6.417], [0.064, -0.404, 18.423], [-7.64, 3.655, 15.584], [-0.698, 4.998, 10.464], [6.663, 2.521, 5.727], [8.452, -6.072, 7.859], [-1.337, -5.208, 11.065], [-6.619, 9.115, 12.474], [0.586, -2.201, 2.861], [-3.334, 10.218, 2.661], [5.9, 7.457, 11.968], [2.321, -5.813, 8.615], [-7.641, 8.776, 10.234], [-2.281, -8.141, 12.609], [3.38, 10.856, 4.975], [-2.996, -8.147, 5.043], [1.042, -0.395, 1.589], [3.304, -10.926, 7.352], [5.435, 1.085, 1.715], [-1.671, 1.684, 7.809], [-6.978, 1.984, 10.048], [8.045, -5.282, 5.129], [8.037, 10.589, 5.473], [-0.664, 5.548, 2.858], [4.244, -10.219, 19.529], [-7.208, 3.919, 2.279], [-3.114, -9.132, 6.143], [10.433, 2.215, 11.395], [8.671, 9.331, 0.966], [-7.66, 8.262, 1.138], [-9.273, 5.202, 5.406], [-7.633, -6.186, 6.53], [-7.133, 1.027, 9.624], [1.935, -10.133, 14.044], [1.945, 8.85, 5.136], [9.028, -10.397, 19.5], [-2.467, 9.423, 5.043], [-2.235, -1.29, 14.43], [-7.553, -9.93, 0.114], [-7.506, -10.399, 19.731], [-6.94, 3.244, 15.603], [6.289, 4.09, 7.427], [-9.168, 2.064, 11.679], [1.976, -6.51, 5.239], [4.688, 2.7, 19.531], [-0.884, 1.148, 16.939], [7.853, 2.056, 19.757], [7.458, -5.341, 3.909], [5.916, -5.196, 3.145], [0.605, 3.423, 2.629], [6.5, 5.032, 6.771], [5.974, 4.868, 16.298], [-4.026, 6.285, 7.461], [0.86, 3.335,

15.771], [-5.224, -1.088, 9.397], [-7.321, 6.17, 13.917]

Problem instance 6: [-7.571, -9.481, 1.966], [3.646, 5.616, 4.234], [7.238, 9.07, 12.872], [-1.674, -7.435, 19.712], [8.846, 3.896, 14.972], [-10.467, -4.489, 5.411], [-9.263, 10.683, 16.055], [1.141, 3.406, 14.181], [-3.385, -9.765, 9.328], [2.415, 4.281, 2.772], [9.543, 8.315, 10.97], [-10.612, 8.877, 3.239], [8.507, 8.689, 19.512], [-3.777, -1.028, 9.299], [4.117, 10.326, 3.182], [-5.409, -4.28, 13.902], [8.957, 1.127, 10.818], [10.146, -4.939, 5.917], [1.197, 10.744, 6.802], [-2.771, 9.861, 4.677], [5.776, -10.628, 16.245], [-10.468, 9.097, 9.539], [-3.217, -10.858, 12.376], [-0.276, -2.392, 6.3], [4.6, -1.815, 3.789], [-7.427, -6.477, 17.436], [-7.338, 5.667, 10.374], [-5.782, 2.352, 0.395], [8.827, -7.586, 2.688], [7.49, -2.76, 0.645], [-3.257, 6.615, 16.618], [0.362, 5.235, 10.454], [9.877, 5.932, 3.913], [-5.936, -0.274, 19.209], [9.753, 2.98, 3.14], [8.287, 5.855, 5.8], [-0.996, 9.237, 7.531], [-10.829, -10.37, 5.591], [-0.596, 8.237, 16.624], [1.374, -5.988, 18.796], [-6.212, -2.613, 11.865], [2.435, 5.989, 9.848], [6.556, -9.95, 1.76], [9.003, 5.528, 9.799], [-1.778, 3.645, 18.906], [10.532, -2.145, 14.223], [-10.172, 6.997, 12.091], [1.596, -9.794, 16.316], [-4.941, -8.378, 15.376], [-2.863, -9.538, 6.761], [7.32, 6.879, 4.483], [5.986, -2.101, 1.412], [-6.611, 2.443, 6.294], [-6.912, -4.988, 6.063], [-6.021, -6.008, 8.031], [9.974, 7.423, 4.707], [-8.55, 8.152, 2.159], [0.377, 3.645, 6.168], [5.049, -9.662, 9.593], [-2.819, 0.943, 9.414], [-10.194, 3.576, 17.297], [-5.9, -6.149, 3.429], [-1.393, 4.731, 3.581], [2.376, 4.708, 4.192], [-7.874, -6.666, 5.409], [7.974, 0.72, 3.246], [-1.344, 2.617, 3.888], [5.68, 6.059, 19.334], [3.389, -9.137, 14.939], [-9.478, 2.889, 2.889], [2.966, -9.243, 5.971], [-5.476, -7.507, 3.701], [9.49, 3.25, 2.687], [1.952, 5.119, 17.596], [-6.722, 2.962, 17.955]

Problem instance 7: [-2.578, -10.16, 4.635], [4.602, 0.962, 18.569], [9.607, 7.358, 19.517], [8.312, 0.056, 19.447], [-5.816, -3.266, 2.273], [10.551, -5.581, 8.534], [6.762, 7.289, 3.19], [-8.7, -8.433, 15.768], [-9.119, -7.275, 18.658], [-1.308, -7.238, 8.68], [7.485, -7.801, 5.384], [-1.09, -5.359, 14.327], [-8.199, 10.359, 15.098], [2.207, -8.569, 19.668], [-9.563, 10.054, 17.34], [10.095, 0.631, 0.627], [-1.308, 0.024, 9.008], [0.524, -8.563, 17.036], [-9.584, 6.681,

14.227], [-9.255, 4.163, 4.151], [6.461, 7.518, 11.503], [2.317, -1.141, 12.834], [-1.977, -0.837, 5.835], [-3.789, -5.47, 6.496], [-9.962, -4.652, 15.109], [-1.764, 3.839, 11.053], [-9.589, -2.794, 11.29], [-8.923, 3.605, 2.679], [1.345, 3.485, 12.451], [-8.773, 0.291, 2.564], [-4.884, -8.232, 7.561], [-8.349, -8.493, 1.66], [-9.701, 2.563, 19.394], [-8.79, -3.482, 9.946], [9.828, 6.834, 9.297], [1.567, 1.258, 7.677], [-6.245, -9.009, 5.582], [-9.62, 3.688, 11.832], [8.314, 4.327, 12.035], [-4.734, 1.551, 13.452], [-9.173, 10.136, 9.623], [10.691, 8.774, 2.969], [-7.203, -2.35, 5.054], [9.913, -2.265, 1.462], [-3.342, -2.964, 13.381], [8.464, 6.524, 3.794], [4.18, -5.36, 0.614], [-0.205, -6.884, 10.23], [5.425, 9.744, 7.98], [9.876, 7.162, 11.309], [2.151, 9.502, 8.327], [10.529, 1.616, 11.534], [7.425, 10.787, 5.395], [-7.867, -7.733, 10.293], [-9.059, -0.131, 0.603], [-0.193, 3.977, 16.168], [-10.545, -10.349, 15.523], [-1.941, -8.453, 19.42], [-7.365, 0.581, 3.849], [-3.517, -9.982, 19.926], [-6.891, -9.577, 17.053], [7.862, -7.156, 1.379], [4.373, -7.414, 2.459], [-10.029, -10.87, 16.344], [-5.797, 4.487, 11.138], [2.845, -6.826, 3.575], [-10.092, 6.825, 19.488], [10.7, 6.449, 10.205], [-4.97, -4.514, 14.014], [-5.611, 3.12, 10.237], [4.002, 1.558, 6.112], [3.418, 0.275, 18.528], [-2.914, -8.363, 19.951], [1.322, 8.94, 6.689], [-3.077, -7.229, 5.632]

Problem instance 8: [-8.329, 6.613, 12.086], [-5.174, 10.437, 11.329], [2.988, 6.367, 18.516], [5.676, -2.278, 4.929], [2.825, -1.671, 14.808], [0.236, 2.768, 0.342], [-2.733, 2.185, 10.204], [3.923, -7.661, 1.415], [0.567, -8.567, 12.247], [-0.415, -0.371, 17.919], [-7.742, -2.82, 17.803], [-3.434, -6.221, 9.652], [-3.728, 10.731, 3.955], [-0.493, 7.664, 0.322], [2.266, 7.131, 2.191], [2.148, 4.473, 18.785], [-8.37, 10.388, 2.62], [8.667, -4.981, 4.381], [-2.167, 3.141, 14.071], [-1.723, -4.264, 13.04], [4.801, 4.874, 7.315], [2.511, -2.146, 9.32], [3.755, 7.129, 19.273], [8.332, 9.217, 19.853], [-7.955, 10.056, 0.627], [6.127, 10.597, 11.567], [3.307, 5.247, 8.756], [-7.504, -10.343, 9.424], [1.862, 4.608, 16.464], [3.114, -4.512, 5.685], [10.458, -1.201, 15.444], [1.835, -8.445, 17.785], [3.058, 5.261, 3.866], [6.652, 7.664, 5.23], [-1.501, -6.403, 0.874], [5.287, 1.58, 2.069], [3.543, -7.204, 13.46], [-8.298, 7.351, 8.279], [-6.472, 4.232, 17.283], [2.832, -5.143, 12.28], [7.963, -10.913, 7.165], [-3.197, -0.047, 4.041], [8.4, -7.478, 19.606], [-6.109, 8.092, 15.425], [-2.036, 2.815, 8.112], [3.787, 9.802, 2.301], [10.406, -9.461, 13.755],

[6.262, -1.429, 6.279], [7.305, 2.7, 4.205], [5.158, 8.293, 1.349], [-1.618, 8.042, 15.003], [6.031, -0.426, 7.103], [5.4, -6.188, 17.926], [10.154, 2.544, 16.498], [6.936, -0.723, 9.684], [-1.824, -1.635, 8.728], [7.825, -10.78, 10.581], [-3.24, -7.07, 17.145], [1.045, 4.322, 0.073], [-4.116, -9.581, 6.492], [7.907, 3.121, 18.383], [4.254, -6.241, 2.801], [-0.986, 6.602, 10.501], [-0.988, 6.25, 15.083], [5.908, -7.644, 3.309], [0.544, 9.938, 9.972], [5.276, -8.303, 3.477], [1.49, -0.819, 8.82], [-1.279, 9.556, 16.004], [5.898, -6.787, 7.504], [-2.014, -7.761, 11.899], [-1.414, -9.013, 15.753], [-0.098, 1.262, 15.077], [8.187, -7.61, 4.088], [-9.793, -6.605, 11.06]

Problem instance 9: [-2.831, -10.708, 0.69], [-2.921, 3.789, 19.162], [-6.462, -8.565, 4.057], [2.155, -3.514, 5.559], [6.218, -8.44, 0.4], [-8.159, -3.802, 7.635], [-4.591, -6.543, 0.179], [-0.652, 2.784, 9.586], [-6.732, -9.905, 19.424], [-6.378, 10.614, 4.893], [-3.958, -4.059, 5.256], [3.202, -1.952, 8.19], [3.05, 1.832, 4.807], [4.461, 3.573, 4.96], [-9.092, -9.067, 14.013], [10.727, 4.764, 13.133], [-9.826, 9.685, 1.832], [8.937, 2.899, 15.116], [-10.401, 5.991, 0.088], [-0.006, -5.262, 1.035], [-2.562, -0.922, 1.893], [-8.251, 5.422, 19.831], [-0.49, -3.711, 15.42], [0.427, -4.777, 18.417], [3.716, -5.462, 15.359], [-3.593, -8.153, 18.732], [-3.617, -6.391, 16.716], [1.626, -4.642, 11.914], [-4.727, -5.797, 3.197], [-8.981, 9.941, 3.384], [-8.171, -8.635, 0.219], [-8.585, -1.71, 11.47], [-0.396, 9.647, 11.754], [4.283, 2.339, 10.66], [8.769, -6.076, 9.828], [-8.737, -5.105, 14.069], [-3.542, 8.222, 5.121], [4.588, 2.539, 13.972], [-1.966, -2.89, 17.73], [3.08, -7.782, 1.335], [-8.821, -7.296, 1.786], [9.854, 9.121, 16.882], [-10.268, 3.58, 2.787], [-5.174, 7.604, 11.437], [5.272, -1.647, 15.754], [-6.74, 4.732, 13.636], [4.971, -9.696, 3.785], [9.721, 3.122, 2.262], [6.656, -3.859, 5.49], [10.676, 4.702, 15.51], [-5.885, -6.215, 16.576], [-3.659, -4.238, 7.399], [9.512, -6.699, 3.743], [3.862, -9.461, 7.916], [-2.88, -6.819, 7.91], [-8.546, -5.589, 16.422], [2.775, 3.504, 19.897], [-0.487, 7.267, 6.979], [10.838, 8.543, 13.986], [6.757, 4.284, 1.499], [-0.969, -3.538, 5.547], [5.502, -10.886, 3.049], [-1.429, 7.63, 19.706], [-5.743, 10.469, 17.996], [7.823, -6.723, 5.043], [5.688, -5.954, 7.228], [-6.215, 7.186, 4.438], [1.148, -2.896, 1.035], [-7.042, -1.44, 14.363], [10.082, -8.943, 5.555], [-1.995, -5.61, 18.936], [0.901, 5.897,

14.24], [5.402, -5.122, 11.418], [-5.823, 10.663, 6.884], [-10.22, -0.265, 13.319]

Problem instance 10: [2.549, 6.855, 2.432], [-1.11, -0.649, 8.911], [7.313, 7.636, 4.89], [4.24, -4.843, 19.765], [0.212, 1.466, 11.362], [4.74, 0.447, 13.669], [-9.636, 5.761, 16.373], [3.271, 6.617, 11.378], [-3.825, -9.275, 8.271], [10.016, 4.477, 19.93], [-6.211, 8.962, 0.389], [-1.07, -3.814, 15.35], [1.676, -7.031, 7.338], [-1.011, 9.504, 7.232], [-2.089, 5.571, 18.935], [-9.715, -10.606, 16.754], [-4.018, 2.664, 13.13], [-4.145, -4.889, 4.36], [5.311, -1.556, 16.088], [-9.189, -8.162, 16.385], [-8.808, -3.777, 8.741], [2.567, -1.28, 5.719], [-5.729, 7.63, 17.658], [1.391, 1.558, 6.896], [9.498, -0.665, 6.717], [5.52, -1.713, 0.201], [1.849, 3.735, 6.454], [-1.929, -10.404, 3.289], [-5.293, -1.111, 1.998], [10.721, -6.218, 14.343], [9.318, 3.436, 7.084], [8.478, 7.362, 12.915], [-4.79, -8.404, 1.272], [4.434, -1.723, 3.344], [5.33, 0.113, 7.431], [10.017, 9.26, 1.958], [-5.556, 3.952, 9.428], [-6.424, 0.147, 3.811], [-7.728, 7.359, 4.69], [-4.264, -5.179, 8.286], [3.801, -2.769, 16.521], [-10.613, -9.019, 3.542], [2.459, -4.452, 19.301], [-5.96, -8.309, 5.814], [-2.156, 4.003, 10.979], [2.074, -6.835, 15.315], [10.069, -1.065, 16.199], [4.386, 7.144, 19.338], [8.334, 0.45, 7.279], [-8.881, 3.217, 17.83], [8.04, 1.02, 2.509], [4.185, 1.749, 0.858], [-5.033, -5.806, 16.202], [-8.254, 3.797, 15.356], [5.224, 0.512, 11.41], [1.746, 0.133, 12.522], [4.267, 8.045, 4.648], [-9.781, -3.176, 2.806], [-7.465, -5.979, 4.737], [-7.934, 2.437, 3.119], [-5.556, 9.26, 18.221], [3.847, -3.524, 3.385], [1.376, -10.013, 6.345], [-2.86, -8.35, 12.14], [7.576, 4.928, 14.506], [7.705, -9.483, 0.805], [-8.82, -2.28, 8.057], [6.597, -1.586, 10.831], [6.074, 3.33, 16.897], [9.962, 6.427, 14.092], [-0.82, 9.129, 9.031], [0.601, -0.554, 3.754], [5.747, 0.879, 18.87], [4.89, -4.714, 17.664], [-4.561, -10.21, 0.424]

A.5 The 100 user scenario

Problem instance 1: [2.782, 8.525, 0.013], [10.963, 0.245, 5.11], [-10.852, -9.216, 19.153], [-5.341, -7.088, 14.729], [11.294, -11.499, 14.45], [3.473, 12.484, 15.753], [4.214, -11.125, 19.662], [-11.124, -0.546, 14.269], [-10.203, -1.633, 3.737], [-5.327, 6.828, 14.803], [-3.587, 0.166, 8.585], [-3.53, -12.6, 12.228], [-9.305, 0.85, 1.925], [0.213, -6.78, 10.052], [11.646, -5.426, 12.703], [1.247, -7.077, 1.513], [9.336, 6.257, 4.606], [7.184, -2.774, 14.304], [-11.826, 0.871, 10.943], [-12.49, -6.301, 10.304], [12.018, -10.913, 5.83], [2.354, 5.112, 0.042], [5.188, 9.922, 0.755], [-12.366, 11.68, 7.282], [1.206, 6.596, 6.955], [-8.037, 10.087, 17.293], [8.09, 3.989, 9.802], [-11.821, -5.498, 7.194], [8.69, -11.454, 18.491], [0.101, -11.988, 4.166], [-8.673, -12.621, 9.775], [-12.073, 8.782, 14.335], [-0.243, -4.051, 13.144], [-3.719, 0.122, 17.368], [-10.113, 2.616, 1.009], [1.392, -9.245, 19.711], [-6.702, 11.06, 7.03], [8.12, -9.396, 19.099], [4.634, -5.749, 8.193], [12.112, 11.141, 0.284], [5.559, -1.146, 10.279], [-0.851, -6.809, 19.315], [-4.415, -10.703, 13.917], [-8.131, -4.11, 12.623], [7.937, -1.466, 18.098], [-4.325, -10.566, 16.108], [7.007, 3.758, 18.198], [-3.047, 10.644, 12.932], [-5.063, -5.913, 1.301], [-3.991, 9.198, 12.931], [1.881, -7.447, 1.617], [6.446, 3.786, 1.716], [-8.391, -1.185, 18.109], [-5.681, 8.558, 2.69], [4.6, -2.025, 10.246], [8.431, -1.059, 4.575], [3.161, 1.632, 5.374], [-7.535, -11.327, 2.798], [9.074, 11.018, 5.496], [-2.614, -4.005, 13.783], [-10.796, -10.151, 18.704], [2.69, 3.482, 6.007], [-6.07, 12.228, 7.005], [-11.734, -4.423, 1.601], [5.346, -0.902, 7.818], [-8.083, 2.489, 11.707], [-4.127, -3.492, 9.733], [-5.096, 7.213, 3.81], [-12.522, 7.439, 2.828], [9.818, -2.619, 9.626], [-7.625, -2.335, 2.721], [-3.092, -3.566, 5.35], [12.499, 2.368, 14.046], [-5.096, 9.393, 14.616], [11.337, -10.397, 13.559], [-11.523, 4.226, 19.723], [-6.444, -0.43, 11.9], [-7.277, 4.04, 11.29], [4.563, 10.959, 7.164], [11.062, 8.528, 16.289], [-1.735, -4.377, 12.526], [4.404, -10.148, 10.574], [-10.046, -9.39, 5.59], [-7.631, 4.489, 4.192], [3.708, 7.52, 17.629], [1.38, -5.396, 15.95], [10.635, 7.929, 18.686], [-0.451, 8.246, 10.804], [-3.663, -10.744, 17.785], [-10.736, -6.084, 6.505], [-10.474, -8.759, 12.94], [8.074, -10.465, 12.492], [-2.148, -4.022, 11.606], [-12.209, -7.332, 9.002], [-2.403,

9.46, 11.23], [-0.875, -4.152, 0.947], [-11.984, 9.057, 19.401], [4.57, 8.494, 1.795], [-11.034, -1.287, 13.446], [2.39, 6.663, 0.478]

Problem instance 2: [-11.132, 10.626, 16.264], [-7.832, -6.224, 1.571], [-4.929, 5.897, 16.276], [8.379, -2.261, 16.322], [1.005, 9.359, 11.183], [6.077, -3.713, 10.679], [-0.613, 7.977, 9.016], [1.514, 12.568, 13.796], [-9.866, -2.294, 5.947], [11.08, 1.423, 12.313], [-5.211, 3.86, 9.247], [-9.406, 2.404, 10.161], [5.861, -8.989, 2.67], [8.341, 5.872, 0.845], [4.222, 11.515, 18.317], [8.654, 1.96, 10.013], [-0.966, 10.396, 16.846], [-7.564, 7.918, 8.267], [11.956, 1.889, 0.845], [-10.838, 6.505, 11.699], [8.939, -0.545, 2.33], [-8.703, -4.984, 19.088], [-11.62, 9.238, 9.598], [5.901, 9.819, 9.552], [9.744, 8.129, 8.333], [6.656, 1.752, 19.653], [-7.953, 2.122, 9.7], [-2.555, 12.156, 0.65], [-0.091, -0.411, 16.267], [-3.258, 3.869, 1.652], [-0.647, 10.465, 6.83], [8.509, 9.466, 12.227], [4.537, 4.864, 11.02], [6.386, -5.571, 1.113], [1.206, 2.444, 9.313], [-3.153, -9.704, 11.642], [-4.172, 3.538, 0.817], [6.751, -0.426, 0.667], [5.796, -11.587, 7.482], [10.464, 10.828, 2.596], [-7.834, 4.988, 8.587], [-8.278, 1.457, 5.198], [-7.575, 10.879, 17.39], [-1.803, -10.756, 7.763], [11.621, -3.995, 13.669], [11.345, -10.284, 18.338], [4.197, 10.603, 7.284], [4.674, -6.025, 18.8], [-10.394, 4.511, 11.776], [-3.87, -3.205, 3.255], [-1.994, -3.155, 9.753], [-6.914, -9.294, 9.477], [8.326, -9.805, 17.368], [8.354, 4.367, 16.078], [1.005, -9.163, 4.173], [-11.1, -8.003, 15.493], [-3.25, -1.677, 7.789], [0.888, 10.733, 9.345], [-4.861, -1.923, 16.332], [7.666, 9.268, 3.99], [4.312, 8.742, 16.745], [-5.235, -10.267, 2.972], [0.615, 9.96, 2.667], [2.59, -12.161, 9.704], [-1.432, 7.878, 12.744], [-1.624, 0.354, 18.737], [1.224, -8.302, 12.95], [-8.933, -3.239, 0.231], [-11.399, -7.612, 19.123], [-4.154, -11.238, 3.749], [0.977, 5.326, 2.71], [-5.823, 0.784, 19.922], [-1.017, -11.393, 2.161], [5.562, -11.111, 0.188], [-3.555, 2.036, 3.338], [4.472, 3.227, 6.197], [-5.782, -1.133, 1.828], [-6.395, 11.879, 17.989], [-10.381, -4.949, 2.727], [11.79, -3.698, 14.912], [-6, -6.204, 18.426], [5.359, -6.5, 18.088], [-0.62, -1.596, 12.274], [-3.742, -10.723, 1.463], [-7.528, 8.469, 17.752], [8.348, 6.723, 14.086], [-1.616, 10.714, 17.943], [-9.213, -6.601, 14.839], [8.022, 9.242, 2.049], [-1.624, -4.371, 5.718], [-7.869, 11.68, 0.854], [-1.113, -0.374, 12.465], [-7.829, -12.041, 14.466], [-10.681, 5.242, 0.254], [0.098, -11.909, 10.529], [2.35,

-11.513, 5.22], [-11.424, 2.608, 5.602], [7.039, -6.477, 4.373], [-3.424, -8.901, 8.182], [7.752, -5.669, 13.997]

Problem instance 3: [-10.901, 8.844, 0.317], [-7.972, 8.235, 8.453], [4.685, -12.485, 9.813], [-2.278, 2.036, 13.425], [2.731, 11.595, 3.374], [2.687, 8.069, 3.359], [12.365, 1.825, 14.871], [-11.066, 8.798, 18.653], [-11.622, 1.853, 2.583], [-0.35, -9.029, 18.295], [-0.751, -3.126, 6.404], [4.923, 2.53, 19.06], [10.025, 7.281, 7.024], [1.51, -7.856, 18.024], [3.582, 7.08, 5.174], [0.22, -4.448, 17.889], [-6.724, -5.019, 17.724], [-1.784, -2.254, 10.177], [-7.539, 11.005, 0.987], [3.472, 0.873, 15.58], [-4.879, -1.771, 17.143], [3.288, -5.61, 9.306], [9.23, -7.505, 18.278], [5.392, 10.229, 5.954], [-5.877, -1.836, 14.029], [-6.836, 7.95, 0.981], [-0.804, -8.822, 3.029], [8.108, -11.332, 14.799], [7.246, -10.906, 1.984], [1.537, -7.791, 2.361], [-10.435, 8.68, 13.116], [-7.487, -4.903, 2.613], [-0.579, 6.921, 6.345], [-3.575, -7.012, 7.044], [8.302, -11.391, 8.951], [-9.182, -11.095, 18.027], [-2.68, -12.271, 11.516], [8.953, 0.818, 18.171], [7.177, -2.201, 7.65], [5.497, -10.394, 7.521], [5.192, 4.896, 19.24], [12.265, 11.766, 8.083], [-2.406, 7.577, 15.402], [-10.205, -1.05, 11.301], [6.909, 11.596, 4.504], [-2.691, 5.459, 18.638], [-6.688, -2.835, 11.642], [9.72, 4.989, 17.364], [-8.98, 3.21, 1.108], [-11.107, -8.255, 13.617], [5.671, 4.725, 17.737], [-3.881, -12.109, 12.164], [8.524, -12.329, 2.338], [12.447, 5.184, 7.92], [-2.541, -9.017, 7.936], [-8.831, 11.433, 11.292], [11.502, -5.469, 10.95], [9.063, 8.317, 14.411], [11.857, -7.985, 13.165], [7.775, 11.007, 5.995], [8.173, -7.6, 19.517], [-5.103, -8.665, 9.469], [-5.004, -11.618, 9.408], [-11.818, -5.474, 2.824], [11.511, -8.61, 14.893], [-5.258, 0.719, 11.666], [-3.398, -0.304, 19.139], [-2.803, 8.945, 15.701], [-3.706, -11.742, 8.455], [10.566, 8.15, 14.294], [11.073, 10.551, 7.301], [-4.684, -0.386, 0.043], [-6.778, 4.662, 12.003], [4.581, -8.186, 13.447], [7.864, 3.156, 18.3], [3.497, -2.597, 19.942], [5.309, -3.068, 15.228], [2.296, -1.663, 5.88], [9.222, 6.708, 10.96], [0.954, 5.55, 15.77], [3.004, 3.501, 8.683], [2.595, 2.674, 16.282], [7.246, 10.073, 12.729], [8.029, 8.856, 1.864], [-7.895, -6.265, 19.174], [-5.946, 3.539, 6.936], [4.919, -10.06, 2.748], [10.076, -4.561, 6.573], [-1.135, 1.081, 15.319], [6.041, 2.016, 4.473], [-2.869, 11.989, 12.467], [-0.231, 4.221, 9.634], [3.573, 9.152, 16.751], [-10.255, -12.438, 3.86], [-7.115, -1.038, 13.289], [5.125,

-1.705, 16.342], [0.426, 8.075, 0.354], [8.864, 9.627, 4.414], [-7.782, -4.568, 19.771], [-10.278, 5.158, 6.543]

Problem instance 4: [6.53, -12.243, 4.379], [-3.212, -1.111, 0.317], [3.683, 12.39, 5.042], [0.923, 2.198, 13.085], [12.137, -11.745, 1.632], [6.349, -1.491, 3.663], [1.66, -6.896, 0.488], [9.216, 8.174, 12.139], [5.624, -9.181, 18.476], [2.866, -8.247, 13.341], [2.786, 6.591, 16.015], [-0.029, -3.371, 4.304], [-4.573, 11.042, 0.392], [-5.776, 11.163, 0.617], [-9.304, 7.854, 18.274], [6.164, -6.864, 17.171], [10.38, 12.552, 7.79], [-12.041, 9.725, 11.239], [-7.75, 3.712, 18.628], [-5.647, -11.053, 8.739], [-9.965, 5.922, 3.949], [6.442, 7.147, 8.156], [-6.132, 3.234, 4.196], [-2.088, -4.999, 4.303], [-1.06, 3.102, 2.976], [4.317, -12.066, 14.047], [2.847, 11.744, 7.218], [7.259, 2.711, 14.707], [2.964, -4.597, 11.635], [1.189, -10.869, 12.252], [6.543, -0.742, 16.254], [9.849, -2.663, 15.11], [-11.644, -2.338, 2.702], [-8.82, -2.416, 2.029], [-2.655, 9.495, 19.324], [-1.395, -4.657, 11.953], [10.161, -7.505, 7.675], [-3.342, -2.266, 12.774], [3.479, -8.523, 7.67], [1.251, -4.072, 0.047], [-11.99, 6.578, 10.272], [-10.034, -9.289, 7.34], [-4.495, 12.299, 16.591], [-2.768, 1.405, 15.162], [2.287, 1.115, 10.452], [-5.431, -4.785, 9.934], [-8.082, -6.573, 10.825], [8.065, 3.217, 0.946], [10.347, -7.78, 10.487], [2.028, -10.971, 8.439], [-7.135, -5.616, 18.14], [-10.332, -1.536, 0.171], [-5.98, -6.586, 9.412], [-0.98, -6.784, 7.455], [1.99, -9.551, 17.721], [-3.74, 9.817, 18.667], [-8.695, -9.56, 15.757], [5.616, 2.304, 16.902], [-9.467, 0.98, 10.426], [1.751, 9.213, 18.491], [-5.405, 4.272, 18.265], [-4.49, 11.34, 18.962], [-7.527, 11.259, 18.402], [10.936, 7.074, 5.144], [3.084, 2.609, 17.681], [-11.032, 8.865, 3.011], [-5.029, 4.197, 6.328], [-11.376, -6.347, 8.086], [-2.51, -9.609, 2.225], [11.138, 11.994, 10.388], [2.442, -1.04, 0.032], [10.456, -2.341, 1.876], [4.656, 2.49, 6.825], [-4.367, -6.601, 18.546], [7.208, 9.232, 19.986], [6.147, -0.246, 4.465], [-1.372, -9.73, 2.666], [3.248, -12.039, 1.16], [-8.337, 11.534, 18.134], [4.702, -10.213, 9.275], [-12.037, -0.543, 7.445], [7.867, 9.689, 19.933], [-2.154, -8.01, 17.152], [8.393, -0.435, 4.884], [-6.337, 11.552, 1.779], [8.704, -4.951, 8.651], [-7.487, -0.082, 15.009], [3.794, 11.801, 17.143], [-5.533, -10.183, 3.831], [-10.467, -9.185, 18.579], [2.596, 6.62, 6.828], [-6.037, -11.141, 14.486], [6.883, -2.365, 11.66], [5.348, -7.574, 12.913], [5.862, 4.73, 4.626],

[-2.51, 11.826, 19.164], [4.773, 1.814, 11.234], [-10.726, 11.378, 1.256], [10.663, 2.827, 9.337], [4.888, -6.323, 3.086]

Problem instance 5: [-6.624, -5.613, 5.265], [8.902, -0.618, 16.87], [4.734, -2.465, 15.504], [-5.963, -6.07, 2.493], [8.775, -9.414, 9.111], [-6.313, 11.111, 7.032], [5.582, 8.843, 0.603], [10.256, -4.555, 11.039], [-9.069, -4.125, 2.82], [6.635, 3.843, 5.066], [-2.793, 7.926, 13.456], [-4.176, -7.119, 0.458], [-8.216, -12.639, 9.832], [-7.264, 10.251, 12.081], [-0.224, -8.625, 13.064], [9.202, -3.442, 17.17], [-7.197, 8.371, 6.903], [5.292, 10.882, 2.404], [8.847, 12.391, 9.041], [9.719, 4.168, 6.417], [0.074, -0.467, 18.423], [-8.822, 4.22, 15.584], [-0.806, 5.771, 10.464], [7.694, 2.911, 5.727], [9.76, -7.011, 7.859], [-1.544, -6.014, 11.065], [-7.643, 10.525, 12.474], [0.676, -2.542, 2.861], [-3.849, 11.799, 2.661], [6.813, 8.611, 11.968], [2.68, -6.713, 8.615], [-8.824, 10.134, 10.234], [-2.634, -9.4, 12.609], [3.903, 12.536, 4.975], [-3.459, -9.407, 5.043], [1.203, -0.456, 1.589], [3.815, -12.616, 7.352], [6.275, 1.253, 1.715], [-1.93, 1.945, 7.809], [-8.057, 2.291, 10.048], [9.289, -6.099, 5.129], [9.281, 12.227, 5.473], [-0.767, 6.406, 2.858], [4.901, -11.8, 19.529], [-8.323, 4.525, 2.279], [-3.595, -10.544, 6.143], [12.047, 2.557, 11.395], [10.012, 10.774, 0.966], [-8.845, 9.54, 1.138], [-10.708, 6.006, 5.406], [-8.813, -7.143, 6.53], [-8.236, 1.186, 9.624], [2.235, -11.7, 14.044], [2.246, 10.22, 5.136], [10.425, -12.006, 19.5], [-2.849, 10.881, 5.043], [-2.581, -1.489, 14.43], [-8.722, -11.466, 0.114], [-8.667, -12.008, 19.731], [-8.013, 3.746, 15.603], [7.261, 4.723, 7.427], [-10.587, 2.383, 11.679], [2.282, -7.517, 5.239], [5.413, 3.118, 19.531], [-1.02, 1.325, 16.939], [9.067, 2.375, 19.757], [8.612, -6.167, 3.909], [6.831, -6, 3.145], [0.698, 3.952, 2.629], [7.506, 5.811, 6.771], [6.898, 5.621, 16.298], [-4.649, 7.257, 7.461], [0.994, 3.851, 15.771], [-6.032, -1.256, 9.397], [-8.454, 7.124, 13.917], [-4.337, -3.818, 5.308], [6.263, 8.276, 13.607], [-8.587, 8.236, 9.075], [-3.312, -0.324, 3.467], [11.02, -9.133, 15.471], [-1.329, -2.249, 7.458], [-10.983, 3.419, 19.734], [12.503, -8.191, 11.85], [2.359, -8.053, 16.588], [8.829, -7.273, 18.987], [3.184, 4.371, 1.48], [-3.939, -0.663, 8.989], [-7.114, -9.574, 10.069], [8.703, 6.193, 19.053], [2.447, -5.207, 18.771], [-9.293, 3.066, 3.217], [6.931, -12.268, 12.425], [2.106, 12.471, 18.195], [-12.633, 6.541, 8.689], [-9.131, -9.915, 8.862],

[-5.797, 12.273, 4.44], [0.882, 4.795, 7.314], [2.286, -5.673, 17.802], [-6.232, -0.512, 3.085], [12.352, 9.475, 12.293]

Problem instance 6: [-8.742, -10.947, 1.966], [4.21, 6.485, 4.234], [8.357, 10.473, 12.872], [-1.933, -8.586, 19.712], [10.214, 4.499, 14.972], [-12.086, -5.184, 5.411], [-10.696, 12.335, 16.055], [1.317, 3.933, 14.181], [-3.909, -11.276, 9.328], [2.789, 4.943, 2.772], [11.019, 9.601, 10.97], [-12.254, 10.25, 3.239], [9.823, 10.033, 19.512], [-4.361, -1.187, 9.299], [4.754, 11.923, 3.182], [-6.246, -4.943, 13.902], [10.342, 1.301, 10.818], [11.716, -5.704, 5.917], [1.383, 12.406, 6.802], [-3.2, 11.386, 4.677], [6.67, -12.272, 16.245], [-12.088, 10.504, 9.539], [-3.715, -12.538, 12.376], [-0.319, -2.762, 6.3], [5.311, -2.095, 3.789], [-8.576, -7.479, 17.436], [-8.473, 6.543, 10.374], [-6.677, 2.716, 0.395], [10.193, -8.76, 2.688], [8.649, -3.186, 0.645], [-3.76, 7.638, 16.618], [0.418, 6.045, 10.454], [11.405, 6.849, 3.913], [-6.855, -0.316, 19.209], [11.261, 3.441, 3.14], [9.569, 6.761, 5.8], [-1.15, 10.666, 7.531], [-12.504, -11.975, 5.591], [-0.688, 9.511, 16.624], [1.586, -6.915, 18.796], [-7.173, -3.017, 11.865], [2.812, 6.916, 9.848], [7.57, -11.489, 1.76], [10.396, 6.383, 9.799], [-2.053, 4.208, 18.906], [12.161, -2.477, 14.223], [-11.746, 8.079, 12.091], [1.842, -11.309, 16.316], [-5.705, -9.674, 15.376], [-3.305, -11.014, 6.761], [8.452, 7.943, 4.483], [6.912, -2.426, 1.412], [-7.634, 2.821, 6.294], [-7.981, -5.76, 6.063], [-6.952, -6.938, 8.031], [11.517, 8.571, 4.707], [-9.873, 9.413, 2.159], [0.435, 4.208, 6.168], [5.83, -11.156, 9.593], [-3.255, 1.089, 9.414], [-11.77, 4.13, 17.297], [-6.812, -7.1, 3.429], [-1.609, 5.463, 3.581], [2.744, 5.437, 4.192], [-9.093, -7.698, 5.409], [9.208, 0.832, 3.246], [-1.552, 3.022, 3.888], [6.559, 6.997, 19.334], [3.913, -10.551, 14.939], [-10.944, 3.336, 2.889], [3.425, -10.673, 5.971], [-6.323, -8.668, 3.701], [10.958, 3.753, 2.687], [2.254, 5.911, 17.596], [-7.762, 3.42, 17.955], [-1.673, 2.21, 1.778], [7.556, 4.064, 18.881], [-5.397, -10.29, 6.56], [-2.49, -7.378, 4.769], [-3.389, -9.454, 7.711], [-1.128, -1.322, 5.145], [3.154, -8.77, 6.255], [2.607, -4.583, 9.054], [-8.648, 4.133, 15.909], [9.715, -12.543, 19.449], [0.395, 12.31, 0.116], [9.984, 1.359, 10.413], [11.302, -5.203, 16.993], [3.4, -6.488, 2.031], [2.9, -8.041, 1.243], [-0.343, 0.112, 12.293], [-7.214, 6.602, 0.9], [-3.81, 12.292, 10.62], [-6.606, -2.126, 10.763], [-5.127, -3.856, 9.688], [3.839, -4.864, 7.984],

[-8.874, 1.866, 15.639], [7.936, 4.89, 16.731], [-10.57, 4.776, 11.334], [-12.272, 5.533, 13.909]

Problem instance 7: [-2.977, -11.731, 4.635], [5.314, 1.111, 18.569], [11.093, 8.496, 19.517], [9.598, 0.065, 19.447], [-6.716, -3.772, 2.273], [12.184, -6.445, 8.534], [7.808, 8.416, 3.19], [-10.046, -9.737, 15.768], [-10.529, -8.401, 18.658], [-1.51, -8.358, 8.68], [8.643, -9.008, 5.384], [-1.259, -6.188, 14.327], [-9.467, 11.962, 15.098], [2.548, -9.895, 19.668], [-11.042, 11.61, 17.34], [11.657, 0.729, 0.627], [-1.51, 0.028, 9.008], [0.606, -9.888, 17.036], [-11.067, 7.714, 14.227], [-10.687, 4.808, 4.151], [7.461, 8.681, 11.503], [2.676, -1.318, 12.834], [-2.283, -0.967, 5.835], [-4.375, -6.316, 6.496], [-11.503, -5.372, 15.109], [-2.037, 4.433, 11.053], [-11.073, -3.227, 11.29], [-10.303, 4.163, 2.679], [1.553, 4.024, 12.451], [-10.13, 0.336, 2.564], [-5.639, -9.505, 7.561], [-9.641, -9.807, 1.66], [-11.202, 2.959, 19.394], [-10.15, -4.021, 9.946], [11.349, 7.891, 9.297], [1.809, 1.453, 7.677], [-7.211, -10.403, 5.582], [-11.108, 4.258, 11.832], [9.6, 4.997, 12.035], [-5.467, 1.791, 13.452], [-10.592, 11.704, 9.623], [12.345, 10.131, 2.969], [-8.317, -2.714, 5.054], [11.447, -2.615, 1.462], [-3.858, -3.423, 13.381], [9.773, 7.533, 3.794], [4.827, -6.189, 0.614], [-0.236, -7.949, 10.23], [6.264, 11.251, 7.98], [11.404, 8.27, 11.309], [2.483, 10.972, 8.327], [12.158, 1.866, 11.534], [8.574, 12.456, 5.395], [-9.084, -8.93, 10.293], [-10.46, -0.151, 0.603], [-0.222, 4.592, 16.168], [-12.176, -11.95, 15.523], [-2.242, -9.76, 19.42], [-8.504, 0.671, 3.849], [-4.061, -11.527, 19.926], [-7.957, -11.059, 17.053], [9.079, -8.263, 1.379], [5.049, -8.561, 2.459], [-11.581, -12.551, 16.344], [-6.694, 5.181, 11.138], [3.285, -7.882, 3.575], [-11.653, 7.881, 19.488], [12.355, 7.447, 10.205], [-5.738, -5.213, 14.014], [-6.479, 3.603, 10.237], [4.621, 1.799, 6.112], [3.947, 0.317, 18.528], [-3.365, -9.656, 19.951], [1.527, 10.324, 6.689], [-3.553, -8.347, 5.632], [-12.045, -5.954, 14.221], [5.353, 10.462, 10.504], [-8.624, -2.451, 16.859], [-11.923, 5.403, 19.878], [0.002, -4.506, 15.211], [-5.909, 10.277, 5.091], [6.882, -2.147, 4.675], [-10.316, 3.79, 18.376], [11.732, -7.194, 17.778], [0.239, 5.068, 14.756], [3.216, 4.768, 7.644], [1.724, -0.219, 19.036], [-2.846, -10.714, 13.515], [-8.101, 11.545, 10.855], [-9.527, 12.629, 0.962], [0.879, 0.26, 15.494], [-4.012, 12.25, 15.559], [7.229, 6.314, 18.972], [4.681, 4.711, 0.267], [-11.888, 6.301, 8.732], [-2.01, -5.145, 8.299], [5.12, -0.392, 1.421], [-7.317, -1.728, 7.253],

[11.418, -5.052, 3.557], [-4.484, 6.988, 8.499]

Problem instance 8: [-9.618, 7.636, 12.086], [-5.975, 12.051, 11.329], [3.45, 7.352, 18.516], [6.554, -2.63, 4.929], [3.262, -1.93, 14.808], [0.272, 3.196, 0.342], [-3.156, 2.523, 10.204], [4.53, -8.846, 1.415], [0.655, -9.893, 12.247], [-0.479, -0.429, 17.919], [-8.94, -3.257, 17.803], [-3.965, -7.184, 9.652], [-4.304, 12.391, 3.955], [-0.569, 8.85, 0.322], [2.617, 8.235, 2.191], [2.48, 5.165, 18.785], [-9.665, 11.995, 2.62], [10.008, -5.752, 4.381], [-2.502, 3.627, 14.071], [-1.989, -4.923, 13.04], [5.544, 5.627, 7.315], [2.9, -2.478, 9.32], [4.335, 8.232, 19.273], [9.621, 10.642, 19.853], [-9.185, 11.612, 0.627], [7.074, 12.237, 11.567], [3.819, 6.058, 8.756], [-8.665, -11.943, 9.424], [2.15, 5.321, 16.464], [3.595, -5.21, 5.685], [12.076, -1.387, 15.444], [2.119, -9.752, 17.785], [3.531, 6.075, 3.866], [7.681, 8.85, 5.23], [-1.734, -7.393, 0.874], [6.104, 1.825, 2.069], [4.091, -8.318, 13.46], [-9.582, 8.488, 8.279], [-7.473, 4.886, 17.283], [3.271, -5.938, 12.28], [9.194, -12.601, 7.165], [-3.692, -0.054, 4.041], [9.7, -8.635, 19.606], [-7.054, 9.344, 15.425], [-2.351, 3.251, 8.112], [4.373, 11.319, 2.301], [12.016, -10.924, 13.755], [7.231, -1.65, 6.279], [8.435, 3.118, 4.205], [5.955, 9.576, 1.349], [-1.868, 9.287, 15.003], [6.964, -0.492, 7.103], [6.235, -7.146, 17.926], [11.724, 2.937, 16.498], [8.009, -0.835, 9.684], [-2.106, -1.888, 8.728], [9.035, -12.448, 10.581], [-3.741, -8.163, 17.145], [1.207, 4.991, 0.073], [-4.753, -11.063, 6.492], [9.13, 3.604, 18.383], [4.912, -7.206, 2.801], [-1.139, 7.623, 10.501], [-1.141, 7.217, 15.083], [6.822, -8.826, 3.309], [0.628, 11.476, 9.972], [6.093, -9.588, 3.477], [1.72, -0.946, 8.82], [-1.477, 11.034, 16.004], [6.81, -7.837, 7.504], [-2.326, -8.961, 11.899], [-1.633, -10.407, 15.753], [-0.113, 1.457, 15.077], [9.453, -8.787, 4.088], [-11.308, -7.626, 11.06], [-7.76, 7.653, 12.387], [-11.94, -4.232, 1.064], [7.158, -8.904, 5.988], [-4.43, -5.004, 5.041], [2.561, -5.04, 16.465], [-6.574, -10.078, 14.764], [-10.697, -7.268, 0.035], [12.022, -8.789, 17.477], [-11.795, 5.444, 4.265], [-8.545, 1.089, 7.696], [-1.934, 3.061, 12.492], [11.397, 6.072, 2.599], [8.83, 6.386, 10.145], [-1.93, 4.861, 10.502], [-6.519, 7.805, 10.156], [-11.013, -0.457, 9.59], [-1.554, -4.671, 8.47], [-10.151, 12.011, 19.433], [1.756, 12.247, 13.706], [-1.518, 10.029, 10.896], [-1.254, 3.915,

11.337], [12.363, 1.907, 5.267], [-11.168, -12.402, 6.889], [-7.166, -10.556, 16.739], [1.975, -4.248, 9.632]

Problem instance 9: [-3.269, -12.364, 0.69], [-3.372, 4.375, 19.162], [-7.462, -9.89, 4.057], [2.489, -4.057, 5.559], [7.179, -9.746, 0.4], [-9.422, -4.39, 7.635], [-5.301, -7.556, 0.179], [-0.753, 3.215, 9.586], [-7.773, -11.437, 19.424], [-7.365, 12.257, 4.893], [-4.57, -4.687, 5.256], [3.698, -2.254, 8.19], [3.522, 2.116, 4.807], [5.151, 4.126, 4.96], [-10.498, -10.47, 14.013], [12.387, 5.501, 13.133], [-11.346, 11.183, 1.832], [10.319, 3.347, 15.116], [-12.01, 6.917, 0.088], [-0.007, -6.076, 1.035], [-2.958, -1.065, 1.893], [-9.528, 6.26, 19.831], [-0.566, -4.285, 15.42], [0.493, -5.517, 18.417], [4.291, -6.307, 15.359], [-4.149, -9.415, 18.732], [-4.176, -7.38, 16.716], [1.878, -5.36, 11.914], [-5.459, -6.694, 3.197], [-10.37, 11.479, 3.384], [-9.435, -9.971, 0.219], [-9.913, -1.974, 11.47], [-0.457, 11.14, 11.754], [4.946, 2.701, 10.66], [10.125, -7.016, 9.828], [-10.089, -5.895, 14.069], [-4.09, 9.494, 5.121], [5.298, 2.932, 13.972], [-2.27, -3.337, 17.73], [3.557, -8.986, 1.335], [-10.186, -8.424, 1.786], [11.378, 10.532, 16.882], [-11.856, 4.134, 2.787], [-5.974, 8.78, 11.437], [6.088, -1.902, 15.754], [-7.783, 5.464, 13.636], [5.74, -11.196, 3.785], [11.225, 3.605, 2.262], [7.686, -4.456, 5.49], [12.328, 5.429, 15.51], [-6.795, -7.176, 16.576], [-4.224, -4.893, 7.399], [10.984, -7.736, 3.743], [4.459, -10.925, 7.916], [-3.325, -7.874, 7.91], [-9.868, -6.454, 16.422], [3.205, 4.046, 19.897], [-0.562, 8.391, 6.979], [12.515, 9.865, 13.986], [7.802, 4.947, 1.499], [-1.118, -4.086, 5.547], [6.354, -12.57, 3.049], [-1.65, 8.81, 19.706], [-6.632, 12.088, 17.996], [9.033, -7.763, 5.043], [6.568, -6.875, 7.228], [-7.176, 8.297, 4.438], [1.326, -3.344, 1.035], [-8.131, -1.663, 14.363], [11.641, -10.327, 5.555], [-2.304, -6.478, 18.936], [1.04, 6.809, 14.24], [6.238, -5.914, 11.418], [-6.724, 12.313, 6.884], [-11.801, -0.306, 13.319], [1.717, -9.93, 12.055], [-10.816, 4.833, 12.906], [0.169, 7.388, 3.572], [-11.35, 3.037, 11.309], [1.915, -8.869, 18.399], [1.554, -4.554, 14.126], [1.978, -1.35, 13.766], [1.858, 3.23, 19.534], [-8.716, -11.853, 11.629], [-6.452, -9.914, 11.749], [6.686, 12.313, 6.082], [-1.342, -11.241, 5.793], [3.783, 5.499, 15.332], [-8.256, 8.046, 4.409], [-2.23, -8.64, 11.334], [-12.015, -2.785, 1.286], [-11.329, 2.245, 5.513], [6.811, 8.68, 3.302], [0.764, 10.714, 10.743], [0.814, 11.175, 8.916], [5.286, 6.981,

9.21], [-1.08, -0.818, 10.918], [11.187, -2.742, 10.817], [-3.174, -5.977, 9.14], [5.037, 2.423, 17.3]

Problem instance 10: [2.943, 7.916, 2.432], [-1.282, -0.75, 8.911], [8.444, 8.817, 4.89], [4.896, -5.592, 19.765], [0.244, 1.693, 11.362], [5.473, 0.517, 13.669], [-11.127, 6.653, 16.373], [3.778, 7.641, 11.378], [-4.417, -10.709, 8.271], [11.565, 5.17, 19.93], [-7.172, 10.349, 0.389], [-1.235, -4.404, 15.35], [1.936, -8.119, 7.338], [-1.168, 10.974, 7.232], [-2.412, 6.433, 18.935], [-11.218, -12.247, 16.754], [-4.64, 3.077, 13.13], [-4.786, -5.645, 4.36], [6.133, -1.797, 16.088], [-10.61, -9.424, 16.385], [-10.171, -4.361, 8.741], [2.965, -1.478, 5.719], [-6.615, 8.81, 17.658], [1.607, 1.799, 6.896], [10.967, -0.768, 6.717], [6.374, -1.979, 0.201], [2.135, 4.313, 6.454], [-2.228, -12.013, 3.289], [-6.111, -1.283, 1.998], [12.38, -7.179, 14.343], [10.759, 3.967, 7.084], [9.79, 8.501, 12.915], [-5.531, -9.705, 1.272], [5.12, -1.99, 3.344], [6.154, 0.131, 7.431], [11.567, 10.693, 1.958], [-6.415, 4.563, 9.428], [-7.418, 0.17, 3.811], [-8.924, 8.497, 4.69], [-4.924, -5.98, 8.286], [4.388, -3.198, 16.521], [-12.255, -10.414, 3.542], [2.839, -5.141, 19.301], [-6.881, -9.594, 5.814], [-2.489, 4.622, 10.979], [2.395, -7.893, 15.315], [11.627, -1.229, 16.199], [5.065, 8.249, 19.338], [9.623, 0.52, 7.279], [-10.254, 3.714, 17.83], [9.284, 1.178, 2.509], [4.833, 2.02, 0.858], [-5.812, -6.704, 16.202], [-9.531, 4.384, 15.356], [6.032, 0.591, 11.41], [2.016, 0.154, 12.522], [4.928, 9.289, 4.648], [-11.294, -3.668, 2.806], [-8.62, -6.904, 4.737], [-9.162, 2.814, 3.119], [-6.415, 10.692, 18.221], [4.442, -4.069, 3.385], [1.588, -11.562, 6.345], [-3.302, -9.642, 12.14], [8.748, 5.691, 14.506], [8.897, -10.95, 0.805], [-10.184, -2.633, 8.057], [7.618, -1.831, 10.831], [7.014, 3.846, 16.897], [11.503, 7.422, 14.092], [-0.947, 10.541, 9.031], [0.694, -0.64, 3.754], [6.636, 1.014, 18.87], [5.646, -5.443, 17.664], [-5.267, -11.79, 0.424], [-10.72, -1.71, 8.619], [-1.018, -2.937, 0.053], [6.624, 0.552, 9.238], [-10.534, 12.301, 1.187], [11.212, 2.101, 1.46], [-0.265, 8.228, 7.891], [-10.86, -7.464, 6.031], [3.797, -4.066, 5.626], [7.644, 7.595, 4.296], [5.445, 6.519, 16.725], [-3.541, -10.011, 15.248], [-0.77, -4.894, 5.555], [0.975, 6.139, 3.227], [8.107, 6.646, 1.64], [8.942, 10.952, 11.063], [-2.55, 10.216, 4.098], [10.474, -6.955, 17.282], [-0.823, -5.334, 13.838], [-6.456, 11.252, 12.18], [4.216, -9.329, 14.415], [-10.169, -9.939, 18.697], [8.748,

-5.168, 7.536], [0.651, 7.051, 3.13], [0.004, -0.404, 7.353], [-6.981, 8.715, 9.055]

A.6 The 150 user scenario

Problem instance 1: [3.407, 10.441, 0.013], [13.426, 0.3, 5.11], [-13.29, -11.287, 19.153], [-6.542, -8.681, 14.729], [13.832, -14.083, 14.45], [4.254, 15.289, 15.753], [5.161, -13.625, 19.662], [-13.624, -0.668, 14.269], [-12.497, -2, 3.737], [-6.524, 8.363, 14.803], [-4.393, 0.203, 8.585], [-4.323, -15.431, 12.228], [-11.397, 1.041, 1.925], [0.261, -8.304, 10.052], [14.264, -6.645, 12.703], [1.528, -8.667, 1.513], [11.434, 7.663, 4.606], [8.799, -3.397, 14.304], [-14.484, 1.067, 10.943], [-15.297, -7.717, 10.304], [14.719, -13.366, 5.83], [2.883, 6.261, 0.042], [6.354, 12.152, 0.755], [-15.145, 14.305, 7.282], [1.478, 8.079, 6.955], [-9.843, 12.354, 17.293], [9.908, 4.886, 9.802], [-14.478, -6.734, 7.194], [10.643, -14.028, 18.491], [0.124, -14.682, 4.166], [-10.622, -15.458, 9.775], [-14.786, 10.756, 14.335], [-0.298, -4.961, 13.144], [-4.555, 0.149, 17.368], [-12.386, 3.204, 1.009], [1.704, -11.322, 19.711], [-8.208, 13.545, 7.03], [9.945, -11.508, 19.099], [5.676, -7.041, 8.193], [14.834, 13.645, 0.284], [6.809, -1.404, 10.279], [-1.042, -8.339, 19.315], [-5.407, -13.109, 13.917], [-9.958, -5.034, 12.623], [9.721, -1.796, 18.098], [-5.297, -12.941, 16.108], [8.581, 4.602, 18.198], [-3.732, 13.037, 12.932], [-6.201, -7.242, 1.301], [-4.888, 11.265, 12.931], [2.303, -9.121, 1.617], [7.895, 4.637, 1.716], [-10.277, -1.451, 18.109], [-6.958, 10.482, 2.69], [5.634, -2.48, 10.246], [10.326, -1.297, 4.575], [3.872, 1.999, 5.374], [-9.228, -13.872, 2.798], [11.113, 13.494, 5.496], [-3.202, -4.905, 13.783], [-13.223, -12.432, 18.704], [3.294, 4.265, 6.007], [-7.435, 14.976, 7.005], [-14.371, -5.418, 1.601], [6.547, -1.105, 7.818], [-9.899, 3.048, 11.707], [-5.055, -4.277, 9.733], [-6.241, 8.834, 3.81], [-15.337, 9.111, 2.828], [12.024, -3.207, 9.626], [-9.339, -2.86, 2.721], [-3.787, -4.367, 5.35], [15.308, 2.9, 14.046], [-6.242, 11.504, 14.616], [13.885, -12.734, 13.559], [-14.112, 5.175, 19.723], [-7.893, -0.527, 11.9], [-8.913, 4.948, 11.29], [5.589, 13.422, 7.164], [13.549, 10.444, 16.289], [-2.125, -5.36, 12.526], [5.394, -12.429, 10.574], [-12.304, -11.5, 5.59], [-9.346, 5.498, 4.192], [4.541, 9.21, 17.629], [1.69, -6.608, 15.95], [13.025, 9.711, 18.686], [-0.552, 10.099, 10.804], [-4.487, -13.159, 17.785], [-13.149, -7.451,

6.505], [-12.828, -10.728, 12.94], [9.889, -12.817, 12.492], [-2.631, -4.926, 11.606], [-14.953, -8.98, 9.002], [-2.943, 11.586, 11.23], [-1.072, -5.085, 0.947], [-14.677, 11.093, 19.401], [5.598, 10.402, 1.795], [-13.514, -1.577, 13.446], [2.928, 8.16, 0.478], [5.055, 4.621, 16.825], [-0.795, -6.667, 17.41], [-13.005, 3.429, 17.818], [-5.859, -9.829, 5.925], [-11.868, 14.503, 6.585], [2.892, 4.162, 3.004], [4.362, -5.438, 17.225], [9.853, -0.983, 19.897], [11.087, 0.031, 10.633], [-3.534, 1.708, 17.287], [-13.719, -11.191, 8.493], [1.972, -9.334, 11.908], [2.979, 1.199, 5.789], [5.564, -2.693, 17.42], [-12.201, -4.575, 8.782], [7.848, -7.385, 15.093], [-7.402, 1.45, 4.921], [4.862, 3.54, 1.892], [-2.795, -7.313, 10.249], [9.455, 13.507, 1.696], [4.879, 0.425, 14.073], [14.993, 1.863, 19.624], [9.576, 3.413, 18.683], [4.94, -0.303, 19.3], [-8.611, -10.165, 11.312], [1.958, 14.731, 2.222], [-3.266, -7.621, 11.195], [-0.197, 5.544, 16.721], [9.794, 14.493, 17.339], [-11.601, -7.649, 6.097], [-6.381, -11.459, 15.426], [-11.456, 1.309, 12.018], [-11.264, -3.934, 10.494], [7.121, -11.191, 18.734], [2.755, 9.621, 17.165], [-5.814, 0.872, 10.471], [-7.841, 13.345, 7.825], [6.07, 15.307, 10.097], [-9.805, -0.819, 6.269], [-10.82, -3.313, 17.763], [-2.233, 1.575, 2.77], [-10.054, -14.961, 0.509], [-2.75, 10.957, 0.38], [-1.336, -15.453, 9.787], [-13.921, -3.07, 18.709], [-8.055, 3.44, 2.535], [8.026, -8.727, 18.795], [15.377, 9.763, 16.278], [2.132, -8.266, 6.296], [-13.89, -10.686, 14.343]

Problem instance 2: [-13.634, 13.014, 16.264], [-9.592, -7.623, 1.571], [-6.037, 7.223, 16.276], [10.262, -2.769, 16.322], [1.23, 11.462, 11.183], [7.443, -4.548, 10.679], [-0.75, 9.77, 9.016], [1.854, 15.393, 13.796], [-12.083, -2.81, 5.947], [13.57, 1.743, 12.313], [-6.382, 4.728, 9.247], [-11.52, 2.944, 10.161], [7.178, -11.009, 2.67], [10.215, 7.191, 0.845], [5.171, 14.104, 18.317], [10.599, 2.4, 10.013], [-1.183, 12.732, 16.846], [-9.265, 9.697, 8.267], [14.643, 2.314, 0.845], [-13.274, 7.967, 11.699], [10.949, -0.668, 2.33], [-10.659, -6.104, 19.088], [-14.231, 11.314, 9.598], [7.227, 12.026, 9.552], [11.934, 9.956, 8.333], [8.152, 2.146, 19.653], [-9.741, 2.599, 9.7], [-3.129, 14.888, 0.65], [-0.111, -0.503, 16.267], [-3.99, 4.739, 1.652], [-0.793, 12.817, 6.83], [10.422, 11.594, 12.227], [5.557, 5.957, 11.02], [7.822, -6.823, 1.113], [1.477, 2.993, 9.313], [-3.862, -11.885, 11.642], [-5.11, 4.333, 0.817], [8.268, -0.522, 0.667], [7.099, -14.191,

7.482], [12.815, 13.262, 2.596], [-9.595, 6.109, 8.587], [-10.139, 1.784, 5.198], [-9.277, 13.324, 17.39], [-2.209, -13.174, 7.763], [14.232, -4.892, 13.669], [13.895, -12.595, 18.338], [5.14, 12.985, 7.284], [5.724, -7.379, 18.8], [-12.731, 5.525, 11.776], [-4.74, -3.925, 3.255], [-2.442, -3.864, 9.753], [-8.467, -11.383, 9.477], [10.197, -12.009, 17.368], [10.231, 5.349, 16.078], [1.23, -11.222, 4.173], [-13.595, -9.801, 15.493], [-3.98, -2.054, 7.789], [1.087, 13.145, 9.345], [-5.954, -2.355, 16.332], [9.388, 11.351, 3.99], [5.281, 10.707, 16.745], [-6.412, -12.574, 2.972], [0.753, 12.198, 2.667], [3.172, -14.894, 9.704], [-1.754, 9.648, 12.744], [-1.988, 0.434, 18.737], [1.499, -10.168, 12.95], [-10.941, -3.967, 0.231], [-13.961, -9.323, 19.123], [-5.088, -13.764, 3.749], [1.196, 6.522, 2.71], [-7.132, 0.96, 19.922], [-1.246, -13.953, 2.161], [6.812, -13.608, 0.188], [-4.354, 2.493, 3.338], [5.477, 3.953, 6.197], [-7.082, -1.388, 1.828], [-7.832, 14.549, 17.989], [-12.715, -6.061, 2.727], [14.44, -4.53, 14.912], [-7.348, -7.598, 18.426], [6.563, -7.961, 18.088], [-0.759, -1.955, 12.274], [-4.583, -13.133, 1.463], [-9.22, 10.372, 17.752], [10.224, 8.234, 14.086], [-1.979, 13.122, 17.943], [-11.284, -8.085, 14.839], [9.825, 11.319, 2.049], [-1.989, -5.354, 5.718], [-9.637, 14.305, 0.854], [-1.363, -0.458, 12.465], [-9.588, -14.748, 14.466], [-13.082, 6.42, 0.254], [0.12, -14.585, 10.529], [2.878, -14.1, 5.22], [-13.991, 3.194, 5.602], [8.621, -7.932, 4.373], [-4.194, -10.901, 8.182], [9.494, -6.943, 13.997], [1.234, -5.096, 7.97], [8.484, -9.944, 13.486], [1.437, 0.393, 16.65], [5.112, 9.883, 15.107], [14.834, -5.841, 8.717], [4.612, -11.8, 9.566], [-7.82, 14.417, 15.273], [-4.611, -9.695, 10.04], [-14.9, -14.567, 17.348], [-1.39, -0.811, 13.725], [0.511, 9.637, 13.329], [-11.384, -10.215, 2.19], [0.895, 2.09, 15.539], [-5.404, 10.077, 0.848], [11.626, 12.144, 0.714], [-4.52, -2.725, 13.343], [13.818, -5.496, 2.145], [-9.333, 5.556, 19.547], [-4.012, -8.028, 17.988], [10.309, 8.383, 9.211], [-2.937, -9.968, 9.927], [11.119, 5.683, 0.544], [-3.377, -9.964, 10.933], [1.793, 9.29, 17.672], [5.628, -2.036, 5.134], [-13.543, -7.092, 0.232], [-15.053, 12.461, 19.448], [2.831, 0.331, 4.623], [7.481, -11.822, 14.778], [-2.383, -6.092, 17.256], [10.661, -4.607, 9.691], [0.229, 15.435, 2.088], [12.367, -4.315, 14.037], [-13.179, 10.191, 16.357], [-11.856, 0.141, 16.239], [-3.344, 1.09, 18.25], [3.201, -14.32, 4.661], [-7.503, -1.679, 16.421], [-4.988, -14.135, 6.768], [-11.634, 4.259, 2.78], [-15.457, 7.754, 9.765], [9.025, -8.701, 0.829], [5.923, -10.348, 11.588], [11.532, -6.499, 6.005], [-4.058, -5.916, 10.526], [8.535, 9.279, 12.01],

[-1.991, 1.218, 9.335], [-11.07, -6.651, 15.966], [13.814, 6.541, 12.258], [-4.843, -6.987, 2.7]

Problem instance 3: [-13.351, 10.832, 0.317], [-9.764, 10.086, 8.453], [5.737, -15.291, 9.813], [-2.79, 2.493, 13.425], [3.344, 14.201, 3.374], [3.291, 9.883, 3.359], [15.145, 2.235, 14.871], [-13.553, 10.775, 18.653], [-14.234, 2.269, 2.583], [-0.429, -11.059, 18.295], [-0.92, -3.829, 6.404], [6.03, 3.098, 19.06], [12.279, 8.918, 7.024], [1.85, -9.621, 18.024], [4.387, 8.671, 5.174], [0.269, -5.447, 17.889], [-8.235, -6.147, 17.724], [-2.186, -2.761, 10.177], [-9.233, 13.478, 0.987], [4.252, 1.07, 15.58], [-5.976, -2.169, 17.143], [4.026, -6.871, 9.306], [11.305, -9.192, 18.278], [6.604, 12.528, 5.954], [-7.198, -2.249, 14.029], [-8.373, 9.737, 0.981], [-0.985, -10.805, 3.029], [9.93, -13.879, 14.799], [8.874, -13.357, 1.984], [1.882, -9.542, 2.361], [-12.78, 10.631, 13.116], [-9.17, -6.004, 2.613], [-0.709, 8.476, 6.345], [-4.379, -8.588, 7.044], [10.167, -13.95, 8.951], [-11.246, -13.588, 18.027], [-3.282, -15.029, 11.516], [10.965, 1.002, 18.171], [8.79, -2.696, 7.65], [6.733, -12.73, 7.521], [6.359, 5.996, 19.24], [15.021, 14.41, 8.083], [-2.946, 9.279, 15.402], [-12.498, -1.287, 11.301], [8.461, 14.202, 4.504], [-3.296, 6.685, 18.638], [-8.191, -3.472, 11.642], [11.905, 6.11, 17.364], [-10.999, 3.931, 1.108], [-13.603, -10.111, 13.617], [6.946, 5.786, 17.737], [-4.753, -14.831, 12.164], [10.44, -15.1, 2.338], [15.245, 6.35, 7.92], [-3.113, -11.044, 7.936], [-10.816, 14.003, 11.292], [14.087, -6.698, 10.95], [11.1, 10.186, 14.411], [14.522, -9.779, 13.165], [9.523, 13.481, 5.995], [10.01, -9.308, 19.517], [-6.25, -10.613, 9.469], [-6.129, -14.229, 9.408], [-14.474, -6.704, 2.824], [14.098, -10.545, 14.893], [-6.439, 0.881, 11.666], [-4.161, -0.372, 19.139], [-3.433, 10.955, 15.701], [-4.539, -14.381, 8.455], [12.941, 9.982, 14.294], [13.562, 12.923, 7.301], [-5.737, -0.473, 0.043], [-8.301, 5.71, 12.003], [5.611, -10.026, 13.447], [9.632, 3.865, 18.3], [4.284, -3.181, 19.942], [6.503, -3.757, 15.228], [2.811, -2.036, 5.88], [11.294, 8.215, 10.96], [1.168, 6.797, 15.77], [3.679, 4.288, 8.683], [3.178, 3.275, 16.282], [8.875, 12.337, 12.729], [9.834, 10.846, 1.864], [-9.669, -7.673, 19.174], [-7.282, 4.334, 6.936], [6.025, -12.321, 2.748], [12.34, -5.586, 6.573], [-1.39, 1.324, 15.319], [7.398, 2.469, 4.473], [-3.514, 14.684, 12.467], [-0.283, 5.169, 9.634], [4.376, 11.209, 16.751], [-12.559, -15.233, 3.86], [-8.714, -1.272, 13.289], [6.277, -2.088, 16.342], [0.521, 9.89, 0.354], [10.856, 11.79, 4.414],

[-9.531, -5.594, 19.771], [-12.588, 6.317, 6.543], [-8.476, -4.576, 7.258], [-1.151, 9.132, 10.331], [-0.43, -4.093, 10.37], [-13.631, 14.17, 18.417], [-9.467, 2.587, 19.573], [12.645, -12.427, 8.429], [7.37, 11.765, 7.14], [4.315, 1.172, 7.333], [15.31, 13.899, 12.641], [5.453, -11.912, 9.957], [-6.375, 1.943, 0.471], [-1.84, 8.933, 19.037], [-2.076, -9.741, 1.34], [1.725, -11.679, 0.36], [-6.363, 0.228, 11.289], [6.221, 6.298, 7.67], [-12.131, -15.365, 4.856], [-2.409, -2.249, 11.663], [0.838, -2.798, 10.468], [-12.95, 13.596, 18.435], [7.082, 7.324, 0.521], [8.86, -11.715, 9.845], [-12.42, -12.11, 11.997], [9.41, -2.08, 14.345], [-4.3, 0.725, 6.939], [3.402, 13.227, 12.449], [7.774, -0.521, 16.781], [6.788, 0.414, 17.401], [4.925, -9.943, 1.171], [-1.45, 12.474, 16.226], [-8.859, 12.831, 5.961], [12.15, 5.937, 7.154], [12.185, 13.124, 11.502], [-7.094, 11.849, 16.562], [3.97, -13.254, 13.916], [10.144, 13.606, 10.741], [-9.544, -13.474, 5.508], [15.353, -0.348, 18.327], [5.522, -1.616, 14.203], [-5.196, 5.233, 3.93], [3.859, 6.91, 17.184], [-9.421, -11.421, 1.54], [-9.557, 13.034, 3.731], [13.666, -12.258, 12.644], [7.919, -2.806, 3], [5.863, -13.141, 6.648], [11.981, 3.683, 18.014], [-1.283, 3.034, 12.203], [-12.919, 14.329, 10.311], [-4.405, 8.569, 16.216]

Problem instance 4: [7.998, -14.994, 4.379], [-3.934, -1.361, 0.317], [4.511, 15.174, 5.042], [1.13, 2.693, 13.085], [14.865, -14.384, 1.632], [7.776, -1.827, 3.663], [2.033, -8.446, 0.488], [11.288, 10.011, 12.139], [6.888, -11.245, 18.476], [3.511, -10.101, 13.341], [3.412, 8.073, 16.015], [-0.035, -4.128, 4.304], [-5.601, 13.523, 0.392], [-7.074, 13.672, 0.617], [-11.394, 9.619, 18.274], [7.55, -8.406, 17.171], [12.713, 15.373, 7.79], [-14.747, 11.911, 11.239], [-9.492, 4.546, 18.628], [-6.916, -13.537, 8.739], [-12.205, 7.252, 3.949], [7.889, 8.754, 8.156], [-7.51, 3.96, 4.196], [-2.557, -6.123, 4.303], [-1.298, 3.799, 2.976], [5.287, -14.778, 14.047], [3.487, 14.384, 7.218], [8.891, 3.32, 14.707], [3.63, -5.631, 11.635], [1.457, -13.311, 12.252], [8.014, -0.909, 16.254], [12.062, -3.262, 15.11], [-14.261, -2.863, 2.702], [-10.803, -2.96, 2.029], [-3.252, 11.629, 19.324], [-1.708, -5.703, 11.953], [12.444, -9.192, 7.675], [-4.093, -2.775, 12.774], [4.261, -10.438, 7.67], [1.532, -4.987, 0.047], [-14.685, 8.056, 10.272], [-12.289, -11.376, 7.34], [-5.505, 15.063, 16.591], [-3.391, 1.72, 15.162], [2.801, 1.366, 10.452], [-6.652, -5.86, 9.934], [-9.898, -8.05, 10.825], [9.878, 3.94, 0.946], [12.672, -9.529, 10.487], [2.484, -13.436, 8.439], [-8.739,

-6.878, 18.14], [-12.654, -1.881, 0.171], [-7.324, -8.066, 9.412], [-1.201, -8.309, 7.455], [2.437, -11.698, 17.721], [-4.58, 12.024, 18.667], [-10.649, -11.709, 15.757], [6.878, 2.822, 16.902], [-11.595, 1.201, 10.426], [2.144, 11.283, 18.491], [-6.62, 5.232, 18.265], [-5.499, 13.888, 18.962], [-9.219, 13.79, 18.402], [13.393, 8.664, 5.144], [3.778, 3.196, 17.681], [-13.512, 10.857, 3.011], [-6.159, 5.14, 6.328], [-13.933, -7.773, 8.086], [-3.075, -11.769, 2.225], [13.641, 14.689, 10.388], [2.991, -1.274, 0.032], [12.806, -2.867, 1.876], [5.702, 3.05, 6.825], [-5.349, -8.084, 18.546], [8.827, 11.307, 19.986], [7.529, -0.301, 4.465], [-1.681, -11.916, 2.666], [3.978, -14.744, 1.16], [-10.21, 14.127, 18.134], [5.759, -12.508, 9.275], [-14.742, -0.665, 7.445], [9.635, 11.867, 19.933], [-2.639, -9.81, 17.152], [10.279, -0.533, 4.884], [-7.761, 14.149, 1.779], [10.661, -6.064, 8.651], [-9.169, -0.1, 15.009], [4.647, 14.453, 17.143], [-6.777, -12.472, 3.831], [-12.819, -11.25, 18.579], [3.179, 8.108, 6.828], [-7.394, -13.645, 14.486], [8.43, -2.896, 11.66], [6.55, -9.276, 12.913], [7.18, 5.792, 4.626], [-3.074, 14.484, 19.164], [5.846, 2.222, 11.234], [-13.137, 13.935, 1.256], [13.059, 3.463, 9.337], [5.987, -7.744, 3.086], [4.038, 15.088, 5.003], [-12.847, 7.581, 5.386], [-4.249, 3.637, 1.989], [2.428, 6.691, 11.751], [-5.392, 4.411, 3.92], [5.099, -10.17, 12.638], [-14.666, 8.205, 15.778], [10.892, -6.467, 3.437], [-10.953, -1.677, 12.921], [-8.725, 1.125, 3.726], [1.032, -5.605, 5.379], [10.333, -7.547, 16.474], [13.238, -1.269, 19.779], [-9.38, 7.072, 13.021], [0.231, 11.186, 14.862], [-7.566, -4.394, 15.297], [-10.896, 9.29, 2.669], [11.366, 6.86, 8.822], [11.615, -0.951, 9.176], [6.04, -4.475, 1.807], [11.65, 14.716, 13.998], [-11.305, -13.023, 0.378], [10.703, 14.382, 19.542], [-10.663, -4.169, 0.007], [-0.071, 7.434, 18.943], [4.114, -4.96, 17.47], [-10.738, -1.81, 13.858], [1.891, -2.932, 15.329], [-1.442, -6.42, 4.622], [1.409, 11.424, 7.159], [1.634, -7.307, 11.123], [9.776, -12.804, 18.33], [-11.687, -5.135, 17.634], [-13.018, 2.111, 11.792], [10.574, -4.094, 1.492], [3.528, -0.655, 15.038], [-10.332, -7.381, 19.165], [-14.009, 7.291, 5.065], [2.062, 11.163, 16.233], [10.097, -14.06, 16.514], [5.104, 15.198, 14.909], [-5.095, 10.881, 3.291], [0.58, -15.337, 11.607], [-9.845, 12.757, 0.841], [11.522, 2.057, 18.013], [-12.331, 11.886, 3.856], [14.648, 3.638, 7.286], [-7.559, -13.745, 17.126], [-12.045, -2.458, 7.325], [-4.285, -12.123, 6.62]

Problem instance 5: [-8.113, -6.875, 5.265], [10.903, -0.757, 16.87], [5.798, -3.019, 15.504], [-7.304, -7.434, 2.493], [10.747, -11.529, 9.111], [-7.732, 13.608, 7.032], [6.836, 10.831, 0.603], [12.561, -5.578, 11.039], [-11.107, -5.052, 2.82], [8.126, 4.707, 5.066], [-3.421, 9.708, 13.456], [-5.115, -8.719, 0.458], [-10.063, -15.48, 9.832], [-8.897, 12.555, 12.081], [-0.274, -10.563, 13.064], [11.27, -4.216, 17.17], [-8.815, 10.253, 6.903], [6.482, 13.328, 2.404], [10.835, 15.176, 9.041], [11.903, 5.105, 6.417], [0.09, -0.572, 18.423], [-10.805, 5.169, 15.584], [-0.987, 7.068, 10.464], [9.423, 3.565, 5.727], [11.953, -8.587, 7.859], [-1.891, -7.365, 11.065], [-9.361, 12.89, 12.474], [0.828, -3.113, 2.861], [-4.715, 14.451, 2.661], [8.344, 10.546, 11.968], [3.282, -8.221, 8.615], [-10.807, 12.411, 10.234], [-3.226, -11.513, 12.609], [4.78, 15.353, 4.975], [-4.236, -11.521, 5.043], [1.473, -0.559, 1.589], [4.673, -15.451, 7.352], [7.686, 1.534, 1.715], [-2.364, 2.382, 7.809], [-9.868, 2.805, 10.048], [11.377, -7.47, 5.129], [11.366, 14.975, 5.473], [-0.94, 7.846, 2.858], [6.002, -14.451, 19.529], [-10.193, 5.542, 2.279], [-4.403, -12.914, 6.143], [14.754, 3.132, 11.395], [12.262, 13.196, 0.966], [-10.833, 11.684, 1.138], [-13.114, 7.356, 5.406], [-10.794, -8.749, 6.53], [-10.087, 1.452, 9.624], [2.737, -14.33, 14.044], [2.75, 12.516, 5.136], [12.768, -14.704, 19.5], [-3.489, 13.326, 5.043], [-3.161, -1.824, 14.43], [-10.682, -14.043, 0.114], [-10.614, -14.707, 19.731], [-9.814, 4.587, 15.603], [8.893, 5.785, 7.427], [-12.966, 2.919, 11.679], [2.795, -9.207, 5.239], [6.629, 3.818, 19.531], [-1.25, 1.623, 16.939], [11.105, 2.908, 19.757], [10.548, -7.553, 3.909], [8.366, -7.348, 3.145], [0.855, 4.841, 2.629], [9.193, 7.117, 6.771], [8.448, 6.885, 16.298], [-5.693, 8.888, 7.461], [1.217, 4.716, 15.771], [-7.388, -1.538, 9.397], [-10.354, 8.725, 13.917], [-5.312, -4.676, 5.308], [7.671, 10.136, 13.607], [-10.517, 10.087, 9.075], [-4.056, -0.397, 3.467], [13.497, -11.186, 15.471], [-1.628, -2.754, 7.458], [-13.451, 4.188, 19.734], [15.313, -10.031, 11.85], [2.889, -9.863, 16.588], [10.813, -8.907, 18.987], [3.899, 5.353, 1.48], [-4.824, -0.812, 8.989], [-8.712, -11.726, 10.069], [10.659, 7.585, 19.053], [2.997, -6.377, 18.771], [-11.382, 3.755, 3.217], [8.489, -15.025, 12.425], [2.579, 15.274, 18.195], [-15.473, 8.011, 8.689], [-11.183, -12.143, 8.862], [-7.099, 15.031, 4.44], [1.08, 5.873, 7.314], [2.8, -6.948, 17.802], [-7.632, -0.627, 3.085], [15.128, 11.605, 12.293], [-2.892, -9.625, 16.157], [-6.892, -0.112, 2.063], [1.451, 8.46, 10.07], [-9.994, -5.256, 11.14], [5.059, -7.096, 5.266], [-4.569, 5.82, 7.144], [8.355,

3.455, 5.79], [-8.847, 9.322, 6.841], [-2.139, -3.25, 19.508], [10.739, -12.795, 12.909], [-9.139, 14.808, 0.448], [-13.765, 3.543, 2.431], [7.132, -7.405, 5.967], [7.832, -10.995, 14.671], [5.458, 13.04, 9.111], [-2.777, -9.16, 0.739], [2.646, -3.017, 19.417], [-4.986, 10.421, 10.388], [-12.242, -8.053, 4.335], [-11.119, 13.065, 14.799], [1.907, -3.198, 4.025], [7.952, -0.222, 14.611], [4.113, 8.248, 9.877], [-4.736, 11.818, 17.56], [-7.837, 6.717, 2.666], [11.703, 9.651, 5.918], [7.621, -1.2, 17.16], [15.042, -2.169, 18.946], [14.997, -9.727, 2.982], [1.563, -14.609, 10.522], [13.472, 10.921, 4.667], [11.677, 6.206, 15.867], [0.389, -13.142, 9.388], [8.408, -14.361, 15.161], [-6.532, 10.1, 1.492], [4.82, 11.172, 7.402], [-10.817, -12.952, 10.846], [2.998, -10.411, 11.719], [8.186, -15.087, 3.9], [13.366, -4.349, 14.328], [1.107, 3.55, 16.162], [-7.228, -1.347, 19.818], [-14.925, 7.238, 9.078], [-4.676, 14.488, 7.999], [4.907, -13.542, 5.672], [-3.194, -0.601, 8.648], [13.613, 14.254, 0.755], [11.101, 2.785, 13.342], [-3.436, -2.026, 19.329], [-12.069, -0.404, 10.785]

Problem instance 6: [-10.707, -13.408, 1.966], [5.156, 7.942, 4.234], [10.236, 12.826, 12.872], [-2.368, -10.515, 19.712], [12.51, 5.51, 14.972], [-14.802, -6.349, 5.411], [-13.1, 15.108, 16.055], [1.613, 4.817, 14.181], [-4.788, -13.81, 9.328], [3.416, 6.054, 2.772], [13.496, 11.759, 10.97], [-15.008, 12.554, 3.239], [12.031, 12.288, 19.512], [-5.342, -1.454, 9.299], [5.823, 14.603, 3.182], [-7.65, -6.053, 13.902], [12.667, 1.594, 10.818], [14.349, -6.985, 5.917], [1.693, 15.194, 6.802], [-3.919, 13.945, 4.677], [8.169, -15.03, 16.245], [-14.805, 12.865, 9.539], [-4.55, -15.356, 12.376], [-0.39, -3.382, 6.3], [6.505, -2.566, 3.789], [-10.503, -9.16, 17.436], [-10.377, 8.014, 10.374], [-8.178, 3.327, 0.395], [12.483, -10.729, 2.688], [10.592, -3.903, 0.645], [-4.605, 9.355, 16.618], [0.512, 7.403, 10.454], [13.968, 8.388, 3.913], [-8.395, -0.387, 19.209], [13.792, 4.215, 3.14], [11.72, 8.28, 5.8], [-1.408, 13.063, 7.531], [-15.314, -14.666, 5.591], [-0.843, 11.648, 16.624], [1.943, -8.469, 18.796], [-8.784, -3.696, 11.865], [3.443, 8.47, 9.848], [9.272, -14.071, 1.76], [12.733, 7.818, 9.799], [-2.515, 5.154, 18.906], [14.894, -3.033, 14.223], [-14.386, 9.895, 12.091], [2.256, -13.85, 16.316], [-6.987, -11.848, 15.376], [-4.048, -13.489, 6.761], [10.352, 9.729, 4.483], [8.465, -2.972, 1.412], [-9.349, 3.455, 6.294], [-9.775, -7.055, 6.063], [-8.514, -8.497, 8.031], [14.105, 10.497, 4.707], [-12.091, 11.528, 2.159], [0.533, 5.154, 6.168], [7.141,

-13.663, 9.593], [-3.986, 1.334, 9.414], [-14.416, 5.058, 17.297], [-8.343, -8.696, 3.429], [-1.971, 6.691, 3.581], [3.361, 6.658, 4.192], [-11.136, -9.428, 5.409], [11.278, 1.019, 3.246], [-1.9, 3.701, 3.888], [8.033, 8.569, 19.334], [4.793, -12.922, 14.939], [-13.404, 4.086, 2.889], [4.194, -13.072, 5.971], [-7.744, -10.616, 3.701], [13.421, 4.597, 2.687], [2.76, 7.239, 17.596], [-9.507, 4.189, 17.955], [-2.049, 2.707, 1.778], [9.255, 4.978, 18.881], [-6.61, -12.603, 6.56], [-3.049, -9.036, 4.769], [-4.15, -11.579, 7.711], [-1.382, -1.619, 5.145], [3.863, -10.741, 6.255], [3.193, -5.613, 9.054], [-10.592, 5.062, 15.909], [11.899, -15.362, 19.449], [0.484, 15.076, 0.116], [12.228, 1.665, 10.413], [13.842, -6.372, 16.993], [4.164, -7.947, 2.031], [3.551, -9.848, 1.243], [-0.42, 0.137, 12.293], [-8.836, 8.085, 0.9], [-4.666, 15.054, 10.62], [-8.091, -2.603, 10.763], [-6.28, -4.723, 9.688], [4.702, -5.957, 7.984], [-10.868, 2.285, 15.639], [9.72, 5.989, 16.731], [-12.945, 5.85, 11.334], [-15.03, 6.777, 13.909], [6.625, 0.866, 12.438], [8.866, -0.531, 3.432], [7.563, 11.388, 1.573], [-10.382, -11.164, 19.321], [9.541, 15.164, 17.983], [-6.891, 3.43, 14.268], [10.797, -9.26, 1.665], [12.078, -1.261, 14.289], [12.673, -6.788, 12.169], [-13.333, 4.687, 17.765], [3.174, 12.256, 5.527], [-0.76, -3.038, 4.778], [11.655, -2.797, 16.521], [-2.317, -7.101, 11.492], [-14.99, 12.957, 5.049], [-9.016, 14.973, 16.473], [5.684, -8.615, 18.798], [-12.323, 5.621, 0.093], [-7.706, -8.012, 7.921], [-1.109, -10.101, 14.262], [-10.567, 12.292, 5.304], [-5.073, 11.267, 16.936], [-6.468, -5.511, 9.093], [-15.159, 8.753, 10.653], [8.085, 12.224, 14.299], [4.271, 3.65, 8.281], [0.68, -5.641, 14.837], [-10.779, -6.435, 17.859], [6.876, 11.587, 15.944], [-8.098, -9.68, 10.014], [7.781, 12.958, 3.576], [-15.12, 14.006, 4.602], [9.379, 2.739, 18.546], [8.403, 15.148, 13.022], [-1.976, 9.027, 0.726], [-13.201, 13.574, 1.224], [-5.371, 6.091, 7.88], [-2.807, -13.98, 0.884], [11.779, -14.519, 1.572], [-13.551, -13.885, 12.107], [14.588, -0.434, 6.048], [11.009, -1.138, 0.289], [-11.735, -11.031, 14.119], [2.499, 7.122, 18.695], [-14.768, 9.038, 9.086], [5.657, 15.456, 17.889], [2.998, -15.059, 14.555], [13.135, -10.544, 8.396], [1.317, 5.963, 2.352], [11.396, -0.805, 15.342]

Problem instance 7: [-3.646, -14.368, 4.635], [6.509, 1.36, 18.569], [13.586, 10.406, 19.517], [11.755, 0.079, 19.447], [-8.225, -4.619, 2.273], [14.922, -7.893, 8.534], [9.562, 10.308,

3.19], [-12.303, -11.925, 15.768], [-12.896, -10.289, 18.658], [-1.849, -10.237, 8.68], [10.585, -11.032, 5.384], [-1.542, -7.579, 14.327], [-11.595, 14.65, 15.098], [3.121, -12.118, 19.668], [-13.524, 14.219, 17.34], [14.277, 0.892, 0.627], [-1.849, 0.034, 9.008], [0.742, -12.11, 17.036], [-13.554, 9.448, 14.227], [-13.089, 5.888, 4.151], [9.137, 10.632, 11.503], [3.277, -1.614, 12.834], [-2.796, -1.184, 5.835], [-5.358, -7.736, 6.496], [-14.089, -6.579, 15.109], [-2.495, 5.429, 11.053], [-13.561, -3.952, 11.29], [-12.618, 5.098, 2.679], [1.901, 4.929, 12.451], [-12.407, 0.411, 2.564], [-6.906, -11.641, 7.561], [-11.807, -12.011, 1.66], [-13.72, 3.625, 19.394], [-12.431, -4.925, 9.946], [13.899, 9.665, 9.297], [2.216, 1.779, 7.677], [-8.831, -12.741, 5.582], [-13.605, 5.215, 11.832], [11.757, 6.119, 12.035], [-6.695, 2.194, 13.452], [-12.973, 14.334, 9.623], [15.12, 12.408, 2.969], [-10.186, -3.324, 5.054], [14.02, -3.203, 1.462], [-4.726, -4.192, 13.381], [11.97, 9.226, 3.794], [5.912, -7.58, 0.614], [-0.289, -9.736, 10.23], [7.672, 13.78, 7.98], [13.967, 10.129, 11.309], [3.041, 13.438, 8.327], [14.89, 2.285, 11.534], [10.501, 15.255, 5.395], [-11.125, -10.936, 10.293], [-12.811, -0.185, 0.603], [-0.272, 5.624, 16.168], [-14.912, -14.636, 15.523], [-2.745, -11.954, 19.42], [-10.415, 0.822, 3.849], [-4.973, -14.117, 19.926], [-9.746, -13.544, 17.053], [11.119, -10.12, 1.379], [6.184, -10.485, 2.459], [-14.184, -15.372, 16.344], [-8.198, 6.345, 11.138], [4.023, -9.653, 3.575], [-14.272, 9.652, 19.488], [15.132, 9.12, 10.205], [-7.028, -6.384, 14.014], [-7.935, 4.412, 10.237], [5.659, 2.204, 6.112], [4.834, 0.388, 18.528], [-4.121, -11.826, 19.951], [1.87, 12.644, 6.689], [-4.352, -10.223, 5.632], [-14.752, -7.292, 14.221], [6.556, 12.813, 10.504], [-10.562, -3.001, 16.859], [-14.602, 6.617, 19.878], [0.003, -5.519, 15.211], [-7.237, 12.587, 5.091], [8.429, -2.629, 4.675], [-12.634, 4.642, 18.376], [14.369, -8.811, 17.778], [0.293, 6.207, 14.756], [3.939, 5.839, 7.644], [2.111, -0.268, 19.036], [-3.485, -13.122, 13.515], [-9.921, 14.139, 10.855], [-11.668, 15.468, 0.962], [1.076, 0.318, 15.494], [-4.913, 15.003, 15.559], [8.854, 7.733, 18.972], [5.733, 5.77, 0.267], [-14.56, 7.717, 8.732], [-2.462, -6.301, 8.299], [6.27, -0.48, 1.421], [-8.961, -2.117, 7.253], [13.984, -6.188, 3.557], [-5.492, 8.558, 8.499], [-5.051, 2.785, 17.239], [5.219, -3.239, 0.304], [-6.227, 9.068, 1.424], [4.851, -2.219, 6.989], [-11.939, -4.193, 19.927], [14.324, -10.403, 5.633], [-10.789, 7.611, 4.248], [5.105, -7.7, 1.457], [9.183, 5.313, 11.731], [-3.031, -0.073, 7.793], [11.511, 6.692, 13.015], [-12.19, -9.829, 1.489], [14.783, -0.352, 8.973], [3.096,

15.139, 4.893], [-3.369, 10.873, 5.043], [-3.364, -13.616, 1.423], [10.929, 10.589, 3.054], [2.904, -10.138, 5.449], [14.496, 9.214, 12.46], [11.291, -13.528, 8.143], [13.761, 8.723, 15.737], [6.849, 7.161, 9.988], [-10.279, -12.401, 6.724], [10.472, 14.285, 3.522], [14.964, -2.947, 4.147], [-13.786, 14.103, 11.75], [1.829, -2.533, 12.113], [-5.543, 10.727, 13.664], [14.72, 3.798, 16.85], [12.848, -9.08, 8.524], [3.216, -6.523, 10.206], [-3.382, -3.997, 8.661], [-12.041, -1.064, 15.915], [-0.35, 5.712, 7.221], [6.823, 2.277, 19.818], [7.645, -6.7, 7.036], [3.717, 9.66, 8.596], [-0.444, 12.458, 3.337], [4.336, 8.774, 0.445], [-1.438, 7.707, 9.044], [6.689, 10.233, 9.838], [3.41, 6.055, 14.035], [11.587, 8.64, 12.742], [-10.641, 4.805, 16.464], [9.955, -1.662, 11.475], [-2.534, 10.545, 13.155], [7.645, -2.679, 0.491], [8.96, 14.387, 17.024], [-7.786, 0.971, 9.698], [5.004, -0.12, 18.793]

Problem instance 8: [-11.78, 9.353, 12.086], [-7.317, 14.76, 11.329], [4.225, 9.005, 18.516], [8.027, -3.222, 4.929], [3.995, -2.364, 14.808], [0.334, 3.914, 0.342], [-3.865, 3.09, 10.204], [5.548, -10.834, 1.415], [0.802, -12.116, 12.247], [-0.586, -0.525, 17.919], [-10.949, -3.989, 17.803], [-4.856, -8.798, 9.652], [-5.272, 15.175, 3.955], [-0.697, 10.839, 0.322], [3.205, 10.085, 2.191], [3.037, 6.326, 18.785], [-11.837, 14.69, 2.62], [12.257, -7.045, 4.381], [-3.065, 4.443, 14.071], [-2.437, -6.03, 13.04], [6.789, 6.892, 7.315], [3.552, -3.035, 9.32], [5.31, 10.083, 19.273], [11.784, 13.034, 19.853], [-11.25, 14.221, 0.627], [8.664, 14.987, 11.567], [4.677, 7.42, 8.756], [-10.612, -14.627, 9.424], [2.633, 6.517, 16.464], [4.404, -6.381, 5.685], [14.79, -1.699, 15.444], [2.595, -11.943, 17.785], [4.324, 7.441, 3.866], [9.408, 10.838, 5.23], [-2.123, -9.055, 0.874], [7.476, 2.235, 2.069], [5.011, -10.187, 13.46], [-11.735, 10.396, 8.279], [-9.153, 5.984, 17.283], [4.006, -7.273, 12.28], [11.261, -15.433, 7.165], [-4.522, -0.066, 4.041], [11.88, -10.576, 19.606], [-8.639, 11.444, 15.425], [-2.879, 3.981, 8.112], [5.356, 13.863, 2.301], [14.717, -13.379, 13.755], [8.856, -2.021, 6.279], [10.331, 3.819, 4.205], [7.294, 11.728, 1.349], [-2.288, 11.374, 15.003], [8.529, -0.603, 7.103], [7.637, -8.752, 17.926], [14.36, 3.597, 16.498], [9.809, -1.022, 9.684], [-2.579, -2.313, 8.728], [11.066, -15.245, 10.581], [-4.582, -9.998, 17.145], [1.478, 6.112, 0.073], [-5.821, -13.55, 6.492], [11.182, 4.414, 18.383], [6.016, -8.826, 2.801], [-1.394, 9.337, 10.501], [-1.398, 8.839, 15.083], [8.355, -10.81, 3.309], [0.77, 14.055, 9.972], [7.462, -11.743,

3.477], [2.107, -1.159, 8.82], [-1.809, 13.514, 16.004], [8.34, -9.598, 7.504], [-2.848, -10.975, 11.899], [-2, -12.746, 15.753], [-0.138, 1.785, 15.077], [11.578, -10.762, 4.088], [-13.849, -9.34, 11.06], [-9.504, 9.373, 12.387], [-14.623, -5.183, 1.064], [8.767, -10.905, 5.988], [-5.426, -6.128, 5.041], [3.136, -6.173, 16.465], [-8.052, -12.343, 14.764], [-13.101, -8.902, 0.035], [14.724, -10.765, 17.477], [-14.446, 6.668, 4.265], [-10.465, 1.334, 7.696], [-2.369, 3.75, 12.492], [13.958, 7.437, 2.599], [10.815, 7.821, 10.145], [-2.364, 5.954, 10.502], [-7.984, 9.559, 10.156], [-13.488, -0.559, 9.59], [-1.904, -5.721, 8.47], [-12.432, 14.71, 19.433], [2.151, 14.999, 13.706], [-1.859, 12.283, 10.896], [-1.535, 4.795, 11.337], [15.141, 2.336, 5.267], [-13.678, -15.189, 6.889], [-8.777, -12.928, 16.739], [2.419, -5.202, 9.632], [-1.488, 3.613, 13.278], [-0.476, 4.588, 1.309], [-13.643, 4.935, 5.468], [-1.989, -13.26, 0.069], [-9.665, -8.619, 12.278], [12.36, 14.104, 6.986], [9.033, -4.34, 17.899], [-8.708, -8.985, 15.622], [13.019, 11.798, 12.19], [-2.221, -8.976, 9.136], [-2.736, 3.429, 2.135], [-13.682, -14.144, 10.505], [3.715, -13.009, 10.558], [14.038, 7.723, 4.088], [0.602, 6.803, 14.513], [-5.526, -8.039, 4.853], [5.28, 14.729, 2.561], [-3.858, 6.441, 1.532], [5.872, -3.278, 3.701], [8.316, -8.681, 0.74], [9.664, 7.961, 12.645], [-10.422, -10.036, 7.073], [11.452, -4.723, 4.639], [13.199, 13.782, 17.894], [-1.635, 2.984, 7.898], [3.312, -15.026, 9.254], [-11.572, -10.731, 14.353], [0.578, -2.077, 12.75], [-6.423, 13.937, 16.167], [-0.922, 0.293, 14.104], [-9.325, -3.515, 16.096], [15.152, 0.646, 11.409], [-3.127, 6.135, 6.437], [7.884, -14.997, 14.245], [4.796, 1.553, 16.201], [-13.294, 2.333, 16.889], [-2.434, 11.763, 10.128], [11.318, -14.345, 0.88], [-15.172, -9.919, 12.744], [11.918, -8.25, 17.552], [14.349, 11.592, 12.354], [4.815, -12.677, 9.992], [-12.752, -10.945, 3.73], [5.115, -3.597, 1.109], [-7.424, -0.133, 7.325], [2.903, 9.279, 2.225], [-15.235, 2.139, 0.031], [-0.406, 0.832, 10.246], [-6.361, -2.094, 0.603], [3.224, -9.177, 8.767]

Problem instance 9: [-4.004, -15.143, 0.69], [-4.13, 5.359, 19.162], [-9.139, -12.112, 4.057], [3.048, -4.969, 5.559], [8.793, -11.936, 0.4], [-11.539, -5.377, 7.635], [-6.493, -9.254, 0.179], [-0.922, 3.937, 9.586], [-9.52, -14.008, 19.424], [-9.02, 15.011, 4.893], [-5.597, -5.74, 5.256], [4.529, -2.76, 8.19], [4.313, 2.591, 4.807], [6.308, 5.053, 4.96], [-12.858, -12.823, 14.013],

[15.171, 6.737, 13.133], [-13.896, 13.696, 1.832], [12.638, 4.099, 15.116], [-14.709, 8.472, 0.088], [-0.009, -7.441, 1.035], [-3.623, -1.304, 1.893], [-11.669, 7.667, 19.831], [-0.693, -5.248, 15.42], [0.603, -6.756, 18.417], [5.256, -7.725, 15.359], [-5.081, -11.531, 18.732], [-5.115, -9.039, 16.716], [2.3, -6.565, 11.914], [-6.685, -8.198, 3.197], [-12.7, 14.059, 3.384], [-11.556, -12.212, 0.219], [-12.14, -2.418, 11.47], [-0.56, 13.643, 11.754], [6.057, 3.308, 10.66], [12.401, -8.592, 9.828], [-12.356, -7.22, 14.069], [-5.009, 11.628, 5.121], [6.489, 3.591, 13.972], [-2.781, -4.088, 17.73], [4.356, -11.005, 1.335], [-12.475, -10.317, 1.786], [13.935, 12.899, 16.882], [-14.521, 5.063, 2.787], [-7.317, 10.753, 11.437], [7.456, -2.33, 15.754], [-9.532, 6.693, 13.636], [7.03, -13.713, 3.785], [13.747, 4.416, 2.262], [9.413, -5.458, 5.49], [15.099, 6.649, 15.51], [-8.322, -8.789, 16.576], [-5.174, -5.993, 7.399], [13.453, -9.474, 3.743], [5.462, -13.38, 7.916], [-4.073, -9.644, 7.91], [-12.085, -7.904, 16.422], [3.925, 4.955, 19.897], [-0.688, 10.277, 6.979], [15.328, 12.082, 13.986], [9.556, 6.059, 1.499], [-1.37, -5.004, 5.547], [7.781, -15.395, 3.049], [-2.021, 10.79, 19.706], [-8.122, 14.805, 17.996], [11.064, -9.507, 5.043], [8.044, -8.42, 7.228], [-8.789, 10.162, 4.438], [1.624, -4.096, 1.035], [-9.959, -2.036, 14.363], [14.258, -12.648, 5.555], [-2.822, -7.934, 18.936], [1.274, 8.339, 14.24], [7.64, -7.243, 11.418], [-8.235, 15.08, 6.884], [-14.453, -0.375, 13.319], [2.103, -12.162, 12.055], [-13.246, 5.919, 12.906], [0.208, 9.049, 3.572], [-13.901, 3.72, 11.309], [2.345, -10.862, 18.399], [1.904, -5.578, 14.126], [2.422, -1.653, 13.766], [2.276, 3.956, 19.534], [-10.675, -14.517, 11.629], [-7.902, -12.143, 11.749], [8.188, 15.08, 6.082], [-1.643, -13.768, 5.793], [4.633, 6.735, 15.332], [-10.112, 9.855, 4.409], [-2.731, -10.581, 11.334], [-14.715, -3.41, 1.286], [-13.875, 2.75, 5.513], [8.342, 10.631, 3.302], [0.936, 13.122, 10.743], [0.997, 13.687, 8.916], [6.474, 8.55, 9.21], [-1.323, -1.002, 10.918], [13.701, -3.358, 10.817], [-3.887, -7.32, 9.14], [6.169, 2.967, 17.3], [-1.186, -1.789, 0.333], [12.026, 5.661, 16.782], [1.974, -6.516, 14.945], [7.779, -14.16, 12.308], [7.591, -10.899, 13.62], [10.106, -7.76, 0.141], [7.87, 1.093, 1.005], [2.65, 2.507, 16.39], [14.32, -8.41, 12.471], [-6.435, 9.321, 5.909], [6.075, -4.382, 12.319], [3.006, 5.957, 1.144], [4.198, -11.227, 10.929], [-5.431, 3.433, 10.117], [7.064, -13.22, 14.913], [2.911, -4.52, 13.563], [-10.546, -2.925, 15.369], [14.96, 1.686, 1.098], [-9.209, 15.347, 19.163], [10.967, -5.124, 12.687], [15.069, -9.256, 2.526], [-5.149, 4.56, 3.464], [6.733, 5.199,

1.569], [-1.694, 7.352, 7.561], [8.654, -8.144, 0.718], [-14.225, -6.452, 2.36], [-14.212, -9.885, 16.284], [-11.8, -14.754, 19.089], [4.135, 13.362, 9.778], [-13.81, 13.724, 17.379], [11.399, -10.114, 5.821], [7.414, -0.592, 9.748], [-11.512, -13.304, 9.083], [-2.099, -3.264, 5.057], [-8.46, 3.768, 7.315], [7.039, 3.773, 9.709], [-12.709, 4.065, 4.931], [-8.714, 2.087, 14.748], [7.127, 14.146, 6.448], [-12.212, -11.446, 4.344], [-4.935, 12.447, 18.205], [-3.283, -8.586, 18.836], [11.756, 7.65, 14.736], [13.224, -11.031, 15.312], [-3.632, -11.551, 3.96], [1.702, -1.107, 14.978], [-13.457, 11.004, 17.333], [10.994, 14.816, 1.855], [-12.049, 4.658, 0.898], [7.178, 2.866, 15.362]

Problem instance 10: [3.605, 9.695, 2.432], [-1.57, -0.918, 8.911], [10.342, 10.799, 4.89], [5.996, -6.848, 19.765], [0.299, 2.073, 11.362], [6.703, 0.633, 13.669], [-13.627, 8.148, 16.373], [4.627, 9.358, 11.378], [-5.41, -13.116, 8.271], [14.164, 6.332, 19.93], [-8.783, 12.675, 0.389], [-1.513, -5.394, 15.35], [2.371, -9.944, 7.338], [-1.43, 13.44, 7.232], [-2.954, 7.878, 18.935], [-13.74, -15, 16.754], [-5.683, 3.768, 13.13], [-5.861, -6.914, 4.36], [7.511, -2.201, 16.088], [-12.995, -11.542, 16.385], [-12.457, -5.341, 8.741], [3.631, -1.811, 5.719], [-8.102, 10.79, 17.658], [1.968, 2.204, 6.896], [13.432, -0.941, 6.717], [7.806, -2.423, 0.201], [2.615, 5.282, 6.454], [-2.728, -14.713, 3.289], [-7.485, -1.571, 1.998], [15.162, -8.793, 14.343], [13.178, 4.859, 7.084], [11.99, 10.412, 12.915], [-6.774, -11.886, 1.272], [6.271, -2.437, 3.344], [7.537, 0.16, 7.431], [14.166, 13.096, 1.958], [-7.857, 5.588, 9.428], [-9.085, 0.208, 3.811], [-10.93, 10.407, 4.69], [-6.031, -7.324, 8.286], [5.375, -3.916, 16.521], [-15.01, -12.754, 3.542], [3.477, -6.296, 19.301], [-8.428, -11.751, 5.814], [-3.049, 5.661, 10.979], [2.933, -9.666, 15.315], [14.24, -1.506, 16.199], [6.203, 10.103, 19.338], [11.786, 0.637, 7.279], [-12.559, 4.549, 17.83], [11.37, 1.442, 2.509], [5.919, 2.474, 0.858], [-7.118, -8.211, 16.202], [-11.673, 5.369, 15.356], [7.388, 0.724, 11.41], [2.469, 0.188, 12.522], [6.035, 11.377, 4.648], [-13.833, -4.492, 2.806], [-10.558, -8.456, 4.737], [-11.221, 3.446, 3.119], [-7.857, 13.095, 18.221], [5.441, -4.984, 3.385], [1.945, -14.16, 6.345], [-4.045, -11.809, 12.14], [10.714, 6.97, 14.506], [10.897, -13.411, 0.805], [-12.473, -3.225, 8.057], [9.33, -2.243, 10.831], [8.59, 4.71, 16.897], [14.089, 9.09, 14.092], [-1.16, 12.91, 9.031], [0.85, -0.784, 3.754], [8.128, 1.242, 18.87], [6.916, -6.666, 17.664], [-6.451, -14.439,

0.424], [-13.129, -2.095, 8.619], [-1.246, -3.597, 0.053], [8.113, 0.677, 9.238], [-12.901, 15.066, 1.187], [13.731, 2.573, 1.46], [-0.325, 10.077, 7.891], [-13.301, -9.141, 6.031], [4.65, -4.98, 5.626], [9.362, 9.303, 4.296], [6.669, 7.984, 16.725], [-4.337, -12.261, 15.248], [-0.943, -5.994, 5.555], [1.194, 7.519, 3.227], [9.929, 8.14, 1.64], [10.951, 13.414, 11.063], [-3.123, 12.512, 4.098], [12.828, -8.518, 17.282], [-1.008, -6.533, 13.838], [-7.907, 13.781, 12.18], [5.163, -11.425, 14.415], [-12.455, -12.173, 18.697], [10.714, -6.33, 7.536], [0.797, 8.635, 3.13], [0.005, -0.495, 7.353], [-8.551, 10.674, 9.055], [-2.587, 4.995, 16.45], [-5.003, -10.969, 15.026], [-12.695, 2.52, 13.869], [-7.795, -5.404, 17.245], [-13.966, 5.31, 9.002], [2.643, -4.277, 0.324], [-2.772, -9.919, 19.792], [-5.677, 10.71, 7.615], [4.958, -12.271, 0.227], [0.906, -11.457, 11.586], [2.445, -1.628, 1.074], [-9.622, -12.796, 7.396], [0.125, 10.286, 14.835], [14.717, -6.742, 11.786], [-0.935, -13.962, 3.858], [-11.486, 9.263, 7.51], [-12.149, 5.102, 9.823], [-2.891, -3.103, 12.567], [-8.478, -9.267, 3.68], [11.471, 7.357, 1.342], [-9.193, -14.589, 9.778], [-6.669, -7.988, 13.524], [-1.089, 8.06, 16.502], [-4.286, -6.674, 10.088], [15.091, -4.716, 19.501], [-13.819, -3.856, 6.363], [-12.895, 0.362, 7.869], [2.434, -14.178, 0.968], [1.271, -1.926, 7.041], [12.709, -10.735, 4.507], [-7.397, 10.71, 10.304], [-8.104, 13.739, 2.959], [4.115, -9.475, 9.659], [8.167, 3.816, 7.838], [7.538, -5.369, 5.861], [-10.606, -5.327, 1.858], [-4.223, 4.427, 10.972], [6.663, -1.606, 4.331], [13.617, -6.615, 15.503], [-9.433, -1.214, 19.598], [-9.697, -9.519, 8.547], [-13.121, -3.302, 17.931], [5, -1.891, 6.831], [-3.692, -9.71, 7.653], [-3.738, -12.299, 4.084], [-0.724, -15.451, 15.447], [-6.591, -12.616, 0.191], [-0.86, 4.91, 18.39], [3.971, -7.836, 10.439], [3.126, 4.349, 3.288]

Bibliography

Abraham, J. (1979). An improved algorithm for network reliability. *IEEE Transactions on Reliability*, R-28(1):58–61.

Al-Turjman, F. M., Hassanein, H. S., and Ibnkahla, M. A. (2013). Efficient deployment of wireless sensor networks targeting environment monitoring applications. *Computer Communications*, 36(2):135–148.

Amaldi, E., Capone, A., Cesana, M., Filippini, I., and Malucelli, F. (2008). Optimization models and methods for planning wireless mesh networks. *Computer Networks*, 52(11):2159–2171.

Aneja, Y. P. and Nair, K. P. K. (1978). The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25(3):549–555.

Avella, P., Boccia, M., and Sforza, A. (2002). A penalty function heuristic for the resource constrained shortest path problem. *European Journal of Operational Research*, 142(2):221–230.

Ball, M. (1979). Computing network reliability. *Operations Research*, 27(4):823–838.

Ball, M. and Nemhauser, G. (1979). Matroids and a reliability analysis problem. *Mathematics of Operations Research*, 4(2):132–143.

Ball, M. and Provan, J. (1982). Bounds on the reliability polynomial for shellable independence systems. *SIAM Journal on Algebraic and Discrete Methods*, 3(2):166–181.

Ball, M. and Van Slyke, R. (1977). Backtracking algorithms for network reliability analysis. *Studies in Integer Programming*, 1:49–64.

Ball, M. O. (1986). Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliability*, 35(3):230–239.

Barnhart, C., Boland, N., Clarke, L., Johnson, E., Nemhauser, G., and Shenoi, R. (1998). Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220.

Beasley, J. E. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394.

Beichelt, F. and Spross, L. (1987). An improved abraham-method for generating disjoint sums. *IEEE Transactions on Reliability*, R-36(1):70–74.

Benyamina, D., Hafid, A., and Gendreau, M. (2009a). Optimal placement of gateways in multi-hop wireless mesh networks: A clustering-based approach. In *IEEE 34th Conference on Local Computer Networks, 2009*, pages 625–632.

Benyamina, D., Hafid, A., and Gendreau, M. (2009b). Wireless mesh network planning: A multi-objective optimization approach. In *IEEE 5th International Conference on Broadband Communications, Networks and Systems (BROADNETS), 2008*, pages 602–609.

Benyamina, D., Hafid, A., Gendreau, M., and Maureira, J. (2011). On the design of reliable wireless mesh network infrastructure with QoS constraints. *Computer Networks*, 55(8):1631–1647.

Birkos, K., Politis, I., Lykourgiotis, A., Kordelas, T., Tselios, C., Mplatsas, M., Dagiuklas, T., Albano, M., Rodriguez, J., Ciftci, S., Pelvan, S., Cimen, E., and Akman, A. (2012). Towards 3d video delivery over heterogeneous networks: The romeo approach. In *Telecommunications and Multimedia (TEMU), 2012 International Conference on*, pages 60–65.

Bondy, J. and Murty, U. (2008). *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer.

Bredin, J., Demaine, E., Hajiaghayi, M., and Rus, D. (2010). Deploying sensor networks with guaranteed fault tolerance. *IEEE/ACM Transactions on Networking (TON)*, 18(1):216–228.

Bruno, R., Conti, M., and Gregori, E. (2005). Mesh networks: commodity multihop ad hoc networks. *IEEE Communications Magazine*, 43(3):123–131.

Camp, J., Robinson, J., Steger, C., and Knightly, E. (2006). Measurement driven deployment of a two-tier urban mesh access network. In *Proceedings of the 4th international conference on Mobile systems, applications and services (MobiSys '06)*, pages 96–109, New York, NY, USA.

Cancela, H. and El Khadiri, M. (1995). A recursive variance-reduction algorithm for estimating communication-network reliability. *IEEE Transactions on Reliability*, 44(4):595–602.

Cancela, H. and El Khadiri, M. (1998). Series-parallel reductions in monte carlo network-reliability evaluation. *IEEE Transactions on Reliability*, 47(2):159–164.

Cancela, H. and El Khadiri, M. (2003). The recursive variance-reduction simulation algorithm for network reliability evaluation. *IEEE Transactions on Reliability*, 52(2):207–212.

Capone, A., Cesana, M., Donno, D. D., and Filippini, I. (2010). Deploying multiple interconnected gateways in heterogeneous wireless sensor networks: An optimization approach. *Computer Communications*, 33(10):1151–1161.

Carlyle, W. M., Royset, J. O., and Kevin Wood, R. (2008). Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 52(4):256–270.

Chen, H., Wu, H., Kumar, S., and Tzeng, N.-F. (2007). Minimum-cost data delivery in heterogeneous wireless networks. *IEEE Transactions on Vehicular Technology*, 56(6):3511–3523.

Choi, J., Woo, S., and Shim, B. (2011). Reliable service provisioning in converged multimedia network environment. *Journal of Network and Computer Applications*, 34(1):394–401.

Colbourn, C. J. and Harms, D. D. (1988). Bounding all-terminal reliability in computer networks. *Networks*, 18(1):1–12.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deeter, D. and Smith, A. E. (1998). Economic design of reliable networks. *IIE Transactions*, 30(12):1161–1174.

deMercado, J., Spyratos, N., and Bowen, B. (1976). A method for calculation of network reliability. *IEEE Transactions on Reliability*, R-25(2):71–76.

Dengiz, B., Altiparmak, F., and Smith, A. E. (1997). Efficient optimization of all-terminal reliable networks, using an evolutionary approach. *IEEE Transactions on Reliability*, 46(1):18–26.

Dengiz, B., Altiparmak, F., and Smith, A. E. (2002). Local search genetic algorithm for optimal design of reliable networks. *IEEE Transactions on Evolutionary Computation*, 1(3):179–188.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Dilmaghani, R. and Rao, R. (2007). Future Wireless Communication Infrastructure with Application to Emergency Scenarios. In *Proceedings of IEEE International Symposium*

on a World of Wireless, Mobile and Multimedia Networks (WOWMOM), 2007, Helsinki, Finland.

Dréo, J., Pétrowski, A., Siarry, P., and Taillard, E. (2006). *Metaheuristics for Hard Optimization*. Springer, Berlin.

Dumitrescu, I. and Boland, N. (2001). Algorithms for the weight constrained shortest path problem. *International Transactions in Operational Research*, 8(1):15–29.

Eppstein, D. (1998). Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Gastpar, M. and Vetterli, M. (2002). On the capacity of wireless networks: the relay case. In *Precedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, volume 3, pages 1577–1586.

Grosan, C., Abraham, A., and Hassainen, A. (2009). Designing resilient networks using multicriteria metaheuristics. *Telecommunication Systems*, 40(1):75–88.

Grossglauser, M. and Tse, D. N. C. (2002). Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 10(4):477–486.

Grover, W. (2004). *Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Prentice Hall, New Jersey.

Guidoni, D., Mini, R., and Loureiro, A. (2010). On the design of resilient heterogeneous wireless sensor networks based on small world concepts. *Computer Networks*, 54(8):1266–1281.

Gupta, P. and Kumar, P. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404.

Han, X., Cao, X., Lloyd, E., and Shen, C. (2010). Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656.

Handler, G. Y. and Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks*, 10(4):293–309.

Harms, D. (1995). *Network Reliability: Experiments with a Symbolic Algebra Environment*. Discrete Mathematics and Its Applications. CRC Press, Boca Raton.

Hartvigsen, D. (1998). The planar multiterminal cut problem. *Discrete Applied Mathematics*, 85(3):203–222.

Hekmat, R. and Van Mieghem, P. (2004). Interference in wireless multi-hop ad-hoc networks and its effect on network capacity. *Wireless Networks*, 10(4):389–399.

Hira, M., Tobagi, F., and Medepalli, K. (2007). Throughput analysis of a path in an IEEE 802.11 multihop wireless network. In *Proceedings of Wireless Communications and Networking Conference (WCNC 2007)*, pages 441–446.

Holmberg, K. and Yuan, D. (2003). A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1):42–57.

Huang, J.-H., Wang, L.-C., and Chang, C.-J. (2008). Throughput-coverage tradeoff in a scalable wireless mesh network. *Journal of Parallel and Distributed Computing*, 68(3):278–290.

Hui, K.-P. (2007). Monte carlo network reliability ranking estimation. *IEEE Transactions on Reliability*, 56(1):50–57.

Hui, K.-P., Bean, N., Kraetzl, M., and Kroese, D. (2003). The tree cut and merge algorithm for estimation of network reliability. *Probability in the Engineering and Informational Sciences*, 17(1):23–45.

Hui, K.-P., Bean, N., Kraetzl, M., and Kroese, D. (2005). The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134(1):101–118.

Hunter, A., Andrews, J., and Weber, S. (2008). Transmission capacity of ad hoc networks with spatial diversity. *IEEE Transactions on Wireless Communications*, 7(12):5058–5071.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, pages 33–65. Springer.

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18(3):391–406.

Jan, R. (1993). Design of reliable networks. *Computers & Operations Research*, 20(1):25–34.

Jun, J. and Sichitiu, M. (2003). The nominal capacity of wireless mesh networks. *IEEE Wireless Communications*, 10(5):8–14.

Karp, R. M. and Luby, M. (1985). Monte-carlo algorithms for the planar multiterminal network reliability problem. *Journal of Complexity*, 1(1):45–64.

Kashyap, A., Khuller, S., and Shayman, M. (2010). Relay placement for fault tolerance in wireless networks in higher dimensions. *Computational Geometry*, 44(4):206–215.

Katoh, N., Ibaraki, T., and Mine, H. (1978). An $O(Kn^2)$ algorithm for k shortest simple paths in an undirected graph with nonnegative arc length. *Transactions of the Institute of Electronics and Communication Engineers of Japan, Section E*, 61:971–972.

Katoh, N., Ibaraki, T., and Mine, H. (1982). An efficient algorithm for k shortest simple paths. *Networks*, 12(4):411–427.

Konak, A. and Bartolacci, M. R. (2007). Designing survivable resilient networks: A stochastic hybrid genetic algorithm approach. *Omega*, 35(6):645–658.

Kroese, D., Hui, K.-P., and Nariai, S. (2007). Network reliability optimization via the cross-entropy method. *IEEE Transactions on Reliability*, 56(2):275–287.

Kubat, P. (1989). Estimation of reliability for communication/computer networks simulation/analytic approach. *IEEE Transactions on Communications*, 37(9):927–933.

Kumamoto, H., Tanaka, K., and Inoue, K. (1977). Efficient evaluation of system reliability by monte carlo method. *IEEE Transactions on Reliability*, R-26(5):311–315.

Li, J., Blake, C., De Couto, D. S., Lee, H. I., and Morris, R. (2001). Capacity of ad hoc wireless networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01)*, pages 61–69.

Lomonosov, M. (1994). On monte carlo estimates in network reliability. *Probability in the Engineering and Informational Sciences*, 8(2):245–264.

Machado, R., Ansari, N., Wang, G., and Tekinay, S. (2010). Adaptive density control in heterogeneous wireless sensor networks with and without power management. *Communications, IET*, 4(7):758–767.

Mehlhorn, K. and Ziegelmann, M. (2000). Resource constrained shortest paths. In Paterson, M., editor, *Lecture Notes in Computer Science: Algorithms-ESA 2000*, volume 1879, pages 326–337. Springer, Berlin.

Misra, S., Hong, S., Xue, G., and Tang, J. (2010). Constrained relay node placement in wireless sensor networks: Formulation and approximations. *IEEE/ACM Transactions on Networking (TON)*, 18(2):434–447.

Moraes, R., Ribeiro, C., and Duhamel, C. (2009). Optimal solutions for fault-tolerant topology control in wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 8(12):5970–5981.

Nel, L. D. and Colbourn, C. J. (1990). Combining monte carlo estimates and bounds for network reliability. *Networks*, 20(3):277–298.

Niyato, D. and Hossain, E. (2009). Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach. *IEEE Transactions on Vehicular Technology*, 58(4):2008–2017.

Pei, X., Jiang, T., Qu, D., Zhu, G., and Liu, J. (2010). Radio-resource management and access-control mechanism based on a novel economic model in heterogeneous wireless networks. *IEEE Transactions on Vehicular Technology*, 59(6):3047–3056.

Provan, J. and Ball, M. (1984). Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516–526.

Qian, Y., Lu, K., Rong, B., and Zhu, H. (2007). Optimal key management for secure and survivable heterogeneous wireless sensor networks. In *IEEE Global Telecommunications Conference (GLOBECOM), 2007*, pages 996–1000.

Ramirez-Marquez, J. E. and Coit, D. W. (2005). A monte-carlo simulation approach for approximating multi-state two-terminal reliability. *Reliability Engineering & System Safety*, 87(2):253–264.

Raniwala, A. and Chiueh, T. (2005). Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, volume 3, pages 2223–2234.

Ribeiro, C. C. and Minoux, M. (1985). A heuristic approach to hard constrained shortest path problems. *Discrete Applied Mathematics*, 10(2):125–137.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170.

Rushdi, A. M. (1984). On reliability evaluation by network decomposition. *IEEE Transactions on Reliability*, R-33(5):379–384.

Shahnaz, A. and Erlebach, T. (2010). Approximating fault-tolerant Steiner subgraphs in heterogeneous wireless networks. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 529–533. ACM.

So, A. and Liang, B. (2009). Optimal placement and channel assignment of relay stations in heterogeneous wireless mesh networks by modified benders decomposition. *Ad Hoc Networks*, 7(1):118–135.

Takagi, H. and Kleinrock, L. (1984). Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257.

Tiwari, R., Mishra, T., Li, Y., and Thai, M. (2007). k-strongly connected m-dominating and absorbing set in wireless ad hoc networks with unidirectional links. In *International Conference on Wireless Algorithms, Systems and Applications (WASA), 2007*, pages 103–112. IEEE.

van der Zijpp, N. and Catalano, S. F. (2005). Path enumeration by finding the constrained k-shortest paths. *Transportation Research Part B: Methodological*, 39(6):545–563.

Van Slyke, R. and Frank, H. (1971). Network reliability analysis: Part I. *Networks*, 1(3):279–290.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107.

Wang, C., Park, M., Willson, J., Farago, A., and Du, D. (2008). Fault-tolerant dual power management in wireless sensor networks. In *Global Telecommunications Conference (GLOBECOM), 2008*, pages 1–6. IEEE.

Wang, Q., Xu, K., Takahara, G., and Hassanein, H. (2007). Device placement for heterogeneous wireless sensor networks: Minimum cost with lifetime constraints. *IEEE Transactions on Wireless Communications*, 6(7):2444–2453.

Wang, W. and Liu, X. (2006). A framework for maximum capacity in multi-channel multi-radio wireless networks. In *IEEE 3rd Consumer Communications and Networking Conference (CCNC 2006)*, volume 2, pages 720–724.

Warburton, A. (1987). Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79.

Wilkov, R. (1972). Analysis and design of reliable computer networks. *IEEE Transactions on Communications*, 20(3):660–678.

Wollmer, R. (1964). Removing arcs from a network. *Operations Research*, 12(6):934–940.

Wu, W., Luo, J., Yang, M., and Yang, L. (2012). Joint interface placement and channel assignment in multi-channel wireless mesh networks. In *IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pages 395–402.

Wu, X., Liu, J., and Chen, G. (2006). Analysis of bottleneck delay and throughput in wireless mesh networks. In *Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2006)*, pages 765–770.

Xiao, M. (2010). Simple and improved parameterized algorithms for multiterminal cuts. *Theory of Computing Systems*, 46(4):723–736.

Xue, F. and Kumar, P. R. (2004). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks.*, 10(2):169–181.

Yang, D., Misra, S., Fang, X., Xue, G., and Zhang, J. (2010). Two-tiered constrained relay node placement in wireless sensor networks: Efficient approximations. In *7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010*, pages 1–9. IEEE.

Yang, D., Misra, S., and Xue, G. (2009). Joint base station placement and fault-tolerant routing in wireless sensor networks. In *Global Telecommunications Conference (GLOBECOM), 2009*, pages 1–6. IEEE.

Yang, K., Wu, Y., and Chen, H. (2007). QoS-aware routing in emerging heterogeneous wireless networks. *IEEE Communications Magazine*, 45(2):74–80.

Yen, J. (1971). Finding the k-shortest loopless paths in a network. *Management Science*, 17(11):712–716.