

MINIMUM SPANNING TREE PROBLEM WITH CONFLICT CONSTRAINTS

by

Murat Umut İzer

B.S., Industrial Engineering, Gazi University, 2010

M.A., Management Information Systems, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor, Prof. İ. Kuban Altınel, whose guidance, encouragement, and invaluable insights were pivotal throughout this research journey. I am equally grateful to my co-supervisor, Prof. Temel Öncan, whose expert advice and constructive feedback greatly enriched this work. I am honored to have the opportunity to work with them. I would also like to thank the jury members Prof. Selda Küçükçifçi, Prof. Taner Bilgiç, Prof. Z. Caner Taşkın, and Assoc. Prof. Evren Güney for devoting their valuable time to this thesis to provide constructive feedback.

Special thanks to my colleagues and friends, whose support and stimulating discussions provided both inspiration and motivation during challenging times. I would like to thank Prof. Çiğdem Alabaş Uslu for her valuable comments and insights, Assist. Prof. Canan Ağlan Gökler for her continuous motivation and encouragement, Assist. Prof. Havva Serim Toro for not leaving me alone during my most stressful times and cheering me up since highschool, and Assist. Prof. Esra Angın for her company and friendship in all circumstances.

Lastly, I extend my heartfelt thanks to my family, Merih, and Molly for their unwavering support, patience, and understanding throughout this journey. Without their encouragement, this thesis would not have been possible.

ABSTRACT

MINIMUM SPANNING TREE PROBLEM WITH CONFLICT CONSTRAINTS

The minimum spanning tree with conflict constraints (MSTC) problem is a combination of the maximum cardinality stable set and the minimum spanning tree problems, and it is proven to be $\mathcal{NP}$ -hard. This thesis investigates this problem in graphs where certain pairs of edges cannot be included simultaneously due to conflict constraints. Such constraints are crucial for applications in network design, particularly in scenarios where resource limitations, regulatory rules, or physical restrictions prevent specific connections from coexisting.

We propose four distinct branch-and-bound algorithms to effectively navigate the complex solution space of this problem. Each algorithm incorporates innovative strategies to enhance computational efficiency. Experimental results indicate that the novel algorithms are very fast and outperform a widely used commercial solver. Furthermore, this thesis provides a comprehensive analysis of the structural properties of conflict constraints and their impact on the solution space. These findings have broad implications for applications in telecommunications, transportation, and energy distribution networks, where optimization under complex constraints is essential.

By addressing both theoretical and practical challenges, this thesis establishes a foundational framework for computing conflict-free minimum spanning trees. The proposed methodologies not only outperform existing solution procedures but also improve best known solutions and set the stage for future advancements in optimization and network design under disjunctive conditions.

ÖZET

ÇATIŞMA KISITLI ENKÜÇÜK KAPSAR AĞAÇ PROBLEMİ

Çatışma kısıtlı enküçük kapsar ağaç problemi, enbüyük ağırlıklı kapsar küme ile enküçük kapsar ağaç problemlerinin bir birleşimidir ve $\mathcal{NP}$ -zor olduğu kanıtlanmıştır. Bu tez, çatışma kısıtları nedeniyle belirli kenar çiftlerinin aynı anda çözümde yer alamadığı çizgelerde bu problemi ele almaktadır. Çatışma kısıtları özellikle kaynak kısıtlamaları, düzenleyici kurallar veya fiziksel kısıtlamaların belli bağlantıların bir arada var olmasını engellediği senaryolarda, ağ tasarımı uygulamaları için çok önemlidir.

Problemin karmaşık çözüm uzayını etkili bir şekilde taramak için dört farklı dalsınır algoritması geliştirildi. Her algoritma, yenilikçi stratejiler kullanarak hesaplama verimliliğini artırmayı ve kısa sürede etkili çözümler sunmayı hedeflemektedir. Deney-sel sonuçlar geliştirilen algoritmaların çok hızlı olduklarını ve yaygın olarak kullanılan bir ticari yazılımı bu açıdan geride bıraktıklarını söyleyebiliriz. Ayrıca bu tezde çatışma kısıtlarının yapısal özellikleri ve bunların çözüm uzayı üzerindeki etkileri kapsamlı biçimde çözümlenmektedir. Bulgular karmaşık kısıtlamalar altında eniyilemenin önemli olduğu telekomünikasyon, ulaştırma ve enerji dağıtım ağları gibi alanlarda geniş kapsamlı uygulama olanakları sunmaktadır.

Geliştirilen yöntemler ve elde edilen bulgular hem kuramsal çalışmalarda hem de uygulamalarda karşılaşılan zorluklara yanıt vererek çatışma kısıtlı enküçük kapsar ağaç problemleri için sağlam bir temel oluşturmaktadır. Önerilen yöntemler, var olan çözüm yaklaşımlarını aşmakla kalmayıp, bilinen en iyi çözümleri iyileştirerek ayırıcı koşullar altında eniyileme ve ağ tasarımı yapmak için gelecekteki ilerlemelere de zemin hazırlamaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	xiii
LIST OF SYMBOLS	xvi
LIST OF ACRONYMS/ABBREVIATIONS	xx
1. INTRODUCTION	1
2. BASIC CONCEPTS AND NOTATIONS	3
3. RELATED WORKS AND OUR CONTRIBUTIONS	7
3.1. Related Works	7
3.2. Our Contributions	10
4. FORMULATIONS	11
4.1. Minimum Spanning Tree Problem	11
4.1.1. Combinatorial Optimization Formulation	11
4.1.2. Undirected Formulations	12
4.1.2.1. Set Packing Formulation	12
4.1.2.2. Cut-set Formulation	13
4.1.2.3. Martin's Formulation	13
4.1.3. Directed Formulations	14
4.1.3.1. Directed Cut-set Formulation	14
4.1.3.2. Single Commodity Flow Formulation	15
4.1.3.3. Multi-commodity Flow Formulation	16
4.2. Maximum Cardinality and Weight Stable Set Problems	17
4.3. Minimum Spanning Tree Problem with Conflicts	19
4.3.1. Binary Integer Linear Programming Formulations	19
4.3.2. Binary Integer Quadratic Programming Formulations	21
4.3.3. Combinatorial Optimization Formulation	21

4.3.4. Reference Formulation	22
5. MAJOR COMPONENTS OF BRANCH-AND-BOUND ALGORITHMS . .	24
5.1. A Generic Branch-and-Bound Frame	24
5.2. Subproblems and Their Relaxations	26
5.3. Subproblem Selection	27
5.4. Branching Rules	27
5.4.1. Corrective Branching Rules	27
5.4.2. Preventive Branching Rules	29
5.4.3. Illustration of Branching Rules	34
5.5. Filtering	40
5.5.1. Infeasibility Test	40
5.5.2. Simple Lower Bounds	43
5.5.3. Probing	45
5.5.4. Illustration of Filtering Operations	47
6. BRANCH-AND-BOUND ALGORITHMS	50
6.1. Edge Branching-based Branch-and-Bound Algorithm	50
6.2. Diving-based Branch-and-Bound Algorithm	52
6.2.1. Recursive Diving Algorithm	52
6.2.2. Iterative Diving Algorithm	54
6.2.3. Illustration of Diving Algorithms	55
6.3. Stable Set-based Branch-and-Bound Algorithm	57
7. UPDATING SPANNING TREES	59
7.1. Deletion	59
7.2. Finding Replacement Edges	60
7.3. Insertion	65
7.4. Cycle Check	66
7.5. Illustration for Spanning Tree Updates	71
8. COMPUTATIONAL EXPERIMENTS	76
8.1. Implementation Details	76
8.1.1. Data Structures	76
8.1.2. Parameters	79

8.2. Test Environment	80
8.2.1. Test Instances	80
8.2.1.1. Existing Instances	80
8.2.1.2. Random Problems	81
8.2.2. Performance Measures	83
8.3. The Determination of the Reference Formulations	84
8.4. The Effect of Infeasibility Test and Probing	89
8.5. Performance of the Branch-and-Bound Algorithms	98
8.6. Performance of Edge Replacement Algorithms	103
8.7. Performance of Cycle Detection Algorithms	105
8.8. The Effect of Graph and Conflict Densities	107
9. CONCLUSION	110
REFERENCES	112
APPENDIX A: PROBLEM INSTANCES AND DETAILED RESULTS OF FORMULATIONS	120
APPENDIX B: DETAILED RESULTS OF BRANCH-AND-BOUND ALGORITHMS	129

LIST OF FIGURES

Figure 1.1.	Sample illustration of offshore wind farm cable routing without cable crossings.	2
Figure 5.1.	Generic BB Algorithm for MSTC Problem.	25
Figure 5.2.	Heuristic to determine a maximal clique including a given vertex subset.	29
Figure 5.3.	Heuristic to determine a maximal subset of a given maximal stable set.	31
Figure 5.4.	Greedy heuristic to determine a maximal stable set and branching edges.	33
Figure 5.5.	Sample graph.	35
Figure 5.6.	Sample conflict graph.	35
Figure 5.7.	Optimal solution of the MST with violated conflict pairs.	36
Figure 5.8.	Optimal solution of the MSTC instance.	36
Figure 5.9.	BB search tree for the MSTC instance with pair branching rule. .	37
Figure 5.10.	BB search tree for the MSTC instance with edge branching rule. .	38
Figure 5.11.	BB search tree for the MSTC instance with stable-set based branching rule.	39

Figure 5.12.	Infeasibility detection	42
Figure 5.13.	Probing heuristic.	46
Figure 5.14.	BB search tree for the MSTC instance with filtering operations. .	48
Figure 5.15.	Illustration of infeasibility test and probing on the sample graph. .	49
Figure 6.1.	Edge branching based branch-and-bound algorithm (BB-E).	51
Figure 6.2.	Recursive diving algorithm (Rec-D).	53
Figure 6.3.	Diving based branch-and-bound algorithm (BB-D).	54
Figure 6.4.	BB search tree for Rec-D algorithm.	56
Figure 6.5.	Stable set branching based branch-and-bound algorithm (BB-S). .	58
Figure 7.1.	Deletion of an edge from the spanning tree.	59
Figure 7.2.	Finding a replacement edge: breadth-first-search.	61
Figure 7.3.	Finding a replacement edge: predecessor array.	62
Figure 7.4.	Finding a replacement edge: link-cut-trees.	63
Figure 7.5.	Finding a replacement edge: depth-first search.	64
Figure 7.6.	Insertion of a new edge to a spanning tree.	65
Figure 7.7.	Disjoint set functions.	66

Figure 7.8.	Union by rank and path compression in disjoint sets.	67
Figure 7.9.	Cycle detection with disjoint sets.	68
Figure 7.10.	Cycle detection with breadth-first-search.	69
Figure 7.11.	Cycle detection with link-and-cut trees.	70
Figure 7.12.	Cycle detection with predecessor array.	71
Figure 7.13.	Edge replacement with breadth-first search on the sample graph. .	72
Figure 7.14.	Edge replacement with predecessor array on the sample graph. . .	73
Figure 7.15.	Edge replacement with link-cut tree on the sample graph.	74
Figure 7.16.	Edge replacement with depth-first search on the sample graph. . .	75
Figure 8.1.	Building a min-heap binary tree.	78
Figure 8.2.	Predecessor array on the forest of included edges.	79
Figure 8.3.	Test instance generator.	82
Figure 8.4.	Performance profiles: MSTC formulations on Zhang's dataset. . .	86
Figure 8.5.	Performance profiles: MSTC formulations on Carrabs' dataset. . .	87
Figure 8.6.	Performance profiles: MSTC formulations on random dataset. . .	89
Figure 8.7.	Performance profiles: plain BB algorithms.	90

Figure 8.8.	Performance profiles: the effect of IT on diving algorithms.	92
Figure 8.9.	Performance profiles: the effect of both IT and Pr on BB-D.	94
Figure 8.10.	Performance profiles: the effect of both IT and Pr on BB-E.	95
Figure 8.11.	Performance profiles: the effect of both IT and Pr on BB-S.	97
Figure 8.12.	Performance profiles: Zhang's dataset.	99
Figure 8.13.	Performance profiles: Carrabs' dataset.	101
Figure 8.14.	Performance profiles: random dataset.	102
Figure 8.15.	Performance profiles: the effect of edge replacement algorithms on small and medium instances.	104
Figure 8.16.	Performance profiles: the effect of edge replacement algorithms on large instances.	105
Figure 8.17.	Performance profiles: cycle detection algorithms.	106

LIST OF TABLES

Table 3.1.	Literature review on MSTC problem.	9
Table 8.1.	Comparison of MSTC formulations on Zhang's Type 1 dataset. . .	85
Table 8.2.	Comparison of MSTC formulations on Zhang's Type 2 dataset. . .	85
Table 8.3.	Comparison of MSTC formulations on Carrabs' dataset.	87
Table 8.4.	Comparison of MSTC formulations on random dataset.	88
Table 8.5.	Comparison of plain BB algorithms.	90
Table 8.6.	Comparison of Rec-D and BB-D.	91
Table 8.7.	The effect of infeasibility tests and probing on BB-D.	93
Table 8.8.	The effect of infeasibility tests and probing on BB-E.	94
Table 8.9.	The effect of infeasibility tests and probing on BB-S.	96
Table 8.10.	Performance of BB algorithms on Zhang's dataset.	98
Table 8.11.	Performance of BB algorithms on Carrabs' dataset.	100
Table 8.12.	Performance of BB algorithms on random dataset.	101
Table 8.13.	Comparison of edge replacement algorithms on small and medium instances.	103

Table 8.14.	Comparison of edge replacement algorithms on large test instances.	104
Table 8.15.	Comparison of cycle detection algorithms.	106
Table 8.16.	The number of proven optimal solutions: the effect of number of vertices.	107
Table 8.17.	The number of proven optimal solutions: the effect of graph density.	108
Table 8.18.	The number of proven optimal solutions: the effect of conflict density.	109
Table A.1.	Test instances and comparison of formulations: Zhang’s dataset. .	121
Table A.2.	Test instances and comparison of formulations: Carrabs’ dataset. .	122
Table A.3.	Test instances and comparison of formulations: random dataset. .	126
Table B.1.	Plain branch-and-bound results: random dataset.	130
Table B.2.	The effect of IT and Pr on Rec-D using random dataset.	134
Table B.3.	The effect of IT and Pr on BB-D using random dataset.	138
Table B.4.	The effect of IT and Pr on BB-E using random dataset.	142
Table B.5.	The effect of IT and Pr on BB-S using random dataset.	146
Table B.6.	Branch-and-bound performance results: Zhang’s dataset.	150

Table B.7. Branch-and-bound performance results: Carrabs’ dataset. 151

Table B.8. Branch-and-bound performance results: random dataset. 155



LIST OF SYMBOLS

$ \cdot $	Cardinality notation
$A(D)$	Arc set of digraph D
B	Set of branching edges
$\mathbb{B}$	Boolean numbers
$\mathbf{c}$	Weight vector
c_e	Weight of edge e
C	Conflict graph
D	Digraph
$d_G(v)$	Degree of vertex v
$d_D^-(v)$	In-degree of vertex v
$d_D^+(v)$	Out-degree of vertex v
$E(G)$	Edge set of graph G
$\mathbf{f}$	Flow variables
F	Set of free edges
$\mathcal{F}$	Family of feasible sets
G	Graph
G'	Subgraph
$\overline{G}$	Complement graph
H	Conflict-free spanning tree
H^*	Minimum conflict-free spanning tree
I	Set of included edges
K	Clique
$\mathcal{K}(G)$	Family of cliques in graph G
L	Set of leaf edges
$\mathcal{L}$	Active node list
M	Matching
$MST^{(t)}$	Relaxed subproblem at node t of branch-and-bound search tree

$MSTC^{(t)}$	Subproblem at node t of branch-and-bound search tree
$N_G(v)$	Neighborhood of a vertex v in graph G
$\mathcal{NP}$	Class of problems solvable in non-deterministic polynomial time
$\mathcal{NP}$ -hard	Class of problems at least as hard as the hardest problem in $\mathcal{NP}$
O	Odd-cycle
$\mathcal{O}$	Big-O notation for problem complexity
$\mathcal{OC}(G)$	Family of odd cycles in graph G
P_{CB}	Fractional polytope described by cycle breaking constraints
P_{CFST}	Fractional conflict free spanning tree polytope
P_{CSTAB}	Fractional cycle-constrained stable set polytope
P_{CUT}	Fractional spanning tree polytope for cut-set formulation
P_{DCUT}	Fractional spanning tree polytope for directed cut formulation
P_{FSTAB}	Fractional stable set polytope
P_{KSTAB}	Fractional clique-constrained stable set polytope
P_{MAR}	Fractional spanning tree polytope for Martin's formulation
P_{MCF}	Fractional spanning tree polytope for multi-commodity flow formulation
P_{SCF}	Fractional spanning tree polytope for single commodity flow formulation
P_{ST}	Spanning tree polytope
P_{SP}	Fractional spanning tree polytope for set packing formulation
P_{STAB}	Stable set polytope
$\mathcal{P}$	Class of problems solvable in polynomial time
$parent(v)$	Parent of vertex v
$pred(v)$	Predecessor of vertex v
$\mathbb{R}$	Real numbers
$(S, \overline{S})$	Cut that partitions $V(G)$ into two disjoint subsets S and $\overline{S}$
S	Subset
t	Node t of branch-and-bound search tree

T	Spanning tree
T^*	Minimum spanning tree
U	Stable set
$V(D)$	Vertex set of digraph D
$V(G)$	Vertex set of graph G
W	Vertex maximal subset
$\mathbf{x}$	Characteristic vector for edges
X	Set of excluded edges
z	Value of the objective function
$\bar{z}$	Best upper bound
$\underline{z}$	Best lower bound
$\alpha(G)$	Stability number of graph G
β	Threshold ratio for probing
$\gamma_G(S)$	Set of edges with both endpoints in S in graph G
Γ	Graph density
$\bar{\Gamma}$	Average graph density
$\delta_G(S)$	Set of edges with exactly one endpoint in S in graph G
$\delta_D^+(S)$	Set of arcs with tail in S in digraph D
$\delta_D^-(S)$	Set of arcs with head in S in digraph D
$\delta_G(v)$	Subset of edges incident to vertex $v \in V(G)$
$\eta(G)$	Clique cover number of graph G
Θ	Coloring
Λ	Clique cover
$\mu(G)$	Chromatic number of graph G
$\nu(G)$	Matching number of graph G
π	Dual variables
Π_n	Path of length n
$\rho(G)$	Edge cover number of graph G
σ_n	Cycle of length n
$\tau(G)$	Vertex cover number of graph G

τ	Performance profiling factor
$\mathcal{V}$	Vertex cover
Ψ	Edge cover
Ω	Conflict density
$\overline{\Omega}$	Average conflict density
$\omega(G)$	Clique number of graph G



LIST OF ACRONYMS/ABBREVIATIONS

ADS	Abstract Data Structure
BB	Branch-and-Bound
BB-D	Diving Based Branch-and-Bound Algorithm
BB-E	Edge Based Branch-and-Bound Algorithm
BB-S	Stable Set Based Branch-and-Bound Algorithm
BC	Branch-and-Cut
BFS	Breadth-First Search
BILP	Binary Integer Linear Programming
CFST	Conflict Free Spanning Tree
COP	Combinatorial Optimization Problem
DFS	Depth-First Search
GA	Genetic Algorithm
ILP	Integer Linear Programming
IT	Infeasibility Tests
LCT	Link-and-Cut Tree
LP	Linear Programming
LH	Lagrangian Heuristic
LS	Local Search
MCP	Maximum Clique Problem
MILP	Mixed Integer Linear Programming
MSP	Maximum Stable Set Problem
MST	Minimum Spanning Tree
MSTC	Minimum Spanning Tree with Conflict Constraints
MWCP	Maximum Weight Clique Problem
MWSP	Maximum Weight Stable Set Problem
Pr	Probing
QMST	Quadratic Minimum Spanning Tree
Rec-D	Recursive Diving

TS	Tabu Search
TT	Tabu Thresholding
UB	Upper Bound



1. INTRODUCTION

The minimum spanning tree (MST) problem is well-known. It has been studied for nearly a century and can be solved in polynomial time with the aid of famous Prim's [1] and Kruskal's algorithms [2]. However, it is later discovered that another algorithm had also been devised by Boruvka in 1926 [3]. The MST problem has a wide variety of real-world applications, e.g., energy distribution [4], oil refinery networks [5], transportation science [6], telecommunication networks [7]. Several variations of the MST problem have already been reviewed in the literature, such as the maximum spanning tree problem [8], the capacitated MST problem [9], the order-constrained MST problem [10], the degree-constrained MST problem [11], and the minimum bottleneck spanning tree problem [12].

Disjunctive constraints in integer linear programming (ILP) models may arise depending on the type of the problem. They can be modeled as *either-or* type and/or *if-then* type of constraints [13]. In graph theory, a disjunction implies the choice of at most one of the edges or vertices of a given pair of edges or vertices, respectively. In this work, we focus on disjunctions on the edge pairs. In other words, if at most one of the edges in a given pair is allowed in a solution, then a disjunction or conflict occurs. This is the opposite of forcing, which arises when at least one of the edges in a given pair must be included in a solution [14].

Although there are various theoretical studies in the literature on the minimum spanning tree with conflict constraints (MSTC) problem [15–20], there is only one real-world application to the best of our knowledge, where optimal methods for routing in offshore wind farm cable deployment with forbidden cable crossings are proposed [21]. The authors model scenarios and solve real-world instances by a hybrid/exact algorithm. Furthermore in [22], an optimization approach for the cable layout design in large offshore wind farms is discussed, aiming to reduce costs through a Mixed ILP (MILP) model enhanced by heuristics, which is proven to be effective in producing

cost-efficient layouts compared to traditional methods. Several real-world problems can be modeled as MSTC, e.g., forbidden pipeline crossings in oil refineries, conflicts in transportation lines due to governmental regulations.

An illustration is provided with Figure 1.1, with a substation, 53 wind turbines, and four different paths of cables connecting wind turbines to the substation. No pair of cables cross over each other, as can be observed.

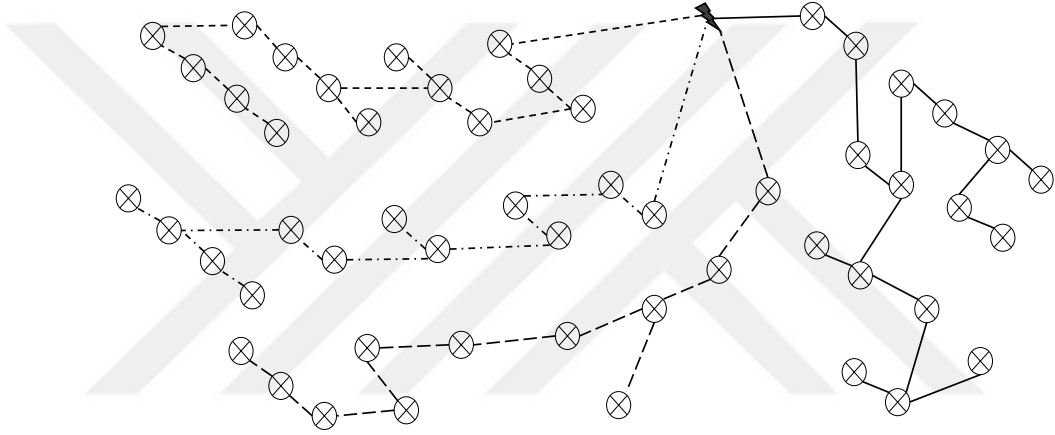


Figure 1.1. Sample illustration of offshore wind farm cable routing without cable crossings.

The primary focus of this study is on the development of efficient exact solution algorithms for solving MSTC problem. The remaining chapters of this report is organized as follows. Chapter 2 introduces basic concepts and notations used within the thesis. Chapter 3 provides a critical survey of related work and a statement of thesis contributions. Chapter 4 gives an outline of the MSTC formulations. Chapter 5 describes the generic branch-and-bound (BB) frame, followed by a review of branching rules. Then, subproblem selection and relaxation strategies are discussed and major components of the BB algorithms are introduced. In Chapter 6, various strategies for updating relaxed subproblems are reviewed. Chapter 7 presents the computational test environment, and reports the results of the experiments. Finally, Chapter 8 concludes the thesis and addresses potential future research avenues.

2. BASIC CONCEPTS AND NOTATIONS

Let $G = (V(G), E(G))$ be a connected, simple graph with vertex set $V(G)$, edge set $E(G)$, and nonnegative edge weights c_e for each $e \in E(G)$. The number of vertices of G is denoted by $|V(G)|$ and the number of edges of G is represented with $|E(G)|$ where $|\cdot|$ represents the cardinality of a set. For each vertex $v \in V(G)$, the neighborhood of v , denoted by $N_G(v)$, is the set of vertices which are adjacent to v . The subset of edges incident to vertex $v \in V(G)$ is denoted by $\delta_G(v)$, and $d_G(v)$ stands for the degree of vertex v ; $d_G(v) = |\delta_G(v)| = |N_G(v)|$. The complement of G is $\overline{G} = (V(\overline{G}), E(\overline{G}))$ where $V(\overline{G}) \equiv V(G)$ and $\{u, v\} \in E(\overline{G})$ if and only if $\{u, v\} \notin E(G)$ for $u \neq v$. The graph density $\Gamma = 2|E(G)|/[|V(G)|(|V(G)| - 1)]$ is a measure of how dense graph G is in terms of edges compared to the maximum number of possible edges.

A path of length n in graph G is a finite sequence of vertices $\Pi_n = v_1, v_2, \dots, v_n$ such that each consecutive vertex pair corresponds to an edge in G , i.e., $\{v_{i-1}, v_i\} \in E(G)$ where $2 \leq i \leq n$. Π_n is a simple path when all vertices $v_1, v_2, \dots, v_n$ of the path are distinct. $\Pi_n = v_1, v_2, \dots, v_n$ is a closed path if $v_1 = v_n$. A cycle of length $n \geq 2$ in a graph G , also called an n -cycle, is a simple closed path $\sigma_n = v_1, v_2, \dots, v_n, v_1$. If n is odd (even), σ_n is an odd-cycle (even-cycle). A chord is an edge which is not included in a cycle but it joins two vertices of the cycle. A chordless cycle σ_n of length $n \geq 4$ is called a hole. Besides, if n is odd, it is called an odd hole. The complement of a hole is called an antihole, and the complement of an odd hole is an odd antihole.

A cut is defined as a partition of $V(G)$ into two disjoint subsets $S \subseteq V(G)$ and $\overline{S} = V(G) \setminus S$. The set of edges that have one endpoint in S and the other endpoint in $\overline{S}$ is $\delta_G(S) = \{\{u, v\} \in E(G) : u \in S, v \in \overline{S}\} \subseteq E(G)$, which is also called a cut-set or simply a cut of G . The set of edges that have both endpoints in S is $\gamma_G(S) = \{\{u, v\} \in E(G) : u \in S, v \in S\} \subseteq E(G)$. A cut is elementary if it does not include any other cut properly.

A *directed graph* (or *digraph*) $D = (V(D), A(D))$ is a set of $V(D)$ vertices and a set of $A(D)$ arcs where each arc $a \in A(D)$ is an ordered pair (u, v) of vertices in $V(D)$. Here, u is the *tail* and v is the *head* of arc a . In a digraph, the *in-degree* of a vertex v , denoted by $d_D^-(v)$, is the number of arcs that point to v (or where v is the head), which is shown as $\delta_D^-(v)$. The *out-degree* of a vertex v , denoted by $d_D^+(v)$, is the number of arcs that point away from v (or where v is the tail), which is shown as $\delta_D^+(v)$. The set of arcs directed out of vertex set S is denoted by $\delta_D^+(S) = \{\{u, v\} \in A(D) : u \in S, v \in \overline{S}\}$ and the set of arcs directed into vertex set S is denoted by $\delta_D^-(S) = \{\{u, v\} \in A(D) : u \in \overline{S}, v \in S\}$.

A *stable* or *independent* set $U \subseteq V(G)$ is a collection of mutually non-adjacent vertices in G , i.e., for every vertex pair in U , there is no edge between them. A stable set is (vertex) maximal if it is not a proper subset of any other stable set. A *clique* $K \subseteq V(G)$ is a subset of the vertices such that every pair of vertices in K are adjacent. A *vertex-coloring* is a partition of vertices $\Theta \subseteq V(G)$ into disjoint stable sets $U_1, \dots, U_k$, which are called as *colors* or *color classes* of the coloring. A *clique cover* is a partition of vertices $\Lambda \subseteq V(G)$ into cliques.

We may define the stability number of graph G as $\alpha(G) = \max \{|U| : U \text{ is a stable set in } G\}$ and the clique number as $\omega(G) = \max \{|K| : K \text{ is a clique in } G\}$. Similarly, we can define the chromatic number as $\mu(G) = \min \{|\Theta| : \Theta \text{ is a coloring of } G\}$ and the clique cover number as $\eta(G) = \min \{|\Lambda| : \Lambda \text{ is a clique cover of } G\}$. U is a stable set of G if and only if U is a clique of $\overline{G}$. It can be verified that $\alpha(G) = \omega(\overline{G})$, $\eta(G) = \mu(\overline{G})$ and $\omega(G) \leq \mu(G)$.

The Maximum (Cardinality) Stable Set Problem (MSP) is the problem of determining a stable set $U \subseteq V(G)$ of the largest cardinality $\alpha(G)$, which is equivalent to the Maximum Clique Problem (MCP) in $\overline{G}$, which finds a clique $K \subseteq V(\overline{G})$ of the largest size $\omega(\overline{G})$. The Maximum Weight Stable Set Problem (MWSP) is the problem of finding a stable set $U \subseteq V(G)$ with maximum weight in G with vertex weights. It is equivalent to the Maximum Weight Clique Problem (MWCP) in $\overline{G}$, where a clique

$K \subseteq V(\overline{G})$ with maximum weight in $\overline{G}$ is found. All these problems are strongly $\mathcal{NP}$ -hard [23].

A graph G is *perfect* if and only if $\omega(G') = \mu(G')$ for every induced subgraph G' of G . Similarly, a graph G is perfect if and only if $\alpha(G') = \eta(G')$ for every induced subgraph G' of G . A graph G is perfect if and only if its complement $\overline{G}$ is perfect. A graph G is perfect if and only if it does not contain any *odd holes* or *odd antiholes* as induced subgraphs [24]. A graph G is perfect if and only if $\omega(G') \times \alpha(G') \geq |V(G')|$ for each induced subgraph G' of G [25]. A graph is *t-perfect* if its stable set polytope is given by rank constraints associated with vertices, edges, triangles, or odd holes ¹ [26]. A graph is *h-perfect* if its stable set polytope is given by rank constraints associated with cliques of arbitrary size and odd holes ² [27]. An *n-ladder* is a graph consisting of two paths of length n , i.e., $\Pi_{1n} = v_{11}, v_{12}, \dots, v_{1n}$ and $\Pi_{2n} = v_{21}, v_{22}, \dots, v_{2n}$, where each consecutive vertex from Π_{1n} and Π_{2n} are connected with an edge. Hence, an *n-ladder* graph has $2n$ vertices and $3(n - 2)$ edges.

A graph G is *connected* if for every pair of vertices u and v , there is a path connecting u and v . A graph G is *acyclic* if it contains no cycles. A *forest* is an acyclic graph and a *tree* is a connected acyclic graph. Similarly, a forest is a disjoint union of trees. A *spanning tree* T is a connected acyclic subgraph of G , which is denoted by $T = (V(T), E(T))$ where $V(T) \equiv V(G)$, $E(T) \subseteq E(G)$ and $|E(T)| = |V(T)| - 1$. An *MST* is the spanning tree with the minimum possible total edge weight. Given the edge weights $c_e, e \in E(G)$, the MST problem aims to find a spanning tree of graph G with minimum total weight. Since there are known polynomial time algorithms to find an MST on a given graph G , MST problem is in class $\mathcal{P}$.

A *conflict graph* $C = (V(C), E(C))$ is defined with $V(C) \equiv E(G)$ and $E(C)$ consists of edge pairs $\{e, f\}$ where $e, f \in E(G)$ and $e \neq f$. Such an edge pair $\{e, f\} \in E(C)$ is defined as a conflicting pair. The vertices of the conflict graph are actually the edges of the original graph G and the edges of the conflict graph are the elements of the

¹*t* is the abbreviation for “trou”, the French word for hole

²*h* stands for “hole-perfect”

conflict set $E(C)$. We clearly assume that a “conflict” is a symmetrical relationship, i.e., e is in conflict with f if and only if f does so with e . The *conflict density* $\Omega = 2|E(C)|/[|V(C)|(|V(C)| - 1)]$ represents the density of conflicts among edges.

A *conflict-free spanning tree* H is any spanning tree, which includes at most one of the edges e or f for each conflicting edge pair $\{e, f\} \in E(C)$. An optimal conflict-free MST H^* is a conflict-free spanning tree with the minimum possible total weight. Given a graph $G = (V(G), E(G))$ with nonnegative edge weights c_e , and a conflict graph $C = (V(C), E(C))$, the MSTC problem aims to find a conflict-free minimum spanning tree H^* of G , if it exists. It is shown that the MSTC problem is strongly $\mathcal{NP}$ -hard [15].

3. RELATED WORKS AND OUR CONTRIBUTIONS

The MSTC problem is probably the most studied combinatorial optimization problem with conflicts regarding the existing literature. A brief critical survey of the related works followed by the outline of the contributions made in this thesis are provided in this chapter.

3.1. Related Works

In the seminal work on the MSTC problem [15], the authors analyze the complexity of the problem for specific classes of conflict graphs. They show that it is strongly $\mathcal{NP}$ -hard even if the conflict graph is a 3-ladder and weights are 0 or 1. They also show that the MSTC problem is polynomially solvable when the conflict graph is a 2-ladder, based on a matroid intersection theorem. They also provide a BILP formulation.

The MSTC problem is further investigated in [16] where the authors observe that obtaining a lower bound is possible by solving the maximum edge-clique partitioning problem on the conflict graph C , and propose a polynomial-time approximate greedy algorithm that has the best performance ratio so far. BILP formulations of the MSTC problem are given, and a Lagrangean relaxation scheme is proposed for calculating stronger lower bounds. A straightforward construction heuristic is proposed in order to obtain upper bounds, and various heuristics, including local search (LS), tabu thresholding (TT), tabu search (TS) and Lagrangean heuristics (LH) are developed. Infeasibility tests such as the maximum stable set test, clique partitioning test, minimum vertex cover test, and a test which uses single commodity flow formulation are devised. The initial random test instances for the MSTC problem are generated, which are also used in later studies for benchmarking.

A preprocessing scheme and a branch-and-cut (BC) algorithm separating exponential number of valid inequalities for the MSTC problem are proposed in [17]. The

preprocessing algorithm is used either to find a feasible solution to the MSTC problem or to reduce its size by bridge detection via depth-first search, variable fixing, and chain probing. The BC algorithm starts with the solution of the relaxed linear programming (LP) model, then searches for either violated cycle-breaking constraints or cut inequalities depending on the chosen formulation, and odd-cycle inequalities by separation procedures. When any of the two procedures is able to separate current solution, the corresponding cut is added and the LP is solved. Otherwise, if both procedures fail, BB iterations are applied. Their results indicate that the developed exact solution algorithm yields an improvement on the bounds, also providing new feasibility and optimality conditions for the MSTC problem. As an endnote, the authors suggested further research on odd hole, maximal clique, and rank inequalities for the MSTC polytope in order to develop more classes of facet-defining valid inequalities.

A BC algorithm employing a new set of valid inequalities, which are degree-cut, conflict-cycle, and odd-cycle inequalities, is proposed in [18]. To obtain upper bounds, a genetic algorithm (GA) designed in [28] is used. However, the authors denote that there is no guarantee of obtaining upper bounds. Three different separation procedures are considered after solving the LP relaxation. If all separation procedures fail, branching is based on variables. The performance of the BC algorithm is compared with the BC algorithm in [17], also new test instances are generated. The authors suggest that finding additional valid inequalities could improve the performance of the BC algorithm.

A Lagrangean relaxation scheme is developed in a recent study [19] in order to obtain good quality lower bounds. It is a combination of a dual ascent procedure and a subgradient algorithm. The dual ascent procedure makes use of Kruskal's algorithm to update the Lagrange multipliers. For the cases where the dual ascent procedure does not increase the objective function value, they switch to the subgradient algorithm. The hybrid algorithm is named as Hybrid Dual Ascent. A variant of this algorithm runs a local search procedure which decreases efficiency, but increases the quality of the lower bound.

Yet another work on the Lagrangean relaxation bounds is realized by [20], where the authors focus on polyhedral properties of fixed cardinality stable sets. The provided bounds are obtained by dualizing subtour constraints while limiting the number of Lagrange multipliers. Their approach is based on integrating dual ascent and a subgradient algorithm to compute lower bounds efficiently for the MSTC problem.

The related literature on the MSTC problem is summarized with Table 3.1 according to the complexity results, preprocessing algorithms, infeasibility test, heuristics and exact solution methods.

Table 3.1. Literature review on MSTC problem.

No.	Author	Year	Complex.	Preproc.	Infeas. Test	Heuristic	Exact
1	Darmann et. al. [15]	2011	Yes	No	No	None	None
2	Zhang et. al. [16]	2011	Yes	No	Yes	LS, TT, TS, LH	None
3	Samer & Urrutia [17]	2015	No	Yes	No	None	BC
4	Carrabs et. al. [18]	2021	No	No	No	GA	BC
5	Carrabs et. al. [19]	2021	No	Yes	No	LH	None
6	Samer & Haugland [20]	2023	No	No	No	LH	None

Various other combinatorial optimization problems with conflicts have already been studied in the literature, e.g., the knapsack problem [29], the maximum flow problem [30,31], the minimum cost flow problem [32], the maximum weight matching problem and the shortest path problem [15], the bin packing problem [33], the transportation problem [34], the assignment problem [35], and the set covering problem [36].

3.2. Our Contributions

Regarding the existing literature on the MSTC problem, our contribution can be classified within the exact solution methods. We propose three novel BB algorithms to solve the MSTC problem, each of which bases its search procedure on a different characteristic of the problem. Detecting infeasibilities and restoring the disconnected tree after the deletion of an edge without solving the problem from scratch, is another contribution. In addition, we evaluate the performance of four different approaches for tree restoration, and four different techniques to detect cycles with the aid of various data structures. The introduction of size-reducer probing and feasible-solution-provider diving procedures into the BB algorithms results in considerable increase in their performances. Finally, a more robust and reliable test problem generator for MSTC instances is established, compared with the existing ones in the literature.

4. FORMULATIONS

Due to the rich inventory of subtour elimination constraints [37], it is possible to obtain a large set of MST problem formulations. Each one of them can result in at least three MSTC problem formulations. We introduce and explain only the ones that we consider as reference formulations for the computational experiments realized within the scope of this thesis.

4.1. Minimum Spanning Tree Problem

4.1.1. Combinatorial Optimization Formulation

The combinatorial optimization problem (COP) version of the MST problem for a given graph G can be described by the family of feasible sets $\mathcal{F}_{ST} = \{T_1, T_2, \dots, T_k\}$ which is a family of spanning trees in G , and the objective function is $c(T) = \sum_{e \in E(T)} c_e$. Then, the problem can be formulated as

$$\min \left\{ \sum_{e \in E(T)} c_e : T \in \mathcal{F}_{ST} \right\}. \quad (4.1)$$

Recall that, we can describe each solution $T_i = (V(T_i), E(T_i))$, for $i = 1, \dots, k$ by the corresponding characteristic vector $\mathbf{x}(T_i) \in \mathbb{B}^{|E(G)|}$, where $x_e(T_i) = 1$ if $e \in E(T_i)$ and $x_e(T_i) = 0$ if $e \notin E(T_i)$. Then, the spanning tree polytope is defined as $P_{ST}(G) = \text{conv}(\{\mathbf{x}(T) : T \in \mathcal{F}_{ST}\})$.

A generic Binary Integer Linear Programming (BILP) model for the MST problem is given below for the sake of increasing readability throughout the thesis. Let us define the decision variable x_e as follows.

$$x_e = \begin{cases} 1, & \text{if edge } e \in E(G) \text{ is included in the spanning tree,} \\ 0, & \text{otherwise.} \end{cases}$$

The generic formulation of the MST problem (MST-gen) is

$$\text{MST-gen:} \quad \min \quad \sum_{e \in E(G)} c_e x_e \quad (4.2)$$

$$\text{s.t.} \quad \sum_{e \in E(G)} x_e = |V(G)| - 1 \quad (4.3)$$

$$\mathbf{x} \in P_{CB}(G) \quad (4.4)$$

$$x_e = 0, 1 \quad e \in E(G). \quad (4.5)$$

The objective function (4.2) aims to minimize the total edge weight of the spanning tree. Constraint (4.3) ensures that exactly $|V(G)| - 1$ edges should be included in the spanning tree. Constraints (4.4) state that any feasible $\mathbf{x} \in \mathbb{B}^{|E(G)|}$ cannot represent a cycle in G , i.e., $P_{CB}(G)$ is the polytope described by the constraints breaking cycles in G . Finally, constraints (4.5) impose the integrality restrictions.

4.1.2. Undirected Formulations

Among the six well-known formulations for the MST problem, three of them are based on the undirected graph $G = (V(G), E(G))$.

4.1.2.1. Set Packing Formulation. In the first undirected formulation of the MST problem, cycles are eliminated using set packing constraints [38], and the set packing formulation (MST-sp) is obtained by replacing constraints (4.4) in MST-gen by the following set of inequalities.

$$\sum_{e \in \gamma_G(S)} x_e \leq |S| - 1 \quad S \subset V(G), |S| \geq 2. \quad (4.6)$$

Constraints (4.6) are so-called set packing constraints ensuring that no cycles are formed. They are also known as cycle-breaking constraints in the literature. Note that such constraints are exponentially many. The LP relaxation of the MST-sp is obtained when the integrality restrictions (4.5) are replaced by

$$x_e \geq 0 \quad e \in E(G). \quad (4.7)$$

In the LP relaxation, the inequalities $x_e \leq 1$ become redundant because they are implied by constraints (4.6) for vertex sets of size two, i.e., for $|S| = 2$. The polytope of the LP relaxation of MST-sp, i.e., the fractional MST-sp polytope, is defined as $P_{SP}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \mathbf{x} \text{ satisfies (4.3), (4.6) and (4.7)}\}$.

4.1.2.2. Cut-set Formulation. In another formulation of the MST problem, cycles are broken by the cut-set inequalities (MST-cut) [38], which is obtained by replacing constraints (4.4) in MST-gen by the following set of inequalities.

$$\sum_{e \in \delta_G(S)} x_e \geq 1 \quad S \subset V(G), S \neq \emptyset. \quad (4.8)$$

Constraints (4.8) imply that a spanning tree must have at least one edge in the cut-set of any subset of the vertices $\delta_G(S)$, ensuring that subsets S and $\bar{S}$ remain connected. Observe that MST-cut has exponentially many cut-set inequalities. In the LP relaxation of MST-cut, constraints (4.5) are replaced by

$$0 \leq x_e \leq 1 \quad e \in E(G). \quad (4.9)$$

The polytope of the LP relaxation of MST-cut, i.e., the fractional MST-cut polytope, is defined as $P_{CUT}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \mathbf{x} \text{ satisfies (4.3), (4.8) and (4.9)}\}$.

4.1.2.3. Martin's Formulation. Another undirected formulation for the MST problem is Martin's formulation (MST-mar) [39]. This is an extended formulation where a new decision variable y_{ij}^k is defined for each vertex pair $\{i, j\} \in E(G)$, and for each vertex $k \in V(G)$. $y_{ij}^k = 1$ indicate that vertex k is within the resulting subtree that includes vertex j after edge $\{i, j\}$ is deleted from the spanning tree. It can be defined more formally as

$$y_{ij}^k = \begin{cases} 1, & \text{if the path between vertices } i \text{ and } k \text{ passes from vertex } j, \\ 0, & \text{otherwise.} \end{cases}$$

Variable x_e keeps its usual definition: $x_e = 1$ if e belongs to a spanning tree; $x_e = 0$ otherwise. MST-mar model is obtained by replacing constraints (4.4) in MST-gen by

introducing the following constraints:

$$y_{ij}^k + y_{ji}^k = x_e \quad e = \{i, j\} \in E(G), k \in V(G) \quad (4.10)$$

$$\sum_{k \in V(G) \setminus \{i, j\}} y_{ik}^j + x_e = 1 \quad e = \{i, j\} \in E(G) \quad (4.11)$$

$$y_{ij}^k = 0, 1 \quad \{i, j\} \in E(G), k \in V. \quad (4.12)$$

Constraints (4.10) ensure that when edge $e = \{i, j\}$ is selected with any vertex k , either the path between vertex i and vertex k passes from vertex j or the path between vertex j and vertex k traverses vertex i . If edge $e = \{i, j\}$ is not in the tree, then for any vertex k , the path between i and k does not traverse vertex j and vice versa. Constraints (4.11) verify that if edge $e = \{i, j\}$ is included in the spanning tree, edges $\{i, k\}$ that connect i do not pass from vertex j . If edge $e = \{i, j\}$ is not included in the tree, then there exists an edge $\{i, k\}$ such that the path between vertices i and j passes from vertex k . Finally, constraints (4.12) are the integrality restrictions for the variables y_{ij}^k .

This formulation has a polynomial number of constraints. The LP relaxation of the MST-mar is obtained when constraints (4.5) are replaced with constraints (4.9). The polytope of the LP relaxation of MST-mar, i.e., the fractional MST-mar polytope, becomes $P_{MAR}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \exists \mathbf{y} \text{ s.t. } (\mathbf{x}, \mathbf{y}) \text{ satisfies (4.3), (4.9) and (4.10) - (4.12)}\}$.

4.1.3. Directed Formulations

4.1.3.1. Directed Cut-set Formulation. A stronger version of MST-cut is its directed version, which is abbreviated as MST-dcut [38]. The MST-dcut model is defined on a digraph $D = (V(D), A(D))$ where edges $e = \{i, j\} \in E(G)$ are represented by symmetric arcs $(i, j) \in A(D)$ and $(j, i) \in A(D)$. Given a root node $r \in V(D)$, if arc (i, j) is included in the spanning tree then $y_{ij} = 1$, and $y_{ij} = 0$ otherwise. The MST-dcut model has the same objective function and integrality restrictions with MST-gen,

but constraints (4.3) and (4.4) are expressed differently as:

$$\sum_{\{i,j\} \in A(D)} y_{ij} = |V(G)| - 1 \quad (4.13)$$

$$\sum_{\{i,j\} \in \delta_D^+(S)} y_{ij} \geq 1 \quad r \in S \subset V(G) \quad (4.14)$$

$$x_e = y_{ij} + y_{ji} \quad e = \{i, j\} \in E(G) \quad (4.15)$$

$$y_{ij} \geq 0 \quad \{i, j\} \in E(G). \quad (4.16)$$

This model is equivalent to the undirected version with the exception of constraints (4.14) implying that every directed cut $\delta_D^+(S)$, which disconnects the root vertex $r \in S$ from the remaining set of vertices $V(G) \setminus S$, has at least one arc leaving it. The LP relaxation of MST-dcut is obtained when constraints (4.5) are replaced with constraints (4.9). The polytope of the LP relaxation of MST-dcut, i.e., the fractional MST-dcut polytope, is defined as $P_{DCUT}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \exists \mathbf{y} \text{ s.t. } (\mathbf{x}, \mathbf{y}) \text{ satisfies (4.9) and (4.13) - (4.16)}\}$.

4.1.3.2. Single Commodity Flow Formulation. The MST problem can also be modeled using single commodity flow constraints [38, 40]. To this end, let us introduce auxillary flow variables f_{ij} which represent the amount of flow of a single commodity from vertex i to vertex j on each edge $e = \{i, j\} \in E(G)$. Note that, exactly one unit of flow must be sent to each vertex from the source vertex s , which is in fact the root of the spanning tree. Also, let $\delta_D^+(i)$ and $\delta_D^-(i)$ denote the outflow to and inflow from vertex i , respectively. The single commodity flow model (MST-scf) is obtained by redefining constraints (4.4) in MST-gen with the following set of inequalities.

$$\sum_{\{i,j\} \in \delta_D^+(s)} f_{ij} - \sum_{\{j,i\} \in \delta_D^-(s)} f_{ji} = |V(G)| - 1 \quad (4.17)$$

$$\sum_{\{j,i\} \in \delta_D^-(v)} f_{ji} - \sum_{\{i,j\} \in \delta_D^+(v)} f_{ij} = 1 \quad v \in V(G), v \neq s \quad (4.18)$$

$$f_{ij} + f_{ji} \leq (|V(G)| - 1)x_e \quad e = \{i, j\} \in E(G) \quad (4.19)$$

$$f_{ij} \geq 0 \quad \{i, j\} \in E(G). \quad (4.20)$$

Constraints (4.17) and (4.18) imply the connectivity of the vertex set $V(G)$ by modelling flow balance at the vertices. Constraints (4.19) establishes a connection between the flow variables $\mathbf{f}$ and edge variables $\mathbf{x}$. Finally, constraints (4.20) ensure the nonnegativity of flow on all edges. This formulation also has a polynomial number of constraints. The LP relaxation of MST-scf is obtained where constraints (4.5) are replaced with constraints (4.9). The polytope of the LP relaxation of MST-scf, i.e., the fractional MST-scf polytope, is defined as $P_{SCF}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \exists \mathbf{f} \text{ s.t. } (\mathbf{x}, \mathbf{f}) \text{ satisfies (4.3), (4.9) and (4.17) - (4.20)}\}$.

4.1.3.3. Multi-commodity Flow Formulation. A stronger version of MST-scf is the multi-commodity flow formulation (MST-mcf) [38]. The MST-mcf introduces auxiliary decision variables f_{ij}^k , which represent the flow of commodity sent to vertex k passing from vertex i to vertex j . The MST-mcf model which shares the same objective function and the integrality and cardinality restrictions with MST-gen, and includes the following flow constraints.

$$\sum_{\{j,i\} \in \delta_D^-(r)} f_{ji}^k - \sum_{\{i,j\} \in \delta_D^+(r)} f_{ij}^k = -1 \quad k \neq r \in V(G) \quad (4.21)$$

$$\sum_{\{j,i\} \in \delta_D^-(v)} f_{ji}^k - \sum_{\{i,j\} \in \delta_D^+(v)} f_{ij}^k = 0 \quad v \neq k \in V(G), v \neq r \quad (4.22)$$

$$\sum_{\{j,i\} \in \delta_D^-(k)} f_{ji}^k - \sum_{\{i,j\} \in \delta_D^+(k)} f_{ij}^k = 1 \quad k \neq r \in V(G) \quad (4.23)$$

$$f_{ij}^k + f_{ji}^l \leq x_e \quad e = \{i, j\} \in E(G), k \neq l \in V(G) \quad (4.24)$$

$$f_{ij}^k \geq 0 \quad \{i, j\} \in E(G), k \in V(G). \quad (4.25)$$

Constraints (4.21) - (4.23) guarantee the flow balance for every commodity k , which ensure the connectivity of the vertex set $V(G)$. Constraints (4.24) establish a connection between the flow variables $\mathbf{f}$ and edge selection variables $\mathbf{x}$. Constraints (4.25) ensure the nonnegativity of flow variables. This formulation also has a polynomial number of constraints. The LP relaxation of MST-mcf is obtained when constraints (4.5) are replaced with constraints (4.9). The polytope of the LP relaxation of MST-mcf, i.e., the fractional MST-mcf polytope, is defined as $P_{MCF}(G) = \{\mathbf{x} \in \mathbb{R}^{|E(G)|} : \exists \mathbf{f} \text{ s.t. } (\mathbf{x}, \mathbf{f}) \text{ satisfies (4.3), (4.9) and (4.21) - (4.25)}\}$.

In [38, 40], the relationships between the spanning tree polytopes are analyzed and $P_{ST} \equiv P_{SP} \equiv P_{MAR} \equiv P_{DCUT} \equiv P_{MCF} \subseteq P_{CUT} \subseteq P_{SCF}$ is proven. The optimal solution $\mathbf{x}^*$ of the polytopes of LP relaxations of both set packing and Martin's formulations (MST-sp and MST-mar) are integral, and an optimal MST is obtained from the solutions of the LP relaxations. MST-cut is a weaker formulation since it can yield a fractional optimal solution that is infeasible for the BILP model. MST-scf is even weaker since it can also yield a fractional optimal solution with an integrality gap being even worse compared to MST-cut. The LP relaxations of both MST-dcut and MST-mcf also give integral optimal solution due to P_{DCUT} and P_{MCF} being integral.

As can be noticed, any MST formulation includes cycle-breaking or subtour elimination constraints in order to guarantee an acyclic structure. They are also the major components of the Asymmetric Travelling Salesman Problem (ATSP), which is extensively studied in the related literature [37]. As a result, integrating one of them into MST-gen by replacing constraints (4.4) with a different set of cycle-breaking constraints result yet in another MST formulation.

4.2. Maximum Cardinality and Weight Stable Set Problems

The COP version of the MSP for a given graph G can be described by the family of feasible sets $\mathcal{F}_{STAB} = \{U_1, U_2, \dots, U_k\}$, which consists of a collection of stable sets $U \subseteq V(G)$, and the objective function $|U|$. Then, the problem becomes

$$\alpha(G) = \max \left\{ |U| : U \in \mathcal{F}_{STAB} \right\}. \quad (4.26)$$

Each solution $U \in \mathcal{F}_{STAB}$ can be described by the corresponding characteristic vector $\mathbf{x}(U) \in \mathbb{B}^{V(G)}$ where $x_v(U) = 1$ if $v \in U$ and $x_v(U) = 0$ if $v \notin U$.

MSP can be formulated on a given graph $G = (V(G), E(G))$ as a BILP with $|E(G)|$ constraints and $|V(G)|$ variables [27]. The decision variable x_v is defined as

$$x_v = \begin{cases} 1, & \text{if vertex } v \in V(G) \text{ is in a stable set,} \\ 0, & \text{otherwise.} \end{cases}$$

The BILP model for the MSP is expressed as:

$$\text{MSP: } \max \sum_{v \in V(G)} x_v \quad (4.27)$$

$$\text{s.t. } x_u + x_v \leq 1 \quad \{u, v\} \in E(G), u \neq v \quad (4.28)$$

$$x_v = 0, 1 \quad v \in V(G). \quad (4.29)$$

The objective function (4.27) aims to maximize the total number of vertices in the stable set $U \subseteq V(G)$. Given nonnegative vertex weights, $c_v, v \in V(G)$, MWSP is formulated by replacing the objective function (4.27) with

$$\max \sum_{v \in V(G)} c_v x_v. \quad (4.30)$$

Constraints (4.28) are the vertex packing constraints which ensure that at most one of the endpoints of an edge can belong to a stable set, and constraints (4.29) are for the integrality restrictions. The LP relaxation is obtained when constraints (4.29) are replaced with

$$x_v \geq 0 \quad v \in V(G). \quad (4.31)$$

Notice that the inequalities $x_v \leq 1, v \in V(G)$ are redundant. Let us define the fractional stable set polytope as $P_{FSTAB}(G) = \{\mathbf{x} \in \mathbb{R}^{|V(G)|} : \mathbf{x} \text{ satisfies (4.28) and (4.31)}\}$. The LP relaxation is quite weak since the vertices of $P_{FSTAB}(G)$ can take the values of $(0, \frac{1}{2}, 1)$. Note that for perfect graphs, $P_{FSTAB}(G) \equiv P_{STAB}(G)$ is satisfied, where $P_{STAB}(G) = \text{conv}\left(\{\mathbf{x}(U) : U \in \mathcal{F}_{STAB}\}\right)$ [41].

In order to strengthen $P_{FSTAB}(G)$, two sets of constraints, which are valid inequalities for the stable set polytope are proposed in the literature [41]. The first one is the set of odd-cycle inequalities

$$\sum_{v \in O} x_v \leq \frac{|O| - 1}{2} \quad O \in \mathcal{OC}(G), \quad (4.32)$$

where $\mathcal{OC}(G)$ is the family of odd cycles of G . They imply that the cardinality of any stable set S that is a subset of edges including an odd cycle $O \subseteq V(G)$ can

be at most $\frac{|O|-1}{2}$. The fractional cycle-constrained stable set polytope is defined as $P_{CSTAB}(G) = \{\mathbf{x} \in \mathbb{R}^{|V(G)|} : \mathbf{x} \text{ satisfies (4.28), (4.31) and (4.32)}\}$.

The second one is the set of clique inequalities

$$\sum_{v \in K} x_v \leq 1 \quad K \in \mathcal{K}(G), \quad (4.33)$$

where $\mathcal{K}(G)$ is the family of cliques in G . They are based on the fact that a clique and a stable set can have at most one vertex in common. The fractional clique-constrained stable set polytope is defined as $P_{KSTAB}(G) = \{\mathbf{x} \in \mathbb{R}^{|V(G)|} : \mathbf{x} \text{ satisfies (4.28), (4.31) and (4.33)}\}$.

It can be shown that $P_{STAB}(G) \subseteq P_{CSTAB}(G)$ and $P_{STAB}(G) \subseteq P_{KSTAB}(G)$ holds [27]. For t-perfect graphs, $P_{CSTAB}(G) \equiv P_{STAB}(G)$ is valid [27]. Furthermore, odd-cycle inequalities inducing an odd hole (4.32) are facet-defining for $P_{STAB}(G)$. For perfect graphs, $P_{KSTAB}(G) \equiv P_{STAB}(G)$ is satisfied [27]. The clique inequalities (4.33) are facet-defining for $P_{STAB}(G)$ if and only if K is a maximal clique in G [42]. Graphs that satisfy $P_{STAB}(G) \equiv P_{CSTAB}(G) \cap P_{KSTAB}(G)$ are called as h-perfect graphs. The odd-cycle inequalities, clique inequalities and nonnegativity restrictions are sufficient to describe $P_{STAB}(G)$ for h-perfect graphs [27].

4.3. Minimum Spanning Tree Problem with Conflicts

4.3.1. Binary Integer Linear Programming Formulations

The MSTC problem can be formulated as a BILP model, which is based on MST problem formulations with the addition of conflict constraints (4.28). A generic formulation for the MSTC problem is given in [16], which is provided as MSTC-gen.

$$\text{MSTC-gen: } \min \sum_{e \in E(G)} c_e x_e \quad (4.34)$$

$$\text{s.t. } \sum_{e \in E(G)} x_e = |V(G)| - 1 \quad (4.35)$$

$$\mathbf{x} \in P_{CB}(G) \quad (4.36)$$

$$x_e + x_f \leq 1 \quad \{e, f\} \in E(C) \quad (4.37)$$

$$x_e = 0, 1 \quad e \in E(G). \quad (4.38)$$

The feasible solution set of MSTC problem is the intersection of the feasible solution sets of both MST and MSP (or MWSP). Constraints (4.35), (4.36) and (4.38) together with objective function (4.34) form MST problem formulation. Constraints (4.37) and (4.38) together with the maximization objective (4.34) form MWSP formulation (or when all edge weights are set to one, they form MSP formulation). The objective function (4.34) aims to minimize the total edge weight of the spanning tree. Constraint (4.35) ensures that exactly $|V(G)| - 1$ edges should be included in the spanning tree. Constraints (4.36) state that the solution space is constrained by the cycle breaking constraints. Constraints (4.37) enforce the conflict conditions between edge pairs. Finally, constraints (4.38) are the integrality restrictions.

Alternatively, the MSTC problem can also be formulated as another BILP model (MSTC-star) [16], where conflict constraints (4.37) are replaced with the following star constraints [43]:

$$d_C(e)x_e + \sum_{f \in N_C(e)} x_f \leq d_C(e) \quad e \in E(G), \quad (4.39)$$

where $N_C(e)$ is the neighborhood of $e \in E(G)$ in conflict graph C and $d_C(e) = |N_C(e)|$.

Furthermore, MSTC-clique formulation can be obtained after replacing conflict constraints (4.37) with the set of maximal clique constraints [42]:

$$\sum_{e \in K} x_e \leq 1 \quad K \in \mathcal{K}(C), \quad (4.40)$$

where $\mathcal{K}(C)$ is the set of maximal cliques of the conflict graph C . The relations between the polytopes of the LP relaxations of MSTC-gen, MSTC-star, and MSTC-clique, i.e., fractional MSTC-gen, MSTC-star, and MSTC-clique polytopes, are $P_{CLIQUE} \subseteq P_{GEN} \subseteq P_{STAR}$. This follows from the fact that clique inequalities (4.40) are facet defining [42], and star inequalities (4.39) can be obtained by aggregating conflict inequalities (4.37) for an edge $e \in E(G)$ over their conflicting edges $N_C(e)$. MSTC-clique is the strongest formulation with the smallest LP-relaxation optimal value. However,

the formulation has an exponential number of maximal clique constraints. As a result, it is not included for benchmarking in our computational tests.

4.3.2. Binary Integer Quadratic Programming Formulations

The MSTC problem can also be formulated as a quadratic MST problem (QMST) [44]. In this formulation, a $|E(G)| \times |E(G)|$ quadratic weight matrix $\mathbf{Q} = (q_{ef})$ is defined:

$$q_{ef} = \begin{cases} c_e, & \text{if } e = f, \\ M, & \text{if } \{e, f\} \in E(C), e \neq f, \\ 0, & \text{if } \{e, f\} \notin E(C), e \neq f, \end{cases}$$

where M is a sufficiently large number. QMST is formulated as:

$$\text{QMST: } \min \sum_{e \in E(G)} \sum_{f \in E(G)} q_{ef} x_e x_f \quad (4.41)$$

$$\text{s.t. } \sum_{e \in E(G)} x_e = |V(G)| - 1 \quad (4.42)$$

$$\sum_{e \in \gamma_G(S)} x_e \leq |S| - 1 \quad S \subset V(G), |S| \geq 2 \quad (4.43)$$

$$x_e = 0, 1 \quad e \in E(G). \quad (4.44)$$

Constraints (4.42), (4.43) and (4.44) are equivalent to MST-sp, whereas the objective function (4.41) is a quadratic objective function incurring a large penalty cost for each conflicting edge pair and minimizing the total edge weights. Exact and heuristic solution algorithms for QMST are proposed in [45].

4.3.3. Combinatorial Optimization Formulation

The COP version of the MSTC problem is described by the family of feasible sets $\mathcal{F}_{CFST} = \{H_1, H_2, \dots, H_k\}$, which is a collection of conflict-free spanning trees (CFSTs) in G , and an objective function of finding a minimum weight CFST. Then, the MSTC problem can be formulated as:

$$\min \left\{ \sum_{e \in E(H)} c_e : H \in \mathcal{F}_{CFST} \right\}. \quad (4.45)$$

We can define the polytope of the LP relaxations of CFSTs as the intersection of the LP relaxations of polytopes of spanning trees in G and stable sets in C , namely, $P_{CFST}(G, C) \equiv P_{ST}(G) \cap P_{FSTAB}(C) \equiv P_{ST}(G) \cap P_{CSTAB}(C) \cap P_{KSTAB}(C)$ [17]. As a result, the addition of conflict constraints (4.37), (4.39), or (4.40) to any valid MST formulation results in a valid MSTC formulation.

4.3.4. Reference Formulation

As previously mentioned, the most reasonable formulation of the cycle breaking constraints in terms of the number of variables and constraints is the one with single commodity formulation, MST-scf. Hence, it is selected as the reference formulation in this thesis:

$$\text{MSTC-scf:} \quad \min \sum_{e \in E(G)} c_e x_e \quad (4.46)$$

$$\text{s.t.} \quad \sum_{e \in E(G)} x_e = |V(G)| - 1 \quad (4.47)$$

$$\sum_{\{i,j\} \in \delta_D^+(s)} f_{ij} - \sum_{\{j,i\} \in \delta_D^-(s)} f_{ji} = |V(G)| - 1 \quad (4.48)$$

$$\sum_{\{j,i\} \in \delta_D^-(v)} f_{ji} - \sum_{\{i,j\} \in \delta_D^+(v)} f_{ij} = 1 \quad v \in V(G), v \neq s \quad (4.49)$$

$$f_{ij} + f_{ji} \leq (|V(G)| - 1)x_e \quad e = \{i, j\} \in E(G) \quad (4.50)$$

$$f_{ij} \geq 0 \quad \{i, j\} \in E(G) \quad (4.51)$$

$$x_e + x_f \leq 1 \quad \{e, f\} \in E(C) \quad (4.52)$$

$$x_e = 0, 1 \quad e \in E(G). \quad (4.53)$$

The objective function (4.46) aims to minimize the total edge weight of the spanning tree. Constraint (4.47) ensures that exactly $|V(G)| - 1$ edges should be included in the spanning tree. Constraint (4.48) makes sure that the amount of flow leaving the source vertex is exactly $|V(G)| - 1$. Constraints (4.49) are the flow balance constraints which guarantee that one commodity is delivered to each vertex. Constraints (4.50)

establish a connection between the flow variables $\mathbf{f}$ and edge variables $\mathbf{x}$. Constraints (4.51) ensure the nonnegativity of flow on all edges. Constraints (4.52) are the conflict constraints. Finally, constraints (4.53) are the integrality restrictions. This formulation has a polynomial number of constraints. The LP relaxation of MSTC-scf is obtained where constraints (4.53) are replaced with (4.9). We refer to the MSTC-scf formulation as the STRONG formulation. A weaker version, which we call the WEAK formulation, can be obtained by replacing constraints (4.52) with the star constraints (4.39) .



5. MAJOR COMPONENTS OF BRANCH-AND-BOUND ALGORITHMS

We start this chapter by describing a generic BB frame in order to clearly identify the major components of the novel algorithms. Then, the major components are introduced in the subsequent sections.

5.1. A Generic Branch-and-Bound Frame

We let $MSTC^{(t)}$ be the subproblem at node t of the BB search tree. $MSTC^{(t)}$ is identified by subsets $I^{(t)}$, $X^{(t)}$ and $F^{(t)}$, which essentially partitions the edge set $E(G)$. They are respectively the edges that are *included* in and *excluded* from a conflict-free MST, and *free* edges to be decided to be included or excluded. In other words $I^{(t)}$ can be treated as a partial feasible solution and $F^{(t)} = E(G) \setminus (I^{(t)} \cup X^{(t)})$ is the subset of edges that are candidate to be added in either to $I^{(t)}$ or $X^{(t)}$, and $X^{(t)}$ is the subset of edges that cannot be in any conflict-free MST. Note that the edges in $I^{(t)}$ do not conflict with the edges in $I^{(t)}$ and $F^{(t)}$. However, edges in $F^{(t)}$ can have conflicts between themselves according to the conflict structure represented by $C[F^{(t)}]$. The partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of the edges $E(G)$ partitions naturally the vertices $V(C)$ of the conflict graph as well, since $E(G) \equiv V(C)$. The optimal solution of $MSTC^{(t)}$ is represented with $H^{(t)}$ and its optimal objective value is denoted with $\bar{z}^{(t)}$. $MST^{(t)}$ stands for a relaxation for $MSTC^{(t)}$ with optimal objective value $\underline{z}^{(t)}$ and optimal solution $T^{(t)} = (V(T^{(t)}), E(T^{(t)}))$, which is not necessarily a feasible solution of $MSTC^{(t)}$. H^* is the current best feasible solution or the *incumbent* whose objective value gives the best upper bound $\bar{z}$. Starting with the original problem $MSTC^{(0)}$, we solve $MST^{(t)}$ at node t of the BB tree. Obviously, an optimal solution $T^{(t)}$ can violate the relaxed constraints of $MSTC^{(t)}$, i.e., conflict constraints. In fact, if $T^{(t)}$ is a feasible solution of $MSTC^{(t)}$, then it is a feasible solution of $MSTC$ and $c(E(T^{(t)}))$ yields an upper bound for the optimal value of $MSTC$. Figure 5.1 describes an outline of a generic BB procedure.

Algorithm 1 Generic BB Algorithm for the MSTC Problem

Input: A graph $G = (V(G), E(G))$ with edge weights $c_e, e \in E(G)$ and its conflict graph $C = (V(C), E(C))$;

Output: A minimum conflict free spanning tree H^* and its weight z^* ;

```

1: begin
2: (Initialization) : Set  $t = 0, \bar{z} \leftarrow \infty, MSTC^{(0)} \leftarrow MSTC, I^{(0)} \leftarrow \{\emptyset\}, X^{(0)} \leftarrow \{\emptyset\}, F^{(0)} \leftarrow E(G),$ 
    $G^{(0)} \leftarrow G, C^{(0)} \leftarrow C$ ;
3: Solve MST relaxation  $MST^{(0)}$ , which is the MST problem on  $G$ ;
4: if  $MST^{(0)}$  is feasible then
5:   Let  $T^{(0)} = (V(T^{(0)}), E(T^{(0)}))$  be its optimal solution and its value  $\underline{z}^{(0)}$ ;
6:   if  $T^{(0)}$  is conflict free then
7:      $T^{(0)}$  is an optimal solution of MSTC;
8:      $H^* \leftarrow T^{(0)}$  and  $\bar{z} = \underline{z}^{(0)}$ , go to 40;
9:   else
10:    Set  $\mathcal{L} \leftarrow \{(MSTC^{(0)}, T^{(0)}), \underline{z}^{(0)}\}$ ;
11:   end if
12: else
13:   STOP, MSTC is infeasible;
14: end if
15: (Termination test): If  $\mathcal{L} = \emptyset$  then go to 40;
16: (Problem selection): Delete subproblem  $MSTC^{(t)}$  with the smallest lower bound value  $\underline{z}^{(t)}$  and
   its relaxed optimal solution  $T^{(t)} = (V(T^{(t)}), E(T^{(t)}))$  from  $\mathcal{L}$ ;
17: (Division): Choose branching edges and create a division using the information  $MST^{(t)}$  provides
   (i.e.,  $I^{(t)}, X^{(t)}, F^{(t)}, T^{(t)}$  and  $\underline{z}^{(t)}$ ). Let  $MST^{(ti)}, i = 1, 2, \dots, d^{(t)}$  be the resulting subproblems
   obtained from  $MSTC^{(t)}$ ;
18: for  $i = 1, 2, \dots, d^{(t)}$  do
19:   (Filtering): If  $MST^{(ti)}$  can be feasible and simple lower bound  $LB^{(ti)} \leq \bar{z}$ , then try to reduce
   the problem size;
20:   Solve the MST relaxation  $MST^{(ti)}$  of the reduced problem;
21:   if  $MST^{(ti)}$  is infeasible then
22:     (Pruning by infeasibility): go to 15;
23:   else
24:     Let  $T^{(ti)}$  be its optimal solution with value  $\underline{z}^{(ti)}$ ;
25:     if  $T^{(ti)}$  is conflict free then
26:        $T^{(ti)}$  is a feasible solution of MSTC;
27:       if  $\underline{z}^{(ti)} < \bar{z}$  then
28:          $H^* \leftarrow T^{(ti)}$  and  $\bar{z} = \underline{z}^{(ti)}$ ;
29:       (Cleaning): Delete all problems with lower bound larger than or equal to  $\bar{z}$  from  $\mathcal{L}$ ;
30:     end if
31:   else
32:     if  $\underline{z}^{(ti)} < \bar{z}$  then
33:       Add  $(MSTC^{(ti)}, T^{(ti)}, \underline{z}^{(ti)})$  to  $\mathcal{L}$ ;
34:     else
35:       (Pruning by bound): go to 15;
36:     end if
37:   end if
38: end if
39: end for
40: STOP, the incumbent solution  $H^*$  that yields  $\bar{z}$  is an optimal solution of MSTC, return  $H^*$  and
    $z^* = \bar{z}$ ;
41: end

```

Figure 5.1. Generic BB Algorithm for MSTC Problem.

One of the key aspects of the BB algorithms is their effort on keeping the active node list $\mathcal{L}$ as short as possible, which consists of subproblems with sizes as small as possible through inexpensive filtering operations realized right after the generation of the subproblems.

5.2. Subproblems and Their Relaxations

A feasible solution of subproblem $MSTC^{(t)}$ is a conflict-free spanning tree of the spanning subgraph $G^{(t)} = (V(G^{(t)}), E(G^{(t)}))$ with $V(G^{(t)}) = V(G)$ and $E(G^{(t)}) = I^{(t)} \cup F^{(t)}$, determined according to the conflict relations represented by the induced conflict subgraph $C^{(t)} = C[F^{(t)}]$. It must include the edge subset $I^{(t)}$. In fact, it is a conflict free spanning tree of G including the subset $I^{(t)}$ and a conflict-free subset of $(|V(G)| - 1) - |I^{(t)}|$ edges from $F^{(t)}$, i.e., a stable set of size $(|V(G)| - 1) - |I^{(t)}|$ of $C^{(t)}$. A feasible solution of the relaxed subproblem $MST^{(t)}$ is also a spanning tree of G including the subset $I^{(t)}$; but the subset of $(|V(G)| - 1) - |I^{(t)}|$ edges from $F^{(t)}$ is allowed to have conflicting pairs. The optimal values $\bar{z}^{(t)}$ and $\underline{z}^{(t)}$ of $MSTC^{(t)}$ and $MST^{(t)}$ yield respectively upper and lower bounds on the optimal value of MSTC. $MST^{(t)}$ is not exactly the ordinary minimum spanning tree problem on $G^{(t)}$ since an optimal solution is restricted to include $I^{(t)}$. In an optimal solution $T^{(t)} = (V(T^{(t)}), E(T^{(t)}))$ of $MST^{(t)}$, conflicts can exist only between the edges that are selected from $F^{(t)}$. It should be kept in mind that for some $e \in F^{(t)}$, $I^{(t)} \cup \{e\}$ can have a cycle although $I^{(t)} \cup \{e\}$ is conflict free, which must be prevented through additional controls.

Except the initial MST relaxation, i.e., $MST^{(0)}$ problem, which is solved using Kruskal's [2] or Prim's algorithm [1], the relaxations are not solved from scratch at search nodes. Their optimal solution are obtained by updating the relaxed optimal solutions at the parent node instead using dynamic tree operations, which are insertions and deletions. They are explained in detail in the following chapter.

Kruskal's algorithm grows a spanning acyclic structure, which is a forest consisting of disjoint subtrees throughout the iterations. In the classical implementation of

the algorithm, edges $e \in E(G)$ are sorted in increasing order, and an edge that does not create an elementary cycle is added to the forest until $|V(G)| - 1$ edges are included, resulting in an MST of graph G . As an improvement over the classical implementation, a disjoint set abstract data structure (ADS) is used.

However Prim's algorithm grows a tree, which is not necessarily spanning. In the naïve implementation, the tree starts with an arbitrary vertex chosen as the root vertex r , and it grows until all vertices of the input graph are included. The resulting tree with $|V(G)| - 1$ edges is an MST for the input graph. As an improvement over the naïve implementation, a priority queue is used and implemented as a binary min-heap.

5.3. Subproblem Selection

The subproblem with the smallest lower bound value in the active node list is chosen. It gives priority to subproblems which will be divided anyway. With this rule, the subproblems with worse lower bounds can be eliminated quickly. We use a priority queue to store the active search nodes, which is implemented using a heap data structure.

5.4. Branching Rules

We consider two types of branching rules: corrective and preventive. A corrective rule aims to eliminate conflicts existing in a relaxed optimal solution, i.e., an MST. However, a preventive rule aims to construct an MST preventing any conflict in it.

5.4.1. Corrective Branching Rules

Three different conflict-based branching rules are considered, in which a dichotomy on the conflict pair is present: *pair branching*, *edge branching* and *clique branching*.

In pair branching, a conflicting edge pair $\{e, f\}$ is selected. Then in each branch, one of the edges is excluded. As can be observed, only the excluded edge subsets are affected using one of the edges only, say edge e .

In edge branching, given a conflicting edge pair $\{e, f\}$, two branches are created. In the first branch, edge e is included, $I^{(t)} \leftarrow I^{(t)} \cup \{e\}$ and all edges that are in conflict with it are excluded, $X^{(t)} \leftarrow X^{(t)} \cup N_C(e)$, and deleted from $F^{(t)}$ with e , $F^{(t)} \leftarrow F^{(t)} \setminus (N_C(e) \cup \{e\})$. This means considerable reduction in the size of the newly generated subproblem in this branch. In the second branch, edge e is excluded, $X^{(t)} \leftarrow X^{(t)} \cup \{e\}$. This time, both the included and excluded edge subsets are affected in one branch, while only the excluded edge subset is affected in the other.

In clique branching, a maximal clique K including a given conflicting edge pair $\{e, f\}$ is found on the conflict graph. Then, K is divided into two subsets with equal sizes, i.e., $|K_1| = |K_2|$, $e \in K_1, f \in K_2$, or almost equal if K is odd. In the first branch, all edges in the first clique subset are excluded, $X^{(t)} \leftarrow X^{(t)} \cup K_1$. In the second branch, all edges in the second clique subset are excluded, $X^{(t)} \leftarrow X^{(t)} \cup K_2$. We make use of a greedy heuristic to determine a maximal clique on the conflict graph during clique branching, which is described with Figure 5.2 below. Again, there is no change in the included edges as it is the case in pair branching.

The performance of corrective branching rules on BB algorithms are analyzed in [35] for the assignment problem with conflicts, in [46] for the shortest path problem with conflicts, and in [47] for the minimum cost flow problem with conflicts. These studies show that the best performance of the BB algorithm is achieved with edge branching, which is followed by clique branching and pair branching. Our preliminary results are in parallel with the existing works. The best performing branching rule is edge branching, as any subproblem $MSTC^{(t)}$ is solved on subgraph of G induced by edges $I^{(t)} \cup F^{(t)}$, i.e., $G[I^{(t)} \cup F^{(t)}]$, and $|F^{(t)}|$ reduces more rapidly compared to pair and clique branching rules. The worst performing branching rule is pair branching from the same perspective. The performance of clique branching rule is in between.

Algorithm 2 Greedy heuristic to determine a maximal clique

Input: A graph $G = (V(G), E(G))$ and a vertex subset S to be included in the clique;

Output: A maximal clique K including vertices in S ;

```

1: begin
2: Set  $K \leftarrow S$  and put in  $S$  vertices in  $V(G)$  that are adjacent with all vertices in  $S$ ;
3: while  $S \neq \emptyset$  do
4:   Select a vertex of  $S$  and call it  $u$ ;
5:    $K \leftarrow K \cup \{u\}$ ;
6:    $S \leftarrow S \setminus \{u\}$ ;
7:   Delete from  $S$  all vertices that are not adjacent with  $u$ ;
8: end while
9: return  $K$ , which is a maximal clique including  $S$ ;
10: end

```

Figure 5.2. Heuristic to determine a maximal clique including a given vertex subset.

5.4.2. Preventive Branching Rules

A preventive rule can be based on the stable sets of the conflict graph since they represent conflict-free subsets of edges of C , as done for the solution of MCP [48] and MWSP [49]. This rule allows the creation of more than two child subproblems using the information provided by the stable sets of conflict subgraph $C^{(t)}$.

As can be observed, when a free edge e is included, i.e., $I^{(t)}$ is set to $I^{(t)} \cup \{e\}$, free edges conflicting with it, i.e., $N_{C^{(t)}}(e)$, are deleted from $F^{(t)}$ together with e , which can cause considerable reduction in subproblem sizes. This is the main motivation of this branching rule, which is based on the following simple proposition. We say a stable set of the conflict graph C is *acyclic* if its elements, which are in fact the edges of G , do not contain a cycle of G .

Proposition 5.1. *Any acyclic stable set of conflict graph C of size $|V(G)| - 1$ is a conflict-free spanning tree of G .*

Proof. Since the vertices of conflict graph C are the edges of G , i.e., $V(C) \equiv E(G)$, then the elements of an acyclic stable set represent an acyclic subset of size $|V(G)| - 1$

of the edges of G , which is a spanning tree of G by definition. Besides, the elements are the vertices of the conflict graph C that are not adjacent and they form a conflict-free subset of the edges of G as a result, and the assertion holds. $\square$

A natural consequence of this proposition is the size of a maximum cardinality acyclic stable set, which is $|V(G)| - 1$. Hence, MSTC problem becomes equivalent to finding an acyclic stable set of the conflict graph with minimum total weight and size $|V(G)| - 1$. As a result, the branching rule we propose, which is based on the following simple proposition, considers only acyclic maximal stable sets and tries to detect first the ones with maximum cardinality. Then, it narrows the search space to surround the ones with the smallest total weight amongst them.

Proposition 5.2. *Let $U \subset V(C)$ be a maximal acyclic stable set of conflict graph C , then an optimal solution of MSTC problem must include a vertex from the set $V(C) \setminus U$.*

Proof. There are two cases. If $|U| < |V(G)| - 1$, then any maximal acyclic stable set of the conflict graph with larger size must include one of the vertices in $V(C) \setminus U$. If $|U| = |V(G)| - 1$, then U is a feasible solution of MSTC and if there exists an acyclic stable set of size $|V(G)| - 1$ with lower weight, which is a conflict free spanning tree, then it must include one of the vertices in $V(C) \setminus U$ again. Besides, $|U| > |V(G)| - 1$ is not possible since this contradicts the assumption that U is acyclic. $\square$

According to this proposition, if one searches for a stable set better than U , either with a larger size or smaller weight, then there are $|V(C)| \setminus U$ potential branching options. It is possible to decrease the number of options by considering a vertex maximal conflict subgraph $C[W]$ with $W \subseteq V(C)$ and $U \subset W$ depending on whether $|U| < |V(C)| - 1$ or $|U| = |V(C)| - 1$.

The following proposition identifies W as a vertex maximal subset of $V(C)$ with respect to the property that U is a maximum cardinality stable set of the subgraph of C induced by W .

Proposition 5.3. *Let $|U| < |V(G)| - 1$ and let $W \subseteq V(C)$ be such that $U \subset W$ and $C[W]$ is a vertex maximal conflict subgraph induced by W with the property that $\alpha(C[W]) = |U|$. Then, a maximal cycle free stable set of C with larger size must include one of the vertices in $V(C) \setminus W$.*

Proof. By definition, U is a maximum (cardinality) stable set of the conflict subgraph induced by W , i.e., $\alpha(C[W]) = |U|$. Besides, W is maximal with respect to this property. Hence, a larger sized stable set of C must include a vertex of $V(C) \setminus W$. $\square$

This proposition implies a decrease of $|W| - |U|$ in the number of branching options to be considered. In fact, the larger its size is, the better W is. A potential approach is to let W be the union of $|U|$ disjoint cliques of C each including a vertex of U . This can be done by a simple heuristic given in Figure 5.3, which grows $|U|$ cliques each being initialized with a vertex of U , by trying to add vertices from $V(C) \setminus U$ to them one by one, if they are adjacent with the other vertices of the considered clique.

Algorithm 3 Heuristic to determine a maximal subset of a given maximal stable set.

Input: Conflict graph $C = (V(C), E(C))$ and a stable set $U \subseteq V(C)$;

Output: A maximal subset W where $U \subseteq W \subseteq V(C)$ and $\alpha(C[W]) = |U|$;

```

1: begin
2: Create  $|U|$  cliques  $K_j = \{\emptyset\}$   $j = 1, 2, \dots, |U|$  and a vertex list  $Q = V(C) \setminus U$ ;
3:  $j = 0$ ;
4: for  $v \in U$  do
5:    $j \leftarrow j + 1$ ;
6:    $K_j \leftarrow K_j \cup \{v\}$ ;
7: end for
8: while  $Q \neq \emptyset$  do
9:   Choose a vertex  $v \in Q$ 
10:   $Q \leftarrow Q \setminus \{v\}$ ;
11:  if there exists a clique  $K_q$  whose vertices are all adjacent with  $v$  (i.e.,  $K_q \subseteq N_C(v)$ )
12:    then
13:       $K_q \leftarrow K_q \cup \{v\}$ ;
14:    end if
15: end while
16:  $W = \bigcup_{j=1}^{|U|} K_j$  is a maximal subset of  $V(C)$  such that  $\alpha(C[W]) = |U|$ , return  $W$ ;
17: end

```

Figure 5.3. Heuristic to determine a maximal subset of a given maximal stable set.

For the situation when $|U| = |V(G)| - 1$, maximal subset W can be obtained through reverse optimization. First of all, such U is acyclic, and it consists of $|V(G)| - 1$ edges of G , which is a spanning tree. It is also conflict free since U is a stable set of the conflict graph, which makes $\mathbf{x} = \{x_e = 1 \text{ for } e \in U \text{ and } x_e = 0 \text{ for } e \in E(G) \setminus U\}$ a feasible solution of the MSTC problem. It is also a feasible solution for the LP relaxation of MST-sp formulation, which yields integer-valued optimal solutions, as $P_{ST} \equiv P_{SP}$. Then, if $\mathbf{x}$ is an optimal solution of this LP, the reduced costs of the edges relative to U are all nonnegative, i.e., $\bar{c}_e \geq 0$ for $e \in E(G)$ with $\bar{c}_e = 0$ for $e \in U$. However, if this is not true, then $W \subset E(G)$ would be a maximal subset of $E(G)$ with $\bar{c}_e \geq 0$ for $e \in W$. In other words, $G[W]$ is the maximal subgraph of G whose MST is $G[U]$. This discussion can be summarized with the following proposition.

Proposition 5.4. *If U is an acyclic stable set with $|U| = |V(G)| - 1$, then $W \subset V(G)$ is such that $\bar{c}_e \geq 0$ for $e \in W$ where $\bar{c}_e$ is the reduced cost of $e \in E(G)$ relative to the spanning tree of G presented by U , i.e., $G[U]$.*

When Kruskal's Greedy Algorithm is used to find an MST, one can determine the optimal dual solution of the LP MST-sp. Each edge e that is added to the tree defines a connected component with vertices Q_e , which is obtained by taking the union of the two disjoint subsets of vertices each including an endpoint of e . Moreover, at some later stage in the algorithm Q_e is combined with one or more other vertices to define a larger component by adding another edge f to the tree. Then, the dual variable associated with Q_e is defined as $\pi_{Q_e} = c_f - c_e$. Notice that $\pi_{Q_e} \geq 0$ since $c_f \geq c_e$ by Kruskal's selection rule. The values of the dual variables associated with the other subsets of $V(G)$ are set to zero, except $\pi_{V(G)} = -c_g$, where g is the last added edge by Kruskal's algorithm. As a result, after putting the edges of U in the increasing order of their weights one can determine the subsets Q_e for $e \in U$ and dual values π_{Q_e} with which reduced costs can be calculated as

$$\bar{c}_e = c_e + \sum_{\{U \subseteq V(G) : e \in \gamma(U), |U| \geq 2\}} \pi_U \quad (5.1)$$

for each edge $e \in E(G)$. Then, the edges with nonnegative reduced costs belong to

W , since they do not destroy the optimality of the spanning tree $G[U]$ with respect to induced subgraph $G[W]$.

A greedy heuristic for finding a maximal stable set in a conflict graph is used instead of solving MCP on $C^{(t)}$ exactly, since this is an $\mathcal{NP}$ -hard problem [50]. The maximal stable set greedy heuristic algorithm for determining branching edges is outlined with Figure 5.4.

Algorithm 4 Greedy heuristic to determine a maximal stable set and branching edges

Input: Graph $G = (V(G), E(G))$ and conflict graph with vertices sorted in increasing order of conflict degrees $C' = (V(C'), E(C'))$, H^* , $\bar{z}$, the partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$;

Output: The set of branching edges B based on the extended maximal stable set U ;

```

1: begin
2: Set  $U, W, B \leftarrow \emptyset$ ,  $F' \leftarrow F^{(t)}$ 
3: for  $e \in V(C')$  do
4:   if  $e \in F'$  then
5:     if  $(I^{(t)} \cup U) \cup \{e\}$  is acyclic then
6:       Set  $U \leftarrow U \cup \{e\}$ ;
7:       Set  $F' \leftarrow F' \cup (N_{C'}(e) \cup \{e\})$ ;
8:     end if
9:   end if
10: end for
11: if  $|U| = |V(G)| - 1 - |I^{(t)}|$  then
12:   if  $c(I^{(t)}) + c(U) < \bar{z}$  then
13:     Set  $H^* \leftarrow (V(G), I^{(t)} \cup U)$  and  $\bar{z} = c(I^{(t)}) + c(U)$ ;
14:   end if
15:   Set  $W \leftarrow U$ 
16: else
17:   Call Algorithm 3 to determine a maximal subset  $W$  of  $V(C[F^{(t)}])$ ;
18: end if
19: for  $e \in F^{(t)} \setminus W$  do
20:   if  $I^{(t)} \cup \{e\}$  is acyclic then
21:      $B \leftarrow B \cup \{e\}$ ;
22:   end if
23: end for
24: Free memory and return  $B$ ;
25: end

```

Figure 5.4. Greedy heuristic to determine a maximal stable set and branching edges.

The heuristic iteratively selects edges while ensuring that the selected edges do not form conflicts. The vertices of the conflict graph, which correspond to the edges

of the original graph, are sorted in the order of increasing degrees only once at the beginning. The ordering is done only once on C , as doing it every time on $C^{(t)}$ is rather inefficient. Then, the edge with the lowest degree e is added to the stable set U , which must not cause a cycle of G with the edges that belong to the union of the stable set and included set, i.e., $U \cup I^{(t)}$.

If $|U| < (|V(G)| - 1) - |I^{(t)}|$, W is constructed using Algorithm 3 as the union of $|U|$ disjoint cliques of $C^{(t)}$ each including an element of U . After the process is over, the branching edges are the edges that are not in W , which must not form a cycle with the included edge set $I^{(t)}$.

On the other hand, when the size of the union of the stable set and included set is equal to the number of edges of a spanning tree, i.e., $|U| = (|V(G)| - 1) - |I^{(t)}|$, then a feasible solution is found, and we set $W \equiv \{e \in F^{(t)} \setminus U : \bar{c}_e < 0\}$. The reduced costs are calculated with the dual values of the original graph G relative to the spanning tree $G[I^{(t)} \cup U]$. In fact, it is not necessary to determine W explicitly; any edge $e \in F^{(t)} \setminus U$ with negative reduced cost can be used to branch.

In our implementation, we consider only the situation when $|U| < (|V(G)| - 1) - |I^{(t)}|$ since the construction of W can be done efficiently, and this is the situation faced almost at every step. However, $|U| = (|V(G)| - 1) - |I^{(t)}|$ can occur rarely and the determination of edges with negative reduced costs requires considerable computational effort. As a result, we prefer to simply set $W \equiv U$ when $|U| = (|V(G)| - 1 - |I^{(t)}|)$ at a search node.

5.4.3. Illustration of Branching Rules

In order to demonstrate the branching rules described in this section, a sample graph $G = (V(G), E(G))$ is generated where $|V(G)| = 7$ and $|E(G)| = 12$, which is illustrated with Figure 5.5.

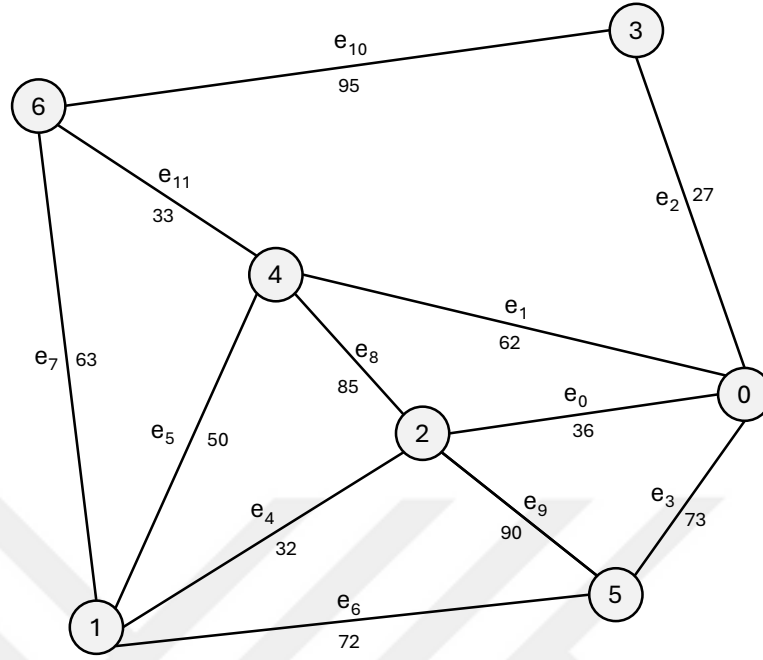


Figure 5.5. Sample graph.

Also, a sample conflict graph $C = (E(G), E(C))$ is generated where $|E(C)| = 9$ as depicted with Figure 5.6.

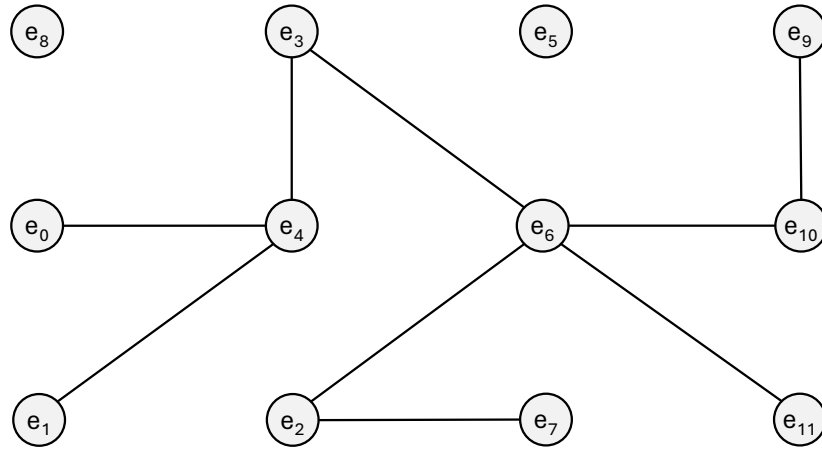


Figure 5.6. Sample conflict graph.

An optimal MST of G consists of the edges $E(T^*) = \{e_0, e_2, e_4, e_5, e_6, e_{11}\}$ with total weight $c(E(T^*)) = 250$. However, it is infeasible since $\{e_0, e_4\}, \{e_2, e_6\}, \{e_6, e_{11}\} \in E(C)$. An optimal MST and violated conflict pairs are illustrated in Figure 5.7.

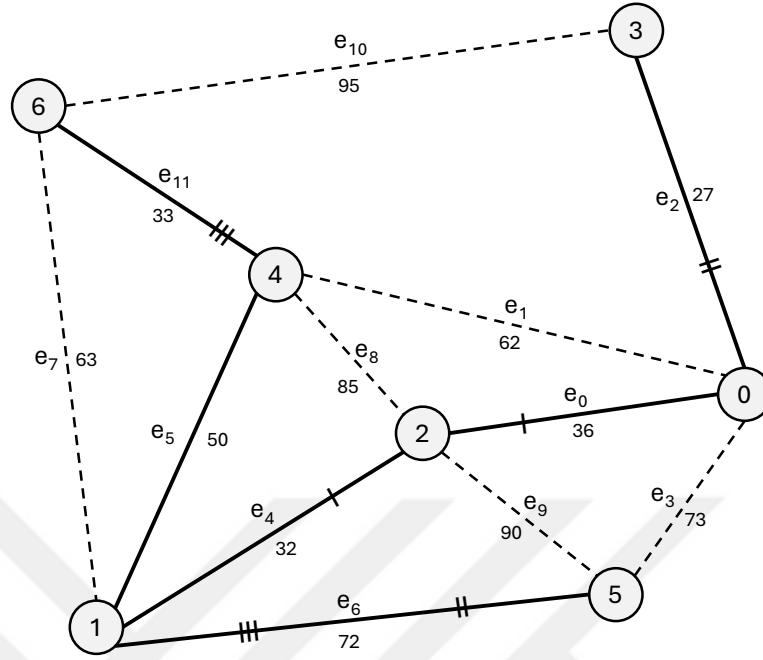


Figure 5.7. Optimal solution of the MST with violated conflict pairs.

An optimal solution to the MSTC instance is obtained with edges $E(H^*) = \{e_0, e_1, e_2, e_3, e_5, e_{11}\}$ and total weight $c(E(H^*)) = 281$, as shown in Figure 5.8.

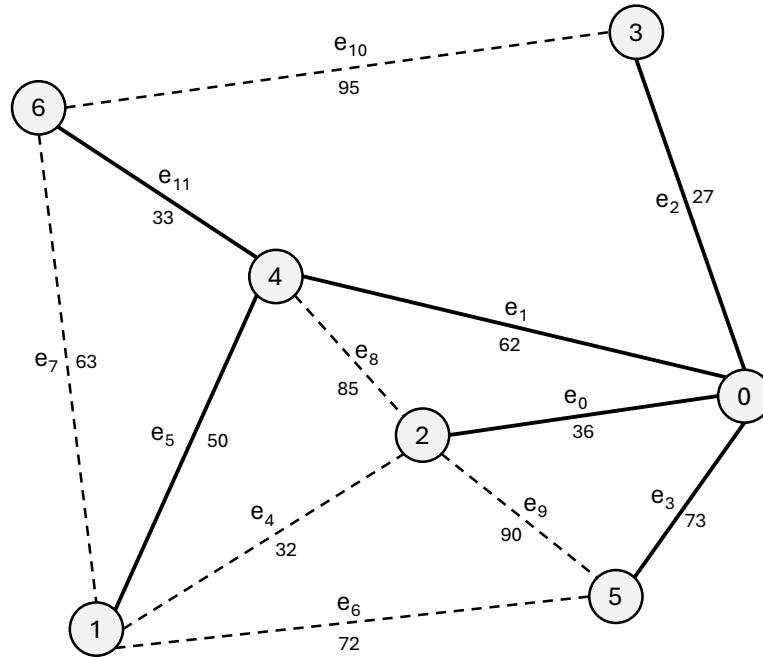


Figure 5.8. Optimal solution of the MSTC instance.

The BB algorithm with pair branching rule is applied to the MSTC instance with its graph and conflict graph illustrated with Figure 5.5 and Figure 5.6, respectively. The complete BB search tree with pair branching rule is given with Figure 5.9.

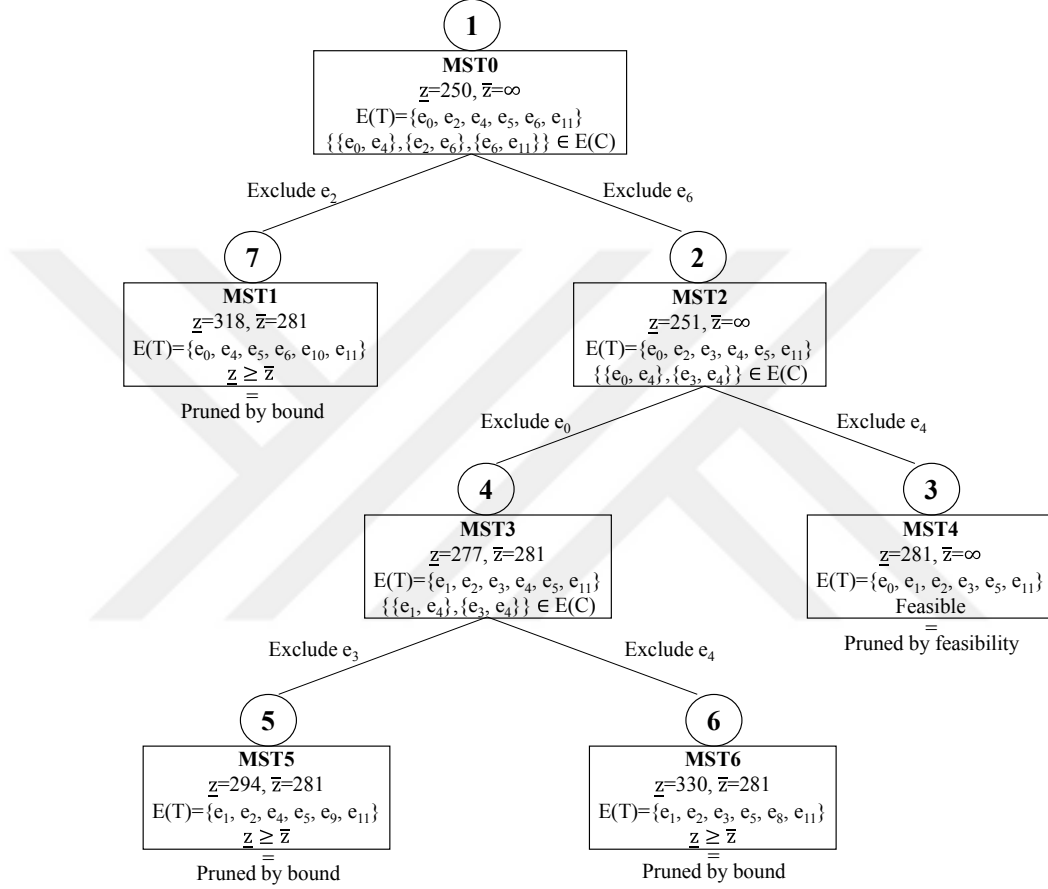


Figure 5.9. BB search tree for the MSTC instance with pair branching rule.

There are seven nodes in the BB search tree when pair branching rule is used. Using edge branching rule, the complete BB search tree is illustrated with Figure 5.10. There are five nodes in the BB search tree, when edge branching is used.

Since there is no clique in the conflict graph as clearly depicted with Figure 5.6, clique branching rule yields the same BB tree obtained by using pair branching rule as described with Figure 5.9.

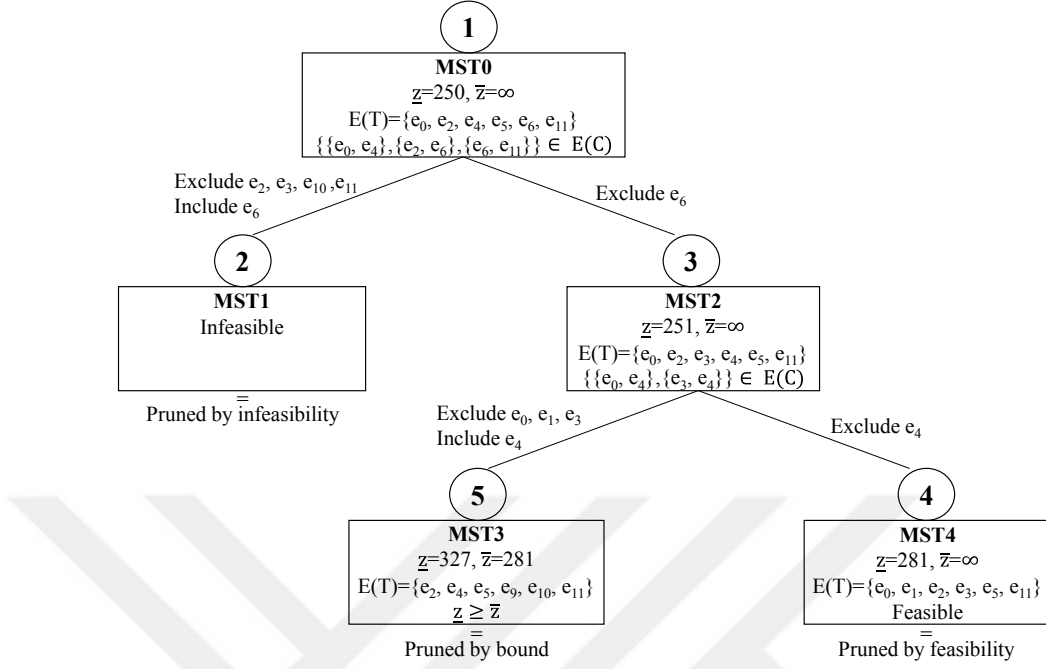


Figure 5.10. BB search tree for the MSTC instance with edge branching rule.

We can conclude from the numerical example that pair branching rule may result in same MSTC subproblems at different levels of the BB tree, so its performance is worse than edge and clique branching rules. Clique branching rule performs better for conflict graphs including cliques with size larger than two, however the problem of repeated MSTC subproblems still exist for this branching rule. The best performance can be obtained by edge branching, as it allows to eliminate subproblems of the BB tree in a single node.

The BB algorithm with stable-set based branching rule is applied to the MSTC instance with its graph and conflict graph illustrated with Figure 5.5 and Figure 5.6, respectively. The complete BB search tree with stable-set based branching rule is given with Figure 5.11. There are sixteen nodes in the BB search tree when stable-set based branching is used. Note that the search tree is wider compared to corrective branching rules. Its advantage is in the size of the subproblems: they are considerably smaller since edges conflicting with the branching edge are deleted.

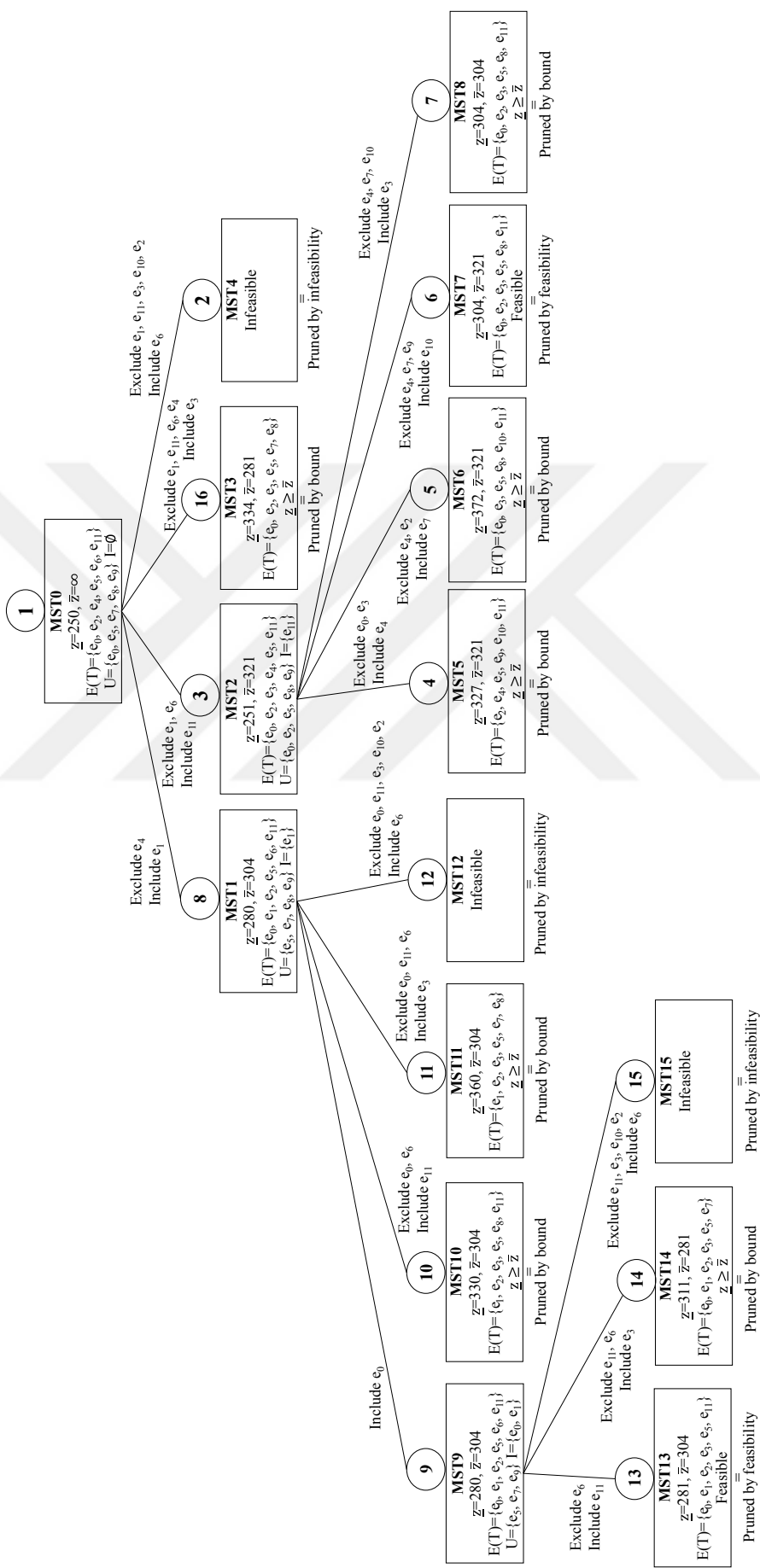


Figure 5.11. BB search tree for the MSTC instance with stable-set based branching rule.

5.5. Filtering

Filtering aims to reduce the size of the active node list and the number of active problems that are included in it. It consists of three sequential operations:

- (i) Infeasibility test for fast pruning by infeasibility,
- (ii) Simple lower bounds for fast pruning by bound,
- (iii) Probing for reducing the size of the problems by deciding on the useless edges.

All these operations are applied on subproblems after their generation, prior to edge removal and tree restoration operations, and addition to the active node list $\mathcal{L}$.

5.5.1. Infeasibility Test

Given a graph $G = (V(G), E(G))$, we can define the stability, vertex cover, matching and edge cover numbers associated with it as

$$\alpha(G) = \max \{|U| : U \text{ is a stable set}\}, \quad (5.2)$$

$$\tau(G) = \max \{|\mathcal{T}| : \mathcal{T} \text{ is a vertex cover}\}, \quad (5.3)$$

$$\nu(G) = \max \{|M| : M \text{ is a matching}\}, \quad (5.4)$$

$$\rho(G) = \max \{|\Psi| : \Psi \text{ is an edge cover}\}. \quad (5.5)$$

It is possible to show that U is a stable set if and only if $V(G) \setminus U$ is a vertex cover. Besides, $\alpha(G) \leq \rho(G)$ and $\nu(G) \leq \tau(G)$ are obvious inequalities.

Theorem 5.5. (*Gallai's theorem [51, 52]*) *If $G = (V(G), E(G))$ is a graph without isolated vertices, then*

$$\alpha(G) + \tau(G) = |V(G)| = \nu(G) + \rho(G).$$

Proposition 5.6. *G does not have a conflict free spanning tree including the edges of $I^{(t)}$ if one of the following conditions hold:*

- i. $G^{(t)}$ has an isolated vertex,
- ii. $G^{(t)}$ has at least two leaves that are not in $I^{(t)}$ and incident with conflicting edges,
- iii. $|X^{(t)}| > |E(G)| - (|V(G)| - 1)$,

- iv. $\alpha(C^{(t)}) < (|V(G)| - 1) - |I'|$,
- v. $\tau(C^{(t)}) > (|E(G)| - |X'|) - (|V(G)| - 1)$,
- vi. $|X^{(t)}| > |E(G)| - (|V(G)| - 1) + (r - \sum_{j=1}^r |K_j^{(t)}|)$, where $K_1^{(t)}, K_2^{(t)}, \dots, K_r^{(t)}$ is a clique partitioning of C ,
- vii. $\rho(C^{(t)}) < (|V(G)| - 1) - |I^{(t)}|$,
- viii. $\nu(G^{(t)}) > (|E(G)| - |X^{(t)}|) - (|V(G)| - 1)$.

Proof.

- i. It is not possible to cover all the vertices of G by a spanning tree including the edges of $I^{(t)}$.
- ii. Clearly any such edge that connects a leaf vertex must be in a spanning tree. If two of such edges are in conflict then one of the leaves cannot be in a conflict free spanning tree and thus a conflict free spanning tree of G does not exist.
- iii. If $|F^{(t)}| < (|V(G)| - 1) - |I^{(t)}|$, then subgraph $G^{(t)}$ cannot have a, possibly disconnected, spanning subgraph of $G^{(t)}$ that has $(|V(G)| - 1) - |I^{(t)}|$ edges and forms a spanning tree of G when combined with the edges in $I^{(t)}$. The right hand side is the number of edges that have to be included in $I^{(t)}$ in order to obtain a spanning tree of G . Then, the condition directly follows since $|F^{(t)}| = |E(G)| - |I^{(t)}| - |X^{(t)}|$.
- iv. Since any conflict-free spanning tree of G is a stable set of C , this inequality implies that the stable sets of $C^{(t)}$ are not large enough to include enough number of conflict-free edges whose union with $I^{(t)}$ form a conflict-free spanning tree of G [16].
- v. As a consequence of Gallai's theorem $\alpha(C^{(t)}) + \tau(C^{(t)}) = |V(C^{(t)})|$, from which $\alpha(C^{(t)}) = |F^{(t)}| - \tau(C^{(t)})$ follows since $E(G^{(t)}) \equiv V(C^{(t)}) \equiv F^{(t)}$. Then, $|F^{(t)}| - \tau(C^{(t)}) < (|V(G)| - 1) - |I^{(t)}|$ by (iii) and $|F^{(t)}| = |E(G)| - |I^{(t)}| - |X^{(t)}|$ by definition, and the condition is obtained [16].
- vi. If $K_1^{(t)}, K_2^{(t)}, \dots, K_r^{(t)}$ is a clique partitioning of C , then any vertex cover must select at least $|K_j^{(t)}| - 1$ vertices from $|K_j^{(t)}|$ for each j . Thus, $\sum_{j=1}^r |K_j^{(t)}|$ is a lower bound of a minimum vertex cover of $C^{(t)}$ and the condition follows from (v).
- vii, viii. They follow as consequences of the obvious inequalities $\alpha(G) \leq \rho(G)$ and $\nu(G) \leq \tau(G)$ respectively from (iv) and (v).

□

Notice that for (iv) and (v), we can use any upper bound on $\alpha(C^{(t)})$ and $\rho(C^{(t)})$, and for (v) and (viii), we can use any lower bound on $\tau(C^{(t)})$ and $\nu(C^{(t)})$, which can be computed more efficiently than the originals as they require solving $\mathcal{NP}$ -hard problems. Clearly, (iv) holds if and only if (v) does, and (vii) holds if and only if (viii) does.

The infeasibility detection procedure we devise, provided in Figure 5.12, starts with $I^{(t)}$, $X^{(t)}$, $F^{(t)} \equiv E(G) \setminus (I^{(t)} \cup X^{(t)})$, and enlarges $I^{(t)}$ and $X^{(t)}$ with the edges deleted from $F^{(t)}$. Eventually, it stops either by finding a feasible solution of $MSTC^{(t)}$, or by detecting its infeasibility if $G^{(t)}$ does not have any conflict free spanning tree, or without any conclusion with possibly enlarged $I^{(t)}$ and $X^{(t)}$, and reduced $F^{(t)}$.

Algorithm 5 Infeasibility test

Input: The partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$, subgraphs $G^{(t)} = (V(G^{(t)}), E(G^{(t)}))$ and $C^{(t)} = (V(C^{(t)}), E(C^{(t)}))$ with $V(G^{(t)}) \equiv V(G)$, $E(G^{(t)}) = I^{(t)} \cup F^{(t)}$, and $C^{(t)} \equiv C[F^{(t)}]$;
Output: An infeasibility decision or a feasible solution if possible;

```

1: begin
2: Generate all edges in  $F^{(t)} \subseteq E(G^{(t)})$  that are connected to a leaf and let  $L$  be the list of
   them;
3: if  $L \neq \emptyset$  then
4:   if  $G^{(t)}$  and  $C^{(t)}$  do not pass the tests of Proposition 5.6 then
5:      $MSTC^{(t)}$  is infeasible, return current  $I^{(t)}$ ,  $X^{(t)}$ ,  $F^{(t)}$ ,  $G^{(t)}$ ,  $C^{(t)}$  and STOP;
6:   else
7:     while  $L \neq \emptyset$  do
8:       Choose an edge  $e$  from  $L$ ;
9:        $L \leftarrow L \setminus e$ ;
10:      Set  $I^{(t)} \leftarrow I^{(t)} \cup \{e\}$ ,  $X^{(t)} \leftarrow X^{(t)} \cup N_{C^{(t)}}(e)$ ,  $F^{(t)} \leftarrow F^{(t)} \setminus (N_{C^{(t)}}(e) \cup l)$ ;
11:       $E(G^{(t)}) \leftarrow E(G^{(t)}) \setminus N_{C^{(t)}}(e)$ ;
12:       $C^{(t)} \leftarrow C[F^{(t)}]$ ;
13:    end while
14:    if  $|I^{(t)}| = |V(G)| - 1$  then
15:       $MSTC^{(t)}$  is feasible and  $I^{(t)}$  is the edge set of a conflict free spanning tree;
16:      Return current  $I^{(t)}$ ,  $X^{(t)}$ ,  $F^{(t)}$ ,  $G^{(t)}$ ,  $C^{(t)}$  and STOP;
17:    else
18:      Go to 4;
19:    end if
20:  end if
21: else
22:  No decision about  $MSTC^{(t)}$ , return current  $I^{(t)}$ ,  $X^{(t)}$ ,  $F^{(t)}$ ,  $G^{(t)}$ ,  $C^{(t)}$  and STOP;
23: end if
24: end

```

Figure 5.12. Infeasibility detection.

If there is no conflict free spanning tree of G that includes the edges of $I^{(t)}$, then $MSTC^{(t)}$ is infeasible. This can be seen by checking one of the conditions listed in Proposition 5.6. They all determine whether conflict subgraph $C^{(t)} = C[F^{(t)}]$ has a stable set of size at least $(|V(G)| - 1) - |I^{(t)}|$. The elements of this stable set together with the elements of $I^{(t)}$ form a conflict free edge subset of G of size at least $|V(G)| - 1$. Clearly, if such stable set does not exist, i.e., all stable sets of C that includes $I^{(t)}$ has size strictly less than $|V(G)| - 1$, then a conflict free spanning tree of G including the edges of $I^{(t)}$ does not exist either.

$G^{(t)}$ and $C^{(t)}$ are respectively the subgraphs of G and C as defined above. Notice that $d_{G^{(t)}}(e) = 1$ for all $e \in L$, i.e., L consists of the free edges in $F^{(t)}$ connecting leaves to $G^{(t)}$. Hence, their addition to $I^{(t)}$ do not create a cycle. Again because of this property, any such edge must be in an optimal solution, which makes a feasible $I^{(t)}$ also optimal for initial $MSTC^{(t)}$.

We apply the infeasibility detection procedure to subproblem $MSTC^{(ti)}$ as soon as it is created during the division of its parent $MSTC^{(t)}$, after initializing with $I^{(ti)}$, $X^{(ti)}$ and $F^{(ti)}$ with $F^{(ti)} \equiv E(G) \setminus (I^{(ti)} \cup X^{(ti)})$. In our implementation we consider only the first three conditions of Proposition 5.6 at every pass from line 4, since they computationally cost almost none; but we cannot say the same for the remaining ones even a heuristic is used to compute upper bounds on $\alpha(C^{(ti)})$ and $\rho(C^{(ti)})$, and lower bounds on $\tau(C^{(ti)})$ and $\nu(C^{(ti)})$.

5.5.2. Simple Lower Bounds

At each search node t , given a set of included edges $I^{(t)}$ of subproblem $MSTC^{(t)}$, then it is possible to compute lower bounds by one of the following propositions. Although they are not stronger than the one obtained by solving the relaxed subproblem $MST^{(t)}$, they can be calculated in seconds.

Proposition 5.7.

$$LB = c(I^{(t)}) + ((|V(G)| - 1) - |I^{(t)}|) c_{\min}, \quad (5.6)$$

where $c_{\min} = \min_{e \in F^{(t)}} \{c_e\}$, is a lower bound on the optimal value of subproblem $MSTC^{(t)}$.

Proof. Recall that the edges in $I^{(t)}$ are not in conflict with neither each others nor any edge in $F^{(t)}$. Besides, $I^{(t)}$ needs $(|V(G)| - 1) - |I^{(t)}|$ additional conflict free edges from $F^{(t)}$ to become a conflict free spanning tree of G . As can be seen, total weight of a conflict free minimum spanning tree of $G^{(t)}$ cannot be smaller than $((|V(G)| - 1) - |I^{(t)}|) c_{\min}$. $\square$

The following trivial proposition is for tightening (5.6) with an additional computational cost of ordering the edges in $F^{(t)}$ according to their increasing weights.

Proposition 5.8. *The upper bound (5.6) can be made tighter by replacing the second term in (5.6) with the sum of the $(|V(G)| - 1) - |I^{(t)}|$ smallest of the weights of the edges in $F^{(t)}$.*

Proof. Trivial. $\square$

Let subproblem $MSTC^{(ti)}$ correspond to the partition $(I^{(ti)}, X^{(ti)}, F^{(ti)})$ of edges $E(G)$, generated on the i th branch. Let $J^{(ti)} \subseteq E(T^{(t)})$ be the edges deleted from $T^{(t)}$, and $M^{(ti)} \subseteq F^{(ti)}$ with $|M^{(ti)}| = |J^{(ti)}|$ be the $|M^{(ti)}|$ smallest weight edges of $F^{(ti)}$. Then, it is possible to compute a tighter lower bound as a consequence of the following proposition.

Proposition 5.9.

$$LB^{(ti)} = \underline{z}^{(t)} - \sum_{e \in J^{(ti)}} c_e + \sum_{e \in M^{(ti)}} c_e. \quad (5.7)$$

is a lower bound on the optimal value of subproblem $MSTC^{(t)}$.

Proof. In order to obtain the optimal value $\underline{z}^{(ti)}$ of the MST relaxation $MST^{(ti)}$ of subproblem $MSTC^{(ti)}$, the edges in $J^{(ti)}$ must be replaced with $|J^{(ti)}|$ edges from free edges $F^{(ti)}$ while keeping the acyclic structure of $T^{(t)}$. However, this restriction is relaxed in the calculation of lower bound (5.7) and the edges in $M^{(ti)}$ are allowed to create cycles. Besides, they are the $|J^{(ti)}|$ smallest weighted edges in $F^{(ti)}$. Therefore, $LB^{(ti)} \leq \underline{z}^{(ti)}$, which is less than the optimal value of $MSTC^{(ti)}$. $\square$

We use the lower bound Proposition 5.9 suggests in the BB algorithm after dividing a parent problem for eliminating some of the newly generated child subproblems for which the infeasibility test is indecisive. In short, this simple bound can cause the pruning by bound of a subproblem without solving its relaxation in seconds. Since we keep the edges in the increasing order of their weights from the beginning, re-ordering of the free edges is not required, which makes the calculations even more efficient.

5.5.3. Probing

Reducing the size of a subproblem is an important issue that effects the performance of the BB algorithm. An edge is forced to be included in a conflict-free MST. If the problem turns out to be infeasible under this condition, then this edge must not appear in a feasible solution and we must delete the edge permanently from the subgraph. Similarly, an edge is forced to be excluded in a conflict-free MST. If the problem turns out to be infeasible under this condition, then this edge must appear in a feasible solution and we must delete all its conflicting edges from the subgraph.

This procedure can be activated for any subproblem $MSTC^{(ti)}$ after the division of $MSTC^{(t)}$, and for every free edge in $F^{(ti)}$, before its addition to the active node list $\mathcal{L}$. Algorithm 5 is used to check the infeasibility after fixing an edge (i.e., adding an edge to $I^{(ti)}$ after deleting it from $F^{(ti)}$). A feasible solution may be obtained while checking the infeasibility, which gives a chance to update the upper bound and incumbent.

Notice that when a free edge is included, it must not create a cycle in the set of the included edges. Hence, two cycle checks at lines 8 and 18 seem necessary. Fortunately, the second one is redundant and dismissed in our implementation. This fact is a consequence of the characteristics of the infeasibility test. If it detects an infeasibility, then whether $I' \cup \{e\}$ is acyclic is not important. If it ends up with a feasible solution, i.e., a conflict free spanning tree, then $I' \cup \{e\}$ is acyclic. Finally, if it is indecisive, then the last added edge e is deleted from I' and the cycle created with its addition to I' is broken at line 29. The probing heuristic is outlined in Figure 5.13.

Algorithm 6 Probing

Input: The partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$;

Output: A reduced partition (I', X', F') of $MSTC'$ or a feasible solution if possible;

```

1: begin
2: Set  $I' \leftarrow I^{(t)}$ ,  $X' \leftarrow X^{(t)}$  and  $F' \leftarrow F^{(t)}$ ;
3: Set  $C' \leftarrow C[F^{(t)}]$ ;
4: for any free edge  $e \in F'$  do
5:   Set  $X' \leftarrow X' \cup \{e\}$ ,  $F' \leftarrow F' \setminus \{e\}$ ;
6:   Run the infeasibility test procedure Algorithm 5 with  $(I', X', F')$ ;
7:   if  $(I', X', F')$  is infeasible then
8:     if  $I' \cup \{e\}$  is acyclic then
9:       Set  $I' \leftarrow I' \cup \{e\}$ ,  $X' \leftarrow (X' \setminus \{e\}) \cup N_{C'}(e)$ ,  $F' \leftarrow F' \setminus (N_{C'}(e) \cup \{e\})$ ;
10:    end if
11:  else if  $|I'| = |V(G)| - 1$  then
12:    if  $c(I') < \bar{z}$  then
13:      Set  $I^* \leftarrow I'$  and  $\bar{z} = c(I')$ ;
14:    end if
15:  else
16:    No decision;
17:    Set  $X' \leftarrow X' \setminus \{e\}$ ,  $F' \leftarrow F' \cup \{e\}$ ;
18:    if  $I' \cup \{e\}$  is acyclic then
19:      Set  $I' \leftarrow I' \cup \{e\}$ ,  $X' \leftarrow X' \cup N_{C'}(e)$ ,  $F' \leftarrow F' \setminus (N_{C'}(e) \cup \{e\})$ ;
20:      Run the infeasibility test procedure Algorithm 5 with  $(I', X', F')$ ;
21:      if  $(I', X', F')$  is infeasible then
22:        Set  $I' \leftarrow I' \setminus \{e\}$ ,  $X' \leftarrow (X' \setminus (N_{C'}(e) \cup \{e\})) \cup \{e\}$ ,  $F' \leftarrow F' \cup (N_{C'}(e) \cup \{e\})$ ;
23:      else if  $|I'| = |V(G)| - 1$  then
24:        if  $c(I') < \bar{z}$  then
25:          Set  $I^* \leftarrow I'$  and  $\bar{z} = c(I')$ ;
26:        end if
27:      else
28:        No decision;
29:        Set  $I' \leftarrow I' \setminus \{e\}$ ,  $X' \leftarrow X' \setminus N_{C'}(e)$ ,  $F' \leftarrow F' \cup (N_{C'}(e) \cup \{e\})$ ;
30:      end if
31:    end if
32:  end if
33: end for
34: Set  $I^{(t)} \leftarrow I'$ ,  $X^{(t)} \leftarrow X'$ ,  $F^{(t)} \leftarrow F'$ ;
35: Return updated  $I^{(t)}$ ,  $X^{(t)}$  and  $F^{(t)}$ ;
36: end

```

Figure 5.13. Probing heuristic.

Although, the number of cycle checks is reduced, still a remarkable computational effort especially for dense subgraphs can be required. This makes where and when to apply probing during the recourse of the BB algorithm a critical decision issue. Since

subgraphs on which subproblems are defined become smaller deeper in the search tree, we do not apply probing during the early stages.

For this purpose, we consider the ratio of the total number of the included and excluded edges over the number of edges of the graph, i.e., $\beta = (|I^{(t)}| + |X^{(t)}|)/|E(G)|$, and apply static probing when $\beta \geq 0.80$. Since $|E(G)| = |I^{(t)}| + |X^{(t)}| + |F^{(t)}|$, then $\beta = 1 - |F^{(t)}|/|E(G)|$, and $\beta \geq 0.80$ becomes equivalent to $|F^{(t)}|/|E(G)| \leq 0.20$. Another option is to use dynamic probing, where probing is applied depending on the conflict density $\Omega = 2|E(G)|/[|V(G)|(|V(G)| - 1)]$. If $\Omega \leq 0.15$, then probing is not applied as the conflict graph becomes very sparse. Probing would have a negative effect instead of any improvement for such cases, as our preliminary computational test suggests. If $\Omega > 0.15$, then probing is applied when $\beta \leq \Omega - 0.1$.

5.5.4. Illustration of Filtering Operations

To visualize the filtering operations, the sample MSTC instance (for which the graph and the conflict graph is previously depicted as Figure 5.5 and Figure 5.6, respectively) is solved with BB algorithm using stable set-based branching rule. All filtering operations are applied including infeasibility test, simple lower bounds, and probing (applied to all BB nodes for the sake of demonstration). The complete BB search tree with filtering operations is provided with Figure 5.14.

Compared with the version without filtering operations, previously given with Figure 5.11, the size of the BB search tree has decreased by half. There are five BB search tree nodes pruned by simple lower bounds. Infeasibility test reduces the size of relaxed subproblems MST4 and MST5 by extending the included edge set with the edges that are connected to a leaf vertex. In MST4, v_0 is a leaf vertex with $d_G(v_0) = 1$ as the edges $\{e_0, e_1, e_3\}$ are excluded and e_2 is the only edge that is connected to v_0 . Hence, e_2 is included and its conflicting edge e_7 is excluded. Similarly in MST5, v_3 is a leaf vertex with $d_G(v_3) = 1$ as e_2 is excluded, leading to the inclusion of e_{10} and exclusion of e_9 .

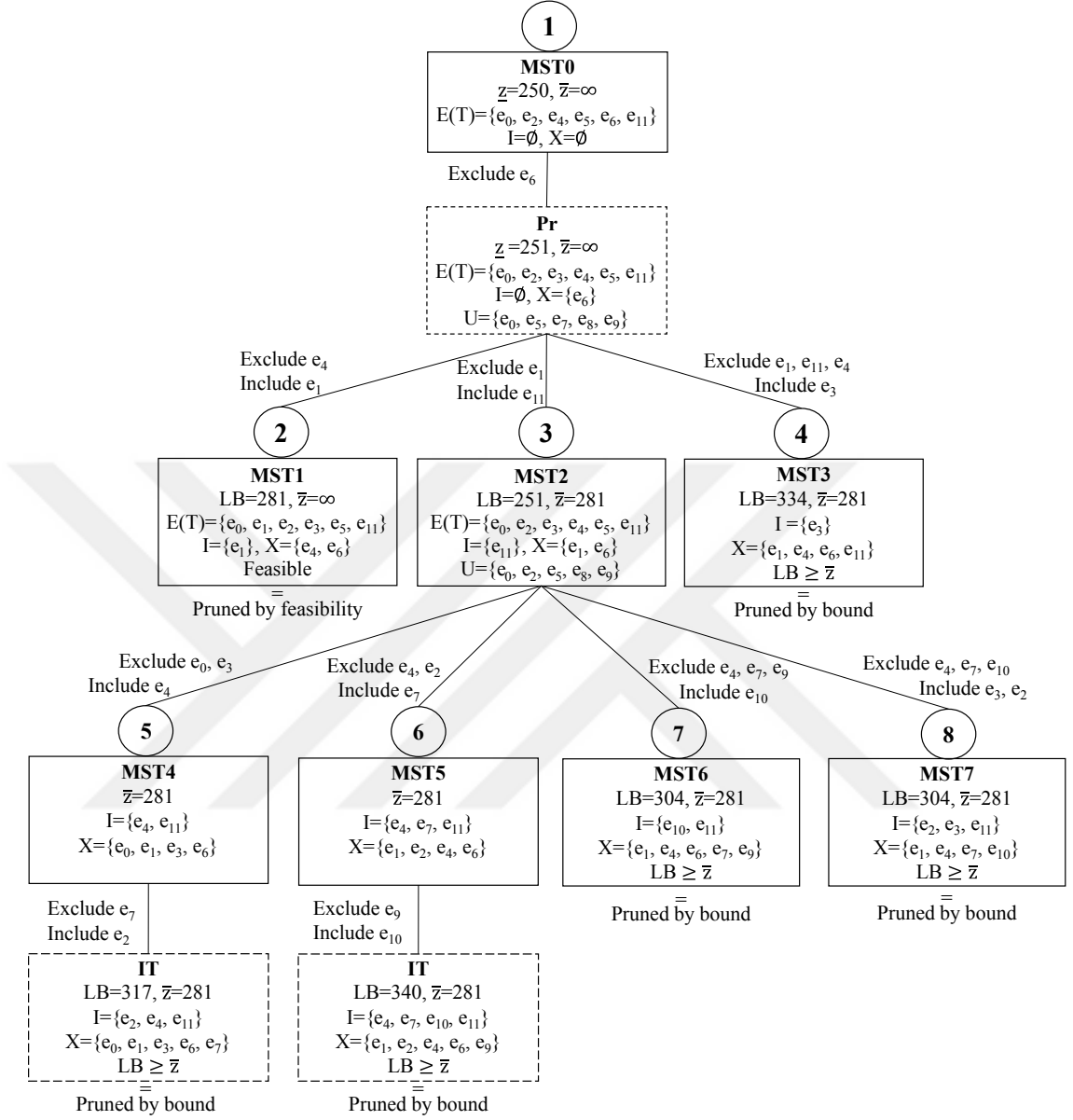


Figure 5.14. BB search tree for the MSTC instance with filtering operations.

Probing gives an exclusion decision for e_6 at the MST relaxation of the root node (MST0) since the inclusion of this edge results in infeasibility, as illustrated in Figure 5.15. When e_6 is included, its neighbor edges in the conflict graph C , i.e., $N_C(e_6) = \{e_2, e_3, e_{10}, e_{11}\}$, are excluded. Then, v_3 becomes an isolated vertex with $d_G(v_3) = 0$, and the subproblem turns out to be infeasible. As a result, e_6 cannot exist in a feasible solution and it is added to the set of excluded edges X . There are no other inclusion or exclusion decisions given by probing for the MSTC instance.

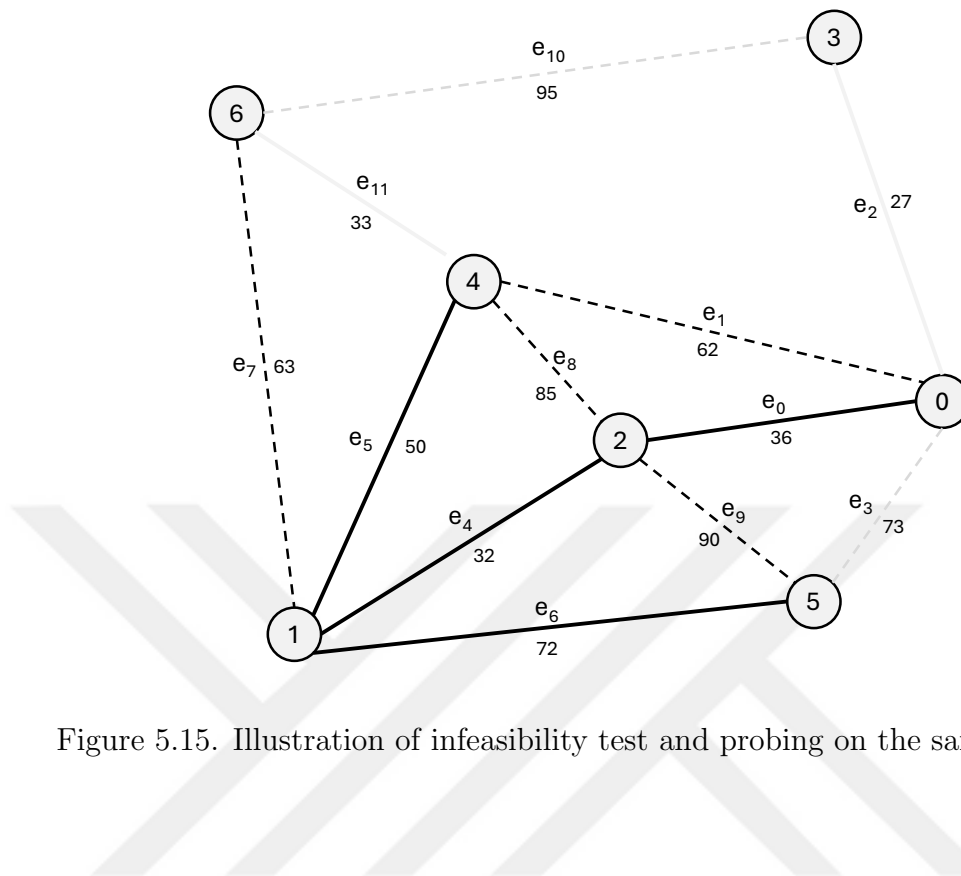


Figure 5.15. Illustration of infeasibility test and probing on the sample graph.

6. BRANCH-AND-BOUND ALGORITHMS

BB algorithms with edge branching rule (BB-E), diving based BB (BB-D), and stable set based BB (BB-S) are described in detail in this chapter. They all use previously introduced major components.

6.1. Edge Branching-based Branch-and-Bound Algorithm

BB-E algorithm systematically explores the space of potential solutions by branching on edges and pruning infeasible problems with the aid of infeasibility tests or suboptimal subproblems using simple lower bounds. BB-E is given in Figure 6.1.

The algorithm begins by solving the relaxed MST problem on the input graph G (ignoring conflicts) using Prim's algorithm and initializes the search with an empty set of the included and excluded edges. At each step, the algorithm selects an edge with the highest number of conflicts in the conflict graph, and branches into two subproblems: one where this edge is included and conflicting edges are excluded, and another where this edge is excluded. Infeasibility tests and lower bounds are used to prune infeasible and suboptimal subproblems, and probing techniques further reduce the problem size. The algorithm stops when until all subproblems are processed, at which point the best feasible solution is returned as an optimal solution for the MSTC problem.

When an edge is removed from the subproblem, a replacement edge needs to be found if it exists. After each deletion, the replacement edges are found from the set of free edges, which are sorted only once before the execution of the BB algorithm, until the tree is restored. If there is not enough edge, the related branch is pruned as the MSTC instance turns out to be infeasible. The next step is to find the most conflicted edge in the induced conflict subgraph, just before the subproblem $MSTC^{(t)}$ is added to the active node list $\mathcal{L}$. If there is no conflict between the edges of the relaxed optimal solution of $MSTC^{(t)}$, then $MSTC^{(t)}$ is feasible and the upper bound $\bar{z}$ is updated.

Algorithm 7 Edge branching based branch-and-bound (BB-E)

Input: A connected simple graph $G = (V(G), E(G))$ with edge weights c_e $e \in E(G)$, conflict graph $C = (V(C), E(C))$, probing threshold β ;

Output: A minimum conflict-free spanning tree H^* and its total weight z^* ;

```

1: begin
2: Set  $t = 0$ ,  $\bar{z} \leftarrow \infty$ ,  $\mathcal{L} \leftarrow \{\emptyset\}$ ,  $MSTC^{(0)} \leftarrow \text{MSTC}$ ,  $I^{(0)} \leftarrow \{\emptyset\}$ ,  $X^{(0)} \leftarrow \{\emptyset\}$ ,  $F^{(0)} \leftarrow E(G)$ ,  $G^{(0)} \leftarrow G$ ,  $C^{(0)} \leftarrow C$ ;
3: Order the edges according to their non-decreasing weights and save the order;
4: Solve MST relaxation  $MST^{(0)}$ , which is the MST problem on  $G$ ;
5: Let  $T^{(0)} = (V(T^{(0)}), E(T^{(0)}))$  be its optimal solution and its value  $\underline{z}^{(0)}$ ;
6: if  $T^{(0)}$  is conflict free then
7:   Set  $H^* \leftarrow T^{(0)}$  and  $\bar{z} \leftarrow \underline{z}^{(0)}$ ;
8: else
9:   Set  $\mathcal{L} \leftarrow \{(MSTC^{(0)}, T^{(0)}, \underline{z}^{(0)})\}$ ;
10: end if
11: while  $\mathcal{L} \neq \emptyset$  do
12:   Delete subproblem  $MSTC^{(t)}$  with the smallest lower bound value  $\underline{z}^{(t)}$  and its relaxed optimal solution  $T^{(t)} = (V(T^{(t)}), E(T^{(t)}))$  from  $\mathcal{L}$ ;
13:   Find a conflicting pair  $e, f \in E(T^{(t)})$  such that, say  $e$  has the the largest degree in  $C^{(t)}$ ;
14:   Generate subproblem  $MSTC^{(t1)}$  with  $I^{(t1)} \leftarrow I^{(t)} \cup \{e\}$ ,  $X^{(t1)} \leftarrow X^{(t)} \cup N_{C^{(t)}}(e)$ ,  $F^{(t1)} \leftarrow F^{(t)} \setminus (N_{C^{(t)}}(e) \cup \{e\})$ ;
15:   Generate subproblem  $MSTC^{(t2)}$  with  $I^{(t2)} \leftarrow I^{(t)}$ ,  $X^{(t2)} \leftarrow X^{(t)} \cup \{e\}$ ,  $F^{(t2)} \leftarrow F^{(t)} \setminus \{e\}$ ;
16:   for  $i = 1$  to  $2$  do
17:     Call Algorithm 5 to check the infeasibility of subproblem  $MSTC^{(ti)}$ ;
18:     if  $MSTC^{(ti)}$  is indecisive then
19:       Calculate simple lower bound  $LB^{(ti)}$ ;
20:       if  $LB^{(ti)} \leq \bar{z}$  then
21:         if  $|F^{(ti)}|/|E(G)| \leq 1 - \beta$  then
22:           Reduce the size of  $MSTC^{(ti)}$  by probing using Algorithm 6;
23:         end if
24:         Update  $T^{(t)}$  using Algorithms 11, 13, 16 to obtain  $T^{(ti)}$  and its value  $\underline{z}^{(ti)}$ ;
25:         if  $T^{(ti)}$  is a spanning tree of  $G^{(ti)}$  then
26:           if  $T^{(ti)}$  is conflict free then
27:             if  $\underline{z}^{(ti)} < \bar{z}$  then
28:               Set  $\bar{z} \leftarrow \underline{z}^{(ti)}$  and  $H^* \leftarrow T^{(t)}$ ;
29:             end if
30:             Delete all problems with lower bound larger than or equal to  $\bar{z}$  from  $\mathcal{L}$ ;
31:           end if
32:         else
33:           Set  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(MSTC^{(ti)}, T^{(ti)}, \underline{z}^{(ti)})\}$ ;
34:         end if
35:       end if
36:     end if
37:   end for
38:   Set  $t \leftarrow t + 1$ ;
39: end while
40: An optimal solution is reached, return  $H^*$  and  $z^* = \bar{z}$ ;
41: end

```

Figure 6.1. Edge branching based branch-and-bound algorithm (BB-E).

6.2. Diving-based Branch-and-Bound Algorithm

A possibility for the computation of a feasible solution that provides an upper bound on the optimal value of the MSTC problem is to use diving. This approach combines edge branching with depth first search (DFS) with the aid of infeasibility test, simple lower bounding, and probing. MST relaxations of subproblems are not solved. Here, there is no such need to keep an active node list $\mathcal{L}$. As a result, savings in bookkeepings of $\mathcal{L}$ is achieved.

A disjoint set abstract data structure (ADS) is created to keep track of connected components in the graph. Global upper bound and incumbent solution are updated during diving. Diving can also be used to solve the original MSTC problem optimally. However, it is a recursive procedure that repetitively calls subproblems at a deeper level.

6.2.1. Recursive Diving Algorithm

The diving function can be considered as a recursive backtracking algorithm, it starts with an empty tree and add edges to it one by one considering the conflicts. It checks at each step whether the current solution is better than the incumbent. If so, it updates the incumbent. If the current solution is not feasible, it backtracks. The process continues until the time limit is reached or an optimal solution is found. The diving algorithm is given in Figure 6.2.

Here, diving is implemented separately as a recursive function, and for each recursive call there exists a cycle check for $I' \cup \{e_i\}$. Using recursion can lead to stack overflow issues if the recursion depth is too large, which can be encountered in large graphs. Replacing recursion calls by an iterative approach might be helpful to avoid the risk of stack overflow. Also, iterative algorithms can offer more precise control over state management, which may improve performance for some cases.

Algorithm 8 Recursive Diving (Rec-D)

Input: Graph $G = (V(G), E(G))$ and conflict graph $C = (V(C), E(C))$, I^* , $\bar{z}$, the partition (I, X, F) of $MSTC$;

Output: An infeasibility decision or an optimal solution;

```

1: begin
2: Set  $(I', X', F') \leftarrow (I, X, F)$ ,  $V(G') \leftarrow V(G)$ ,  $E(G') \leftarrow I' \cup F'$ ,  $C' \leftarrow C[F']$ ;
3: Run the infeasibility test Algorithm 5 with  $I'$ ,  $X'$ ,  $F'$ ,  $G'$  and  $C'$  for  $MSTC'$ ;
4: if Problem  $MSTC'$  is feasible then
5:   Pruning by feasibility,  $I'$  is the edge set of a conflict free spanning tree, which is optimal
   as a consequence of infeasibility tests leaf consideration rule;
6:   if  $c(I') < \bar{z}$  then
7:     Set  $I^* \leftarrow I'$  and  $\bar{z} = c(I')$ ;
8:   end if
9:   Free memory and return current  $I'$ ,  $X'$ ,  $F'$ ,  $G'$  and  $C'$ ;
10: end if
11: if Problem  $MSTC'$  is infeasible then
12:   Pruning by infeasibility, free memory and return current  $I'$ ,  $X'$ ,  $F'$ ,  $G'$  and  $C'$ ;
13: end if
14: if No decision about the infeasibility of problem  $MSTC'$  then
15:   if  $|F'|/|E(G)| \leq 1 - \beta$  then
16:     Apply probing by interchanging lines (18-29) and (5-17) of Algorithm 6 to reduce
     the size of  $MSTC'$ ;
17:   end if
18:   Prepare new recursive calls for Diving;
19:   Order the edges in  $F'$  according to some criterion and let  $F' = \{e_1, e_2, \dots, e_p\}$  after
   the ordering. Ordering is not necessary;
20:   for  $i = 1$  to  $p$  do
21:     if  $I' \cup \{e_i\}$  is acyclic then
22:       Determine lower bound  $\underline{z}'$  for  $I' \cup e_i$ ;
23:       if  $\underline{z}' \leq \bar{z}$  then
24:         Set  $I_i \leftarrow I' \cup \{e_i\}$ ;
25:         Set  $X_i \leftarrow X' \cup (N_{C'}(e_i) \cup \{e_1, e_2, \dots, e_{i-1}\})$ ;
26:         Set  $F_i \leftarrow F' \setminus (N_{C'}(e_i) \cup \{e_1, e_2, \dots, e_i\})$ ;
27:         Call Diving with  $I_i$ ,  $X_i$ ,  $F_i$ ,  $G_i = (V(G), I_i \cup F_i)$  and  $C_i = C[F_i]$  for  $MSTC^{(i)}$ ;
28:       else
29:         Pruning by bound;
30:       end if
31:     else
32:       Pruning by infeasibility;
33:     end if
34:   end for
35: end if
36: Free memory and return current  $I'$ ,  $X'$ ,  $F'$ ,  $G'$  and  $C'$ 
37: end

```

Figure 6.2. Recursive diving algorithm (Rec-D).

6.2.2. Iterative Diving Algorithm

As an alternative approach, an iterative version of the diving algorithm (BB-D) to solve the MSTC problem from scratch is developed, which is called as diving-based branch-and-bound (BB-D). It uses diving strategy by exploring the solution space in a depth-first manner, making branching decisions based on edge inclusion or exclusion while managing conflicts. BB-D is outlined in Figure 6.3 below.

Algorithm 9 Diving-based branch-and-bound (BB-D)

Input: Graph $G = (V(G), E(G))$ and conflict graph $C = (V(C), E(C))$;

Output: A minimum weight conflict free spanning tree H^* , and its value z^* ;

```

1: begin
2: Order the edges  $e \in E(G)$  in a fixed sequence;
3: Set  $t \leftarrow 0$ ,  $I_1^{(t)}, X_1^{(t)}, I_2^{(t)}, X_2^{(t)} \leftarrow \emptyset$ ,  $F^{(0)} \leftarrow E(G)$ ,  $\bar{z} \leftarrow \infty$ ;
4: if  $F^{(t)} \neq \emptyset$  then
5:   Select the edge  $e \in F^{(t)}$  with the lowest rank in the order;
6:   Set  $I_1^{(t)} \leftarrow I_1^{(t)} \cup \{e\}$ ,  $X_1^{(t)} \leftarrow X_1^{(t)} \cup N_C(e)$ ,  $F^{(t)} \leftarrow F^{(t)} \setminus (\{e\} \cup N_C(e))$ ;
7: else
8:   if  $I_1^{(t)} \neq \emptyset$  then
9:     Select the edge  $e \in I_1^{(t)}$  with the highest rank in the order;
10:    Set  $I^{(t)} \leftarrow I_1^{(t)} \setminus \{e\}$ ,  $X_1^{(t)} \leftarrow (X_1^{(t)} \cup \{e\}) \setminus N_C(e)$ ,  $F^{(t)} \leftarrow F^{(t)} \cup N_C(e)$ ;
11:   else
12:     STOP, the incumbent solution  $H^*$  that yields  $\bar{z}$  is an optimal solution, output  $H^*$  and  $\bar{z}$ ;
13:   end if
14: end if
15: Run the infeasibility test Algorithm 5 with  $I_1^{(t)}, X_1^{(t)}, F^{(t)}$  to update  $I_2^{(t)}, X_2^{(t)}, F^{(t)}$ ;
16: if No decision about infeasibility then
17:   if  $|F^{(t)}|/|E(G)| \leq 1 - \beta$  then
18:     Apply probing by interchanging lines (18-29) and (5-17) of Algorithm 6 to reduce the size
       of the subproblem by updating  $I_2^{(t)}, X_2^{(t)}, F^{(t)}$ ;
19:   end if
20:   Compute a lower bound  $\underline{z}^{(t)}$ ;
21: else
22:   Set  $\underline{z}^{(t)} \leftarrow \infty$ ;
23: end if
24: if  $\underline{z}^{(t)} < \bar{z}$  then
25:   if  $I_1^{(t)} \cup I_2^{(t)}$  is acyclic then
26:     if  $|I_1^{(t)} \cup I_2^{(t)}| = |V(G)| - 1$  then
27:       if  $c(I_1^{(t)} \cup I_2^{(t)}) < \bar{z}$  then
28:         Set  $H^* \leftarrow I_1^{(t)} \cup I_2^{(t)}$  and  $\bar{z} = c(I_1^{(t)} \cup I_2^{(t)})$ ;
29:       end if
30:     end if
31:   end if
32: end if
33: Set  $t \leftarrow t + 1$ ,  $I_1^{(t)} \leftarrow I_1^{(t-1)}$ ,  $X_1^{(t)} \leftarrow X_1^{(t-1)}$ ,  $I_2^{(t)}, X_2^{(t)} \leftarrow \emptyset$ ,  $F^{(t)} \leftarrow F^{(t-1)}$  and go to 4;
34: end

```

Figure 6.3. Diving based branch-and-bound algorithm (BB-D).

By diving into the most promising branches first, the algorithm can reach feasible solutions quickly while using simple lower bounding using Proposition 5.8 and infeasibility test to prune unpromising branches. This approach efficiently narrows down the search space while ensuring the optimal solution is eventually found. In BB-D, an improvement is done when there is no free edge in a subproblem. Before backtracking, the last added edge to the included set is placed in the excluded set, which may result in more efficient backtracking and reducing the size of the solution space, so that less search nodes are needed to be explored.

6.2.3. Illustration of Diving Algorithms

The diving algorithms are demonstrated on the sample MSTC instance where its graph and conflict graph are previously illustrated with Figure 5.5 and Figure 5.6, respectively. The complete search tree for Rec-D algorithm (without applying infeasibility test and probing) on the sample MSTC instance is depicted with Figure 6.4.

In the root node, lower bound is 240, and all edges are free. Notice that an MST relaxation is not solved at the root node for diving algorithms. In the first node, edge e_2 is included, and the excluded set X is updated with conflicting edges $\{e_6, e_7\}$. The lower bound remains at 240, and the algorithm continues diving by selecting the next free edge. Similarly, in the second node, e_4 is included, and the free edges that are in conflict with e_4 are excluded, i.e., $\{e_0, e_3, e_1\}$. This results in an increased lower bound value of 317, prompting further exploration.

As the search process continues, the algorithm includes additional edges, such as e_{11} and e_5 . At each step, the algorithm checks for cycles and the number of free edges. If a cycle is formed in I , then the branch is not created at all. When there are not enough free edges, the branch is marked infeasible and pruned from the search. For example, when e_9 is included in some branches, the resulting subproblem is infeasible as there are not enough free edges to form an MST. This reduces the search space significantly, allowing the algorithm to focus on potentially optimal paths.

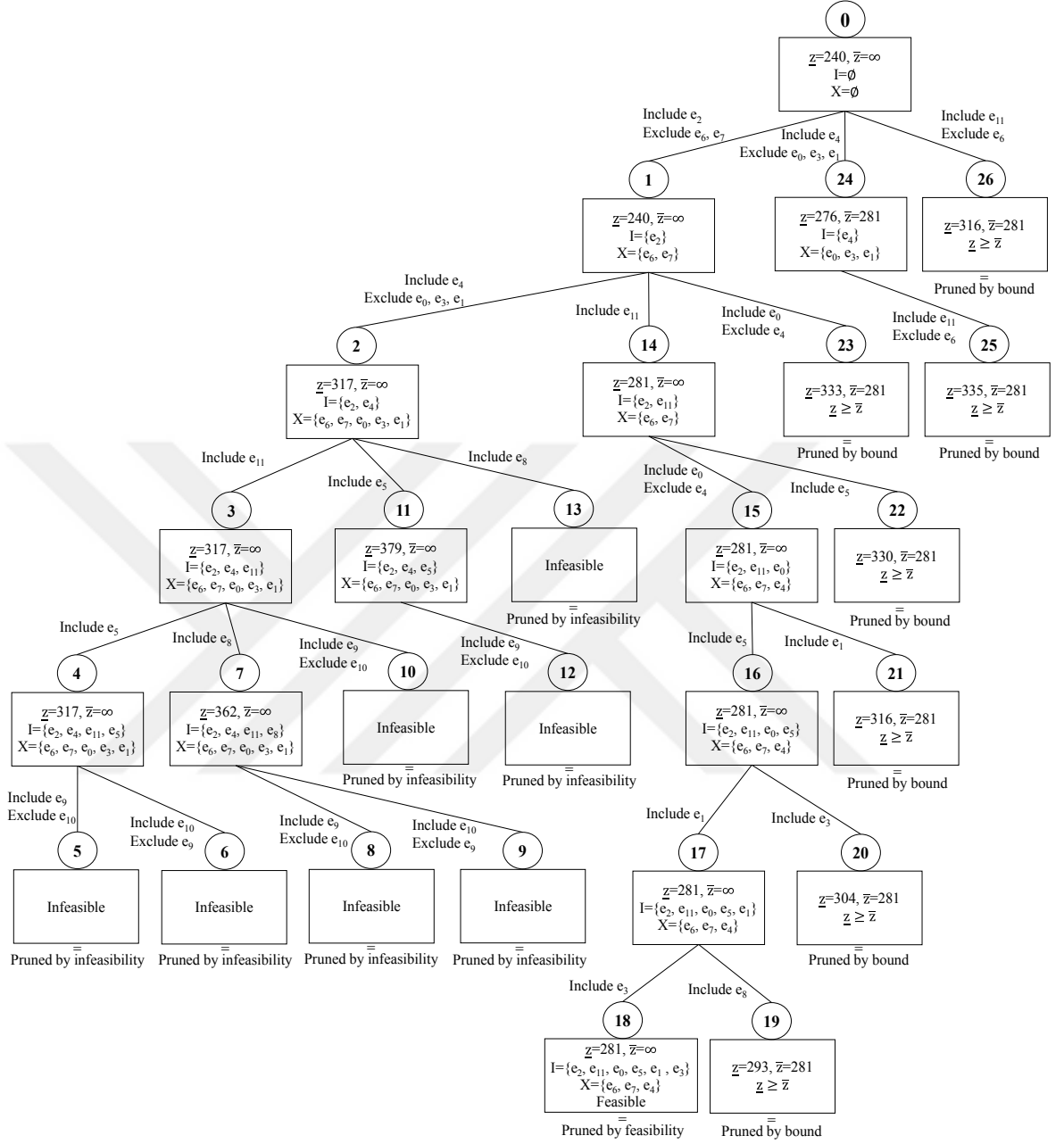


Figure 6.4. BB search tree for Rec-D algorithm.

When a feasible solution is found, the algorithm evaluates its total weight, and updates the incumbent solution if the total weight is less than the upper bound. In node 18, which is one of the deeper nodes, a feasible solution is found with a total weight of 281, and the incumbent solution is updated together with the upper bound. All subsequent branches with a lower bound greater than 281 are pruned, further optimizing the search process.

The diving approach allows the algorithm to efficiently handle large problem instances by focusing only on relevant portions of the solution space. This structured exploration and pruning enable the algorithm to quickly converge on optimal solutions without exhaustively enumerating all possibilities.

The search tree for BB-D is very similar with a difference: $X^{(t)}$ grows more rapidly compared to Rec-D when there are no remaining free edges to be explored. On the sample MSTC instance, BB-D terminates by exploring 25 nodes where Rec-D finds the optimal solution by exploring 26 nodes. The difference between the nodes explored by Rec-D and BB-D dramatically increases in favor of BB-D especially for large MSTC instances, which is also indicated by our preliminary results.

6.3. Stable Set-based Branch-and-Bound Algorithm

Stable set-based branch-and-bound algorithm (BB-S) is based on a preventive branching rule, where branching decisions are guided by maximal stable sets in the conflict graph, which identify the list of edges for branching. This stable set approach, which is explained in detail in Subsection 5.4.2, tries to search the solution space by only including new edges, resulting in subproblems obtained by deleting the edges in conflict with the branching edge.

For each subproblem $MSTC^{(t)}$ in the active node list $\mathcal{L}$, BB-S finds the branching edges $B = \{e_1, e_2, \dots, e_{d^{(t)}}\}$ using the maximal stable set heuristic Algorithm 4. Then, $d^{(t)}$ branches are created at search node t , to be added as subproblems. For subproblem i , the branching edge is included, $I \leftarrow I \cup \{e_i\}$ and the edges that are in conflict with the branching edge are excluded, together with the edges which have been included in the previous subproblems, $X \leftarrow X \cup (N_C(e_i) \cup \{e_1, e_2, \dots, e_{i-1}\})$. The reason for excluding the edges which have been previously included is to avoid the repetition of branches. The remaining steps are similar with BB-E, with the exception of finding the edge with the highest degree in the conflict subgraph induced by the edges of the spanning tree becoming unnecessary. BB-S algorithm is outlined in Figure 6.5.

Algorithm 10 Stable set branching based branch-and-bound (BB-S)

Input: A connected simple graph $G = (V(G), E(G))$ with edge weights c_e $e \in E(G)$, conflict graph $C = (V(C), E(C))$, probing threshold β ;

Output: A minimum conflict-free spanning tree H^* and its total weight z^* ;

```

1: begin
2: Set  $t = 0, i = 1, \bar{z} \leftarrow \infty, \mathcal{L} \leftarrow \{\emptyset\}, MSTC^{(0)} \leftarrow MSTC, I^{(0)} \leftarrow \{\emptyset\}, X^{(0)} \leftarrow \{\emptyset\}, F^{(0)} \leftarrow E(G),$ 
    $G^{(0)} \leftarrow G, C^{(0)} \leftarrow C$ ;
3: Order the edges according to their non-decreasing weights and save the order;
4: Order the edges in increasing order of conflict degrees and save the order;
5: Solve MST relaxation  $MST^{(0)}$ , which is the MST problem on  $G$ ;
6: Let  $T^{(0)} = (V(T^{(0)}), E(T^{(0)}))$  be its optimal solution and its value  $\underline{z}^{(0)}$ ;
7: if  $T^{(0)}$  is conflict free then
8:   Set  $H^* \leftarrow T^{(0)}$  and  $\bar{z} \leftarrow \underline{z}^{(0)}$ ;
9: else
10:   Set  $\mathcal{L} \leftarrow \{(MSTC^{(0)}, T^{(0)}, \underline{z}^{(0)})\}$ ;
11: end if
12: while  $\mathcal{L} \neq \emptyset$  do
13:   Delete subproblem  $MSTC^{(t)}$  with the smallest lower bound value  $\underline{z}^{(t)}$  and its relaxed optimal
     solution  $T^{(t)} = (V(T^{(t)}), E(T^{(t)}))$  from  $\mathcal{L}$ ;
14:   Call the maximal stable set heuristic (Algorithm 4) using  $C^{(t)}, I^{(t)}, X^{(t)}, F^{(t)}$  and get the list
     of branching edges  $B$ ;
15:   for  $e \in B$  do
16:     Set  $B \leftarrow B \setminus \{e\}$ 
17:     Generate subproblem  $MSTC^{(ti)}$  with  $I^{(ti)} \leftarrow I^{(t)} \cup \{e\}, X^{(ti)} \leftarrow X^{(t)} \cup N_{C^{(t)}}(e), F^{(ti)} \leftarrow$ 
        $F^{(t)} \setminus (N_{C^{(t)}}(e) \cup \{e\})$ ;
18:     Call Algorithm 5 to check the infeasibility of subproblem  $MSTC^{(ti)}$ ;
19:     if  $MSTC^{(ti)}$  is indecisive then
20:       Calculate simple lower bound  $LB^{(ti)}$ ;
21:       if  $LB^{(ti)} \leq \bar{z}$  then
22:         if  $|F^{(ti)}|/|E(G)| \leq 1 - \beta$  then
23:           Reduce the size of  $MSTC^{(ti)}$  by probing using Algorithm 6;
24:         end if
25:         Update  $T^{(t)}$  using Algorithms 11, 13, 16 to obtain  $T^{(ti)}$  and its value  $\underline{z}^{(ti)}$ ;
26:         if  $T^{(ti)}$  is a spanning tree of  $G^{(ti)}$  then
27:           if  $T^{(ti)}$  is conflict free then
28:             if  $\underline{z}^{(ti)} < \bar{z}$  then
29:               Set  $\bar{z} \leftarrow \underline{z}^{(ti)}$  and  $H^* \leftarrow T^{(ti)}$ ;
30:               Delete all problems with lower bound larger than or equal to  $\bar{z}$  from  $\mathcal{L}$ ;
31:             end if
32:           else
33:             Set  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(MSTC^{(ti)}, T^{(ti)}, \underline{z}^{(ti)})\}$ ;
34:           end if
35:         end if
36:       end if
37:     end if
38:     Set  $i \leftarrow i + 1$ 
39:   end for
40:   Set  $t \leftarrow t + 1$ ;
41: end while
42: An optimal solution is reached, return  $H^*$  and  $z^* = \bar{z}$ ;
43: end

```

Figure 6.5. Stable set branching based branch-and-bound algorithm (BB-S).

7. UPDATING SPANNING TREES

As discussed earlier, our implementation of BB does not solve a relaxed subproblem $MST^{(t)}$ from scratch, except the initial subproblem $MST^{(0)}$. Instead, the spanning tree is updated using dynamic tree operations: (i) deletion, (ii) insertion. After each deletion, a replacement edge needs to be found to restore the spanning tree for BB-E and BB-S. During each insertion, a cycle check becomes necessary only for Rec-D and BB-D. These dynamic tree operations make use of sophisticated ADS to be efficiently implemented. There is a number of works in the literature on dynamic tree operations, such as [53–56]. In addition, several studies are conducted in the literature to compare the performances of different dynamic tree implementations [57].

7.1. Deletion

Deletion operation is done for free edges of the graph $e \in F^{(t)}$, and they can be either within the optimal solution of subproblem $MSTC^{(t)}$ or not. During BB-S and BB-E, deletion is applied at subproblem $MSTC^{(t)}$ during tree restoration phase one-by-one for each edge $e \in F^{(t)}$ to be deleted from the corresponding spanning tree. Then, a replacement edge is searched, if it exists. The deletion algorithm is formally described in Figure 7.1.

Algorithm 11 Deletion of a free edge from a spanning tree of G

Input: Spanning tree $T = (V(T), E(T))$ and edge $e \in F^{(t)}$;

Output: Updated spanning tree;

1: **begin**

2: Set $X^{(t)} \leftarrow X^{(t)} \cup \{e\}$, $F^{(t)} \leftarrow F^{(t)} \setminus \{e\}$;

3: Update the induced conflict subgraph $C[F^{(t)}]$;

4: **if** $e \in E(T)$ **then**

5: Update the weight of the tree by setting $c(E(T)) \leftarrow c(E(T)) - c_e$;

6: Set $E(T) \leftarrow (E(T) \setminus \{e\})$;

7: **end if**

8: **return** updated tree T and its weight $c(E(T))$;

9: **end**

Figure 7.1. Deletion of an edge from the spanning tree.

As soon as the deletion operation is applied to the spanning tree, the replacement edge needs to be found. Several implementations to find the replacement edges are explained in the following section.

7.2. Finding Replacement Edges

Replacement edges can be found using different strategies with the aid of several ADS. We have considered four different algorithms to find the replacement edges for our final implementation. When an edge is removed, the tree becomes disconnected with two components forming a cut. The first three algorithms search for a replacement edge in the cut. The fourth algorithm looks for an elementary cycle which contains the removed edge together with a replacement edge.

- (i) Find replacement edge with breadth-first search
- (ii) Find replacement edge with predecessor array
- (iii) Find replacement edge with link-and-cut trees
- (iv) Find replacement edge with depth-first search

Our initial approach is to identify the vertices in each of the two components, after the removal of an edge. However, searching through the vertices that are in distinct components is not efficient, as not every vertex pair has an edge between them. Instead, we consider searching through the array of free edges which is sorted only once according to edge weights. Since the weight of a removed edge should be less than or equal to the weight of a replacement edge, the loop starts from this index. This would prevent unnecessary traversal of all free edges, and this strategy is used in all algorithms mentioned above. The four different find-cut algorithms are applied to the MSTC instance with its graph and conflict graph illustrated in Figure 5.5 and Figure 5.6, respectively.

In the first version, a breadth-first search (BFS) algorithm is used to traverse the graph starting from a source vertex, and visiting all vertices reachable from the

source vertex. A queue is used to maintain the order in which vertices are processed, and a boolean array is used to keep track of the visited vertices. Initially, one of the endpoints of a removed edge is marked as visited, and added to the queue. Then, a vertex from the front of the queue is removed. All of its unvisited neighbors are labeled as visited using the adjacency list of the tree. This process, which is outlined in Figure 7.2, is repeated until the queue is empty.

Algorithm 12 Find Replacement Edge with Breadth-First-Search

Input: A disconnected spanning tree $T = (V(T), E(T))$ and a deleted edge $f = \{p, q\}$;

Output: A replacement edge e if possible;

```

1: begin
2: Set  $S, Q = \emptyset, i, j, k \leftarrow 0$ ;
3: for  $i \in V(G)$  do
4:   Label all vertices as unvisited,  $S[i] \leftarrow 0$ ;
5: end for
6: Add one endpoint of edge  $f = \{p, q\}$  to the queue,  $Q \leftarrow Q \cup \{p\}$ ;
7: Label the endpoint  $p$  as visited,  $S[p] \leftarrow 1$ ;
8: while  $Q \neq \emptyset$  do
9:   Select vertex  $j$  from the front of the queue and remove it from there,  $Q \leftarrow Q \setminus \{j\}$ ;
10:  for  $k \in N_T(j)$  do
11:    if vertex  $k$  is not visited,  $S[k] = 0$  then
12:      Add vertex  $k$  to the queue,  $Q \leftarrow Q \cup \{k\}$ ;
13:      Label vertex  $k$  as visited,  $S[k] \leftarrow 1$ ;
14:    end if
15:  end for
16: end while
17: for  $e = \{u, v\} \notin E(T)$  do
18:  if  $S[u] \neq S[v]$  and  $c_e < c_f$  then
19:    Edge  $e$  is the replacement edge for  $f$ , set  $c_f = c_e$  and  $f \leftarrow e$ ;
20:  else
21:    Edge  $e$  creates a cycle within  $E(T)$ , it is not a replacement edge for  $f$ ;
22:  end if
23: end for
24: end

```

Figure 7.2. Finding a replacement edge: breadth-first-search.

In the second version, the predecessor information of all vertices is stored in an array. Using this array, a traversal from the head vertex to the root vertex is realized and the visited vertices are labelled with 1 if they are in the first component, and with 2 if they are in the second component. Two arrays are created representing the

vertices in components 1 and 2, and a temporary array. Then, searching through all vertices except the root vertex, the predecessor array is traversed. If a visited vertex is encountered, all vertices in the temporary array is added to the array of the relevant component. If a replacement edge is found, the predecessor array is updated accordingly. The algorithm is depicted in Figure 7.3.

Algorithm 13 Find Replacement Edge with Predecessor Array

Input: A disconnected spanning tree $T = (V(T), E(T))$ and a deleted edge $f = \{p, q\}$;

Output: A replacement edge e if possible;

```

1: begin
2: Find the root of  $T = (V(T), E(T))$  by moving up starting from  $p$  using predecessor
   relations and labelling traversed vertices including the root with  $l_1$ ;
3: Label  $q$  with  $l_2$ ;
4: for  $v \in V(T)$  do
5:   Move up to the root using predecessor relations starting from  $v$  until a labelled vertex
      $w$  is encountered;
6:   if label is  $l_1$  then
7:     All vertices of the path from  $v$  to  $w$  are in  $T_1 = (V(T_1), E(T_1))$ , label them with  $l_1$ ;
8:   else
9:     All vertices of the path from  $v$  to  $w$  are in  $T_2 = (V(T_2), E(T_2))$ , label them with  $l_2$ ;
10:  end if
11: end for
12: Set  $c_e = \infty$  and  $e \leftarrow \emptyset$ ;
13: for  $v \in V(T_1)$  do
14:   for  $g \in \delta_{T_1}(v)$  do
15:     if endpoints of  $g$  have different labels and  $c_g < c_e$  then
16:        $c_e = c_g$  and  $e \leftarrow g$ ;
17:     end if
18:   end for
19: end for
20: if  $e \neq \emptyset$  then
21:    $e = \{u, v\}$  is the replacement edge of  $f$  with  $u \in V(T_1)$  and  $v \in V(T_2)$ ;
22:   Make  $u$  the predecessor of  $v$ ;
23:   Reverse the predecessor relations on the path from  $q$  to  $v$ ;
24: else
25:   Edge  $e$  creates a cycle within  $E(T)$ , it is not a replacement edge for  $f$ ;
26: end if
27: return
28: end

```

Figure 7.3. Finding a replacement edge: predecessor array.

The third algorithm implements a link-and-cut tree (LCT), introduced in [58]. Dynamic trees are special ADS which contain connectivity information of rooted trees

as there occur dynamic changes. These trees support operations such as linking two trees together, cutting an edge to split a tree into two separate trees, and connectivity checks for two nodes. Dynamic trees have many different implementations such as LCTs and splay trees, top trees, and Euler-tour trees [54]. LCTs are dynamic trees that allow efficient path-related queries and updates. The operations mentioned above can be performed in logarithmic time complexity with the aid of LCTs. They use a combination of rotations and path-reversal techniques (also called as splay operations). The replacement edge algorithm using LCT is outlined in Figure 7.4.

Algorithm 14 Find Replacement Edge with Link-Cut-Tree

Input: A disconnected spanning tree $T = (V(T), E(T))$ and a deleted edge $f = \{p, q\}$;

Output: A replacement edge e if possible;

```

1: begin
2: for  $e = \{u, v\} \notin E(T)$  do
3:   if  $root(u) \neq root(v)$  and  $c_e < c_f$  then
4:     Edge  $e$  is the replacement edge for  $f$ , set  $c_f = c_e$  and  $f \leftarrow e$ ;
5:      $link(u, v)$ ;
6:   else
7:     Edge  $e$  creates a cycle within  $E(T)$ , it is not a replacement edge for  $f$ ;
8:   end if
9: end for
10: return
11: end

```

Figure 7.4. Finding a replacement edge: link-cut-trees.

The last algorithm finds the elementary cycle that contains the removed edge, and a free edge. It is proposed in [56] in order to find all replacement edges in an MST where a single edge is removed. They use a special case of the disjoint set ADS, proposed in [59]. DFS is applied to a tree which stores information about the parents of all vertices, IN and OUT numbers updated by a counter. IN is assigned the value of the counter first time a vertex is visited, and OUT is assigned the value of the counter when a vertex is visited for the last time. We had to slightly modify the algorithm as we do not need to obtain all replacement edges. Instead, it stops when a free edge with minimum weight is found. The algorithm is depicted in Figure 7.5.

Algorithm 15 Find Replacement Edge with Depth-First Search

Input: A disconnected spanning tree $T = (V(T), E(T))$, a deleted edge $f = \{p, q\}$;

Output: A replacement edge e if possible.

```

1: begin
2: Run DFS on  $T$  prior to deletion of  $f$  to obtain  $IN, OUT, P$ ;
3: for  $i \in V(G)$  do
4:   Initialize disjoint sets,  $make-set(i)$ ;
5: end for
6: Sort the non-tree edges in nondecreasing order of their weights;
7: for  $e = \{u, v\} \notin E(T)$  do
8:   if  $IN[u] < IN[v] < OUT[u]$  then
9:      $u$  is the ancestor of  $v$ ;
10:  else
11:    if  $IN[v] < IN[u] < OUT[v]$  then
12:       $v$  is the ancestor of  $u$ , set  $k_1 \leftarrow IN[v], k_2 \leftarrow IN[u]$ ;
13:    else
14:      if  $IN[u] < IN[v]$  then
15:         $u$  is left of  $v$ , set  $k_1 \leftarrow OUT[u], k_2 \leftarrow IN[v]$ ;
16:      else
17:         $u$  is right of  $v$ , set  $k_1 \leftarrow OUT[v], k_2 \leftarrow IN[u]$ ;
18:      end if
19:    end if
20:    Set  $s \leftarrow u$ ;
21:    while  $k_1 < k_2$  do
22:      if  $find-set(s) = s$  then
23:        if  $(s = p \text{ and } P[s] = q) \text{ or } (s = q \text{ and } P[s] = p)$  then
24:          Edge  $e$  is a replacement edge for  $f$ .
25:          return
26:        end if
27:         $union(s, P[s])$ ;
28:      end if
29:      Set  $s = find-set(s)$ ;
30:      if  $v$  is the ancestor of  $u$  then
31:        Set  $k_2 \leftarrow IN[s]$ ;
32:      else if  $u$  is left of  $v$  then
33:        Set  $k_1 \leftarrow OUT[s]$ ;
34:      else if  $u$  is right of  $v$  then
35:        Set  $k_2 \leftarrow IN[s]$ ;
36:      end if
37:    end while
38:  end if
39:  Switch the endpoints of  $e = \{u, v\}$  and go to 8;
40: end for
41: No replacement edge is found for  $f$ ;
42: return
43: end

```

Figure 7.5. Finding a replacement edge: depth-first search.

7.3. Insertion

The insertion operation is done in two different context. For the case of BB-S and BB-E, insertion operation is applied at subproblem $MSTC^{(t)}$ only for a replacement edge $e \in F^{(t)}$ when an edge $f \in E(T^{(t)})$ is deleted from the corresponding MST, if such a replacement edge exists. Here, a cycle check is not necessary since e is chosen as a replacement edge for f , where the check for a cycle is already done during the search procedure for e .

In the case of Rec-D and BB-D, a free edge $e \in F^{(t)}$ is examined to be inserted into the set $I^{(t)}$. In that case, a cycle check must be performed as e might create a cycle within $I^{(t)}$. The insertion algorithm is formally described in Figure 7.6:

Algorithm 16 Insertion of a new edge to a spanning tree of G

Input: Spanning tree $T = (V(T), E(T))$ and edge $e \in F^{(t)}$;

Output: Updated spanning tree;

```

1: begin
2: if  $e$  is to be inserted into set  $I^{(t)}$  then
3:   Check for a cycle in  $I^{(t)} \cup \{e\}$  using cycle detection algorithms;
4:   if  $I^{(t)} \cup \{e\}$  creates a cycle then
5:     return
6:   else
7:     Update the weight of included edges by setting  $c(I^{(t)}) \leftarrow c(I^{(t)}) + c_e$ ;
8:     Set  $I^{(t)} \leftarrow I^{(t)} \cup \{e\}$ ;
9:     Adjust the data structure used for cycle checks to represent updated  $I^{(t)}$ ;
10:    return updated set  $I^{(t)}$  and its weight  $c(I^{(t)})$ ;
11:   end if
12: else
13:   Update the weight of the tree by setting  $c(E(T)) \leftarrow c(E(T)) + c_e$ ;
14:   Set  $E(T) \leftarrow (E(T) \cup \{e\})$ ;
15:   return updated tree  $T$  and its weight  $c(E(T))$ ;
16: end if
17: end

```

Figure 7.6. Insertion of a new edge to a spanning tree .

As described above, the insertion algorithm is very straightforward when there is no need for a cycle check, it only updates the edges of the spanning tree and its weight. After that, the relevant ADS that is used for the implementation of cycle

checks is updated to represent the new tree. The algorithms for cycle checks are briefly explained in the following section.

7.4. Cycle Check

For checking cycles during diving, we tested four different algorithms, which are defined below:

- (i) Check-cycle with disjoint-sets
- (ii) Check-cycle with breadth-first search
- (iii) Check-cycle with link-and-cut trees
- (iv) Check-cycle with predecessor array

The first algorithm uses disjoint-set ADS to check for cycles. It contains the functions *make-set*, *find-set* and *union* [60]. The function *make-set*(x) creates a set containing a single item x . The function *find-set*(x) finds the set that contains x . The function *union*(x, y) merges the set that contains x , and the other set that contains y into a single set, resulting in $\text{find-set}(x) = \text{find-set}(y)$. To illustrate the disjoint set functions, a simple example with three items is given in Figure 7.7.

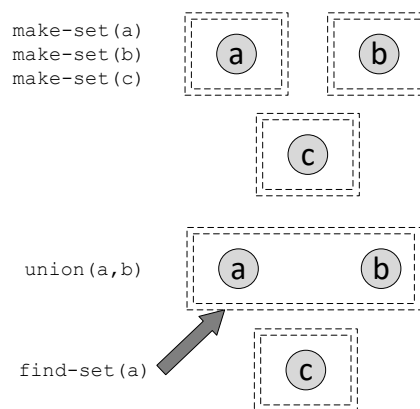


Figure 7.7. Disjoint set functions.

Disjoint set ADS can be implemented as linked lists or trees. In our application, the vertices of graph G are represented as disjoint sets where each set belongs to a tree. In the tree, each element has a pointer to its parent and the root points to itself. The tree does not need to be balanced or binary.

For faster convergence; union by rank, where the tree with smaller size is made a subtree of the other tree during *union* operation, and path compression, where the pointers of the parent nodes in the path from u to root r are made pointing to r directly during *find-set* operation, are implemented. Note that tree rank is an upper bound on tree height since path compression does not maintain height information. To visualize disjoint set trees, an example with two trees is given in Figure 7.8.

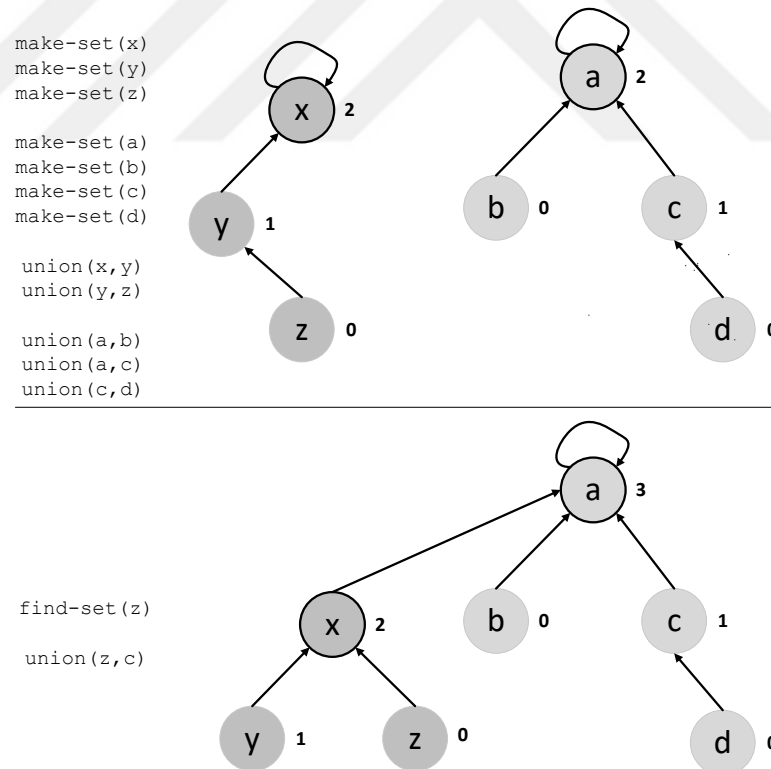


Figure 7.8. Union by rank and path compression in disjoint sets.

However, the traditional version of disjoint-sets with path compression and union-by-rank would not work, as the diving algorithm includes backtracking. As an alterna-

tive, persistent disjoint-sets that allow backtracking should be used, which are proposed in [61, 62]. The roots of the sets that the endpoints $\{u, v\}$ of the last included edge e is found, and if $\{u, v\}$ belong to the same set, then there exists a cycle.

If e does not result in a cycle in the forest induced by the set of included edges, the states of the disjoint sets which are updated during a union operation are stored in a stack, and a rollback function is defined which restores the states of these disjoint sets during backtracking. The description of check cycle with disjoint sets algorithm is illustrated in Figure 7.9.

Algorithm 17 Check Cycle with Disjoint Sets

Input: $G = (V(G), E(G))$, the partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$, $e \in F^{(t)}$;

Output: Detection of a cycle in $I^{(t)} \cup e$;

```

1: begin
2: Select edge  $e = \{u, v\}$ ;
3:  $u \leftarrow$  endpoint  $e$ ;
4:  $v \leftarrow$  endpoint  $e$ ;
5:  $pu \leftarrow \text{find-set}(u)$ ;
6:  $p_v \leftarrow \text{find-set}(v)$ ;
7: if  $pu \neq p_v$  then
8:   Edge  $e$  does not create a cycle when added to  $I^{(t)}$ ;
9:    $\text{union}(u, v)$ ;
10: else
11:   Edge  $e$  creates a cycle when added to  $I^{(t)}$ ;
12: end if
13: return
14: end

```

Figure 7.9. Cycle detection with disjoint sets.

In the second algorithm, BFS is used to traverse the forest induced by the set of included edges, starting from one endpoint u of the last included edge e . The process stops if the other endpoint v is encountered, resulting in a cycle. Otherwise, the algorithm traverses all vertices reachable from u .

A queue, a boolean array, and an adjacency list of the included edges is used in this algorithm, similar to the version used to find the replacement edges. For backtracking, the adjacency list of the included edges is updated by removing the corresponding

elements from the list. The formal procedure of check cycle with BFS algorithm is provided in Figure 7.10.

Algorithm 18 Check Cycle with Breadth-First-Search

Input: $G = (V(G), E(G))$, the partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$, $e \in F^{(t)}$;

Output: Detection of a cycle in $I^{(t)} \cup e$;

```

1: begin
2: Set  $S, Q = \emptyset, j \leftarrow 0$ ;
3: for  $i \in V(G)$  do
4:   Label all vertices as unvisited,  $S[i] \leftarrow 0$ ;
5: end for
6: Add one endpoint of edge  $e = \{u, v\}$  to the queue,  $Q \leftarrow Q \cup \{u\}$ ;
7: while  $Q \neq \emptyset$  do
8:   Select vertex  $j$  which is at the front of the queue,  $Q \leftarrow Q \setminus \{j\}$ ;
9:   for  $k \in N_{G[I^{(t)}}(j)$  do
10:    if Vertex  $k$  is not visited,  $S[k] = 0$  then
11:      Add vertex  $k$  to the queue,  $Q \leftarrow Q \cup \{k\}$ ;
12:      Label vertex  $k$  as visited,  $S[k] \leftarrow 1$ ;
13:    end if
14:  end for
15: end while
16: if  $S[u] \neq S[v]$  then
17:   Edge  $e$  does not create a cycle when added to  $I^{(t)}$ ;
18: else
19:   Edge  $e$  creates a cycle when added to  $I^{(t)}$ ;
20: end if
21: return
22: end

```

Figure 7.10. Cycle detection with breadth-first-search.

The third algorithm implements a LCT on the forest induced by the set of included edges. Cycle check on the included edges is done by comparing the roots of two endpoints $\{u, v\}$ of the last included edge e . If the roots are the same, then e causes a cycle in the set of included edges. The endpoints $\{u, v\}$ are linked if it does not form a cycle in the set of included edges. Cut operation of LCTs is used for backtracking.

Splay operations, which are self-adjusting operations used in splay trees, where a node is moved to the root of the tree through a series of rotations (zig, zig-zig, or zig-zag), are used to update the corresponding LCT. These operations ensure that tree

manipulations occur in amortized logarithmic time. The use of LCTs allows for fast updates and queries, making it a robust choice for large-scale graphs where cycles must be frequently detected. Check cycle with LCT algorithm is depicted in Figure 7.11.

Algorithm 19 Check Cycle with Link-and-Cut Trees

Input: $G = (V(G), E(G))$, the partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$, $e \in F^{(t)}$;

Output: Detection of a cycle in $I^{(t)} \cup e$;

```

1: begin
2:   Select edge  $e = \{u, v\}$ ;
3:    $u \leftarrow \text{endpoint } e$ ;
4:    $v \leftarrow \text{endpoint } e$ ;
5:    $pu \leftarrow \text{root}(u)$ ;
6:    $p_v \leftarrow \text{root}(v)$ ;
7:   if  $pu \neq p_v$  then
8:     Edge  $e$  does not create a cycle when added to  $I^{(t)}$ ;
9:      $\text{link}(u, v)$ ;
10:  else
11:    Edge  $e$  creates a cycle when added to  $I^{(t)}$ ;
12:  end if
13:  return
14: end

```

Figure 7.11. Cycle detection with link-and-cut trees.

In the fourth version, we store the predecessor information of all vertices in an array on the forest induced by the set of included edges. Using this array, we traverse from the endpoints $\{u, v\}$ of the last included edge e to their roots using predecessor information, and storing the corresponding paths. Traversing on the path with smaller size, if the other endpoint of e is encountered, then e forms a cycle in I' .

If no cycle is formed, the traversed path is stored in a history vector and then reversed, and the predecessor information is updated accordingly. For backtracking, the original path before the addition of edge e is obtained from the history vector, and the predecessor information is restored. Cycle check algorithm using predecessor array is outlined in Figure 7.12.

Algorithm 20 Check Cycle with Predecessor Array

Input: $G = (V(G), E(G))$, the partition $(I^{(t)}, X^{(t)}, F^{(t)})$ of $MSTC^{(t)}$, $e \in F^{(t)}$;

Output: Detection of a cycle in $I^{(t)} \cup e$;

```

1: begin
2: Select one endpoint of edge  $e = \{u, v\}$ ;
3: Move up to  $root(u)$  using the predecessor relations starting from  $u$  and label the traversed
   vertices;
4: Move up to  $root(v)$  using the predecessor relations starting from  $v$ ;
5: if A labelled vertex is encountered during traversal then
6:   Edge  $e$  creates a cycle when added to  $I^{(t)}$ ;
7: else
8:   Edge  $e$  does not create a cycle when added to  $I^{(t)}$ ;
9:   if The path from  $u$  to  $root(u)$  is shorter than the path from  $v$  to  $root(v)$  then
10:    Reverse the predecessor relations on the path from  $u$  to  $root(u)$ ;
11:    Make  $v$  the predecessor of  $u$ ;
12:   else
13:    Reverse the predecessor relations on the path from  $v$  to  $root(v)$ ;
14:    Make  $u$  the predecessor of  $v$ ;
15:   end if
16: end if
17: return
18: end

```

Figure 7.12. Cycle detection with predecessor array.

7.5. Illustration for Spanning Tree Updates

Here, the four replacement edge algorithms are illustrated to provide insights to the reader about their dynamics. The sample graph previously depicted with Figure 5.5 is used for illustration. Initially, the BFS algorithm to find replacement edges is illustrated in Figure 7.13.

BFS begins at both endpoints of the removed edge ($e_4 = \{v_1, v_2\}$). Initially, v_2 is labelled, and the vertices that are in the neighborhood of v_2 on tree T are also labelled, $N_T(v_2) = \{v_0\}$. Then, the vertices that are in the neighborhood of v_0 are labelled, $N_T(v_0) = \{v_3\}$. There are no unlabelled vertices in $N_T(v_3) = \{v_0\}$, so the first connected component of T becomes $S = \{v_0, v_2, v_3\}$. On the other side, v_1 is labelled, and the vertices that are in the neighborhood of v_1 are also labelled,

$N_T(v_1) = \{v_4, v_5\}$. Then, the vertices that are in the neighborhood of v_4 are labelled, $N_T(v_4) = \{v_6\}$. After that, there are no unlabelled vertices in $N_T(v_6) = \{v_4\}$, and $N_T(v_5) = \{v_1\}$, so the second connected component of T becomes $\bar{S} = \{v_1, v_4, v_5, v_6\}$.

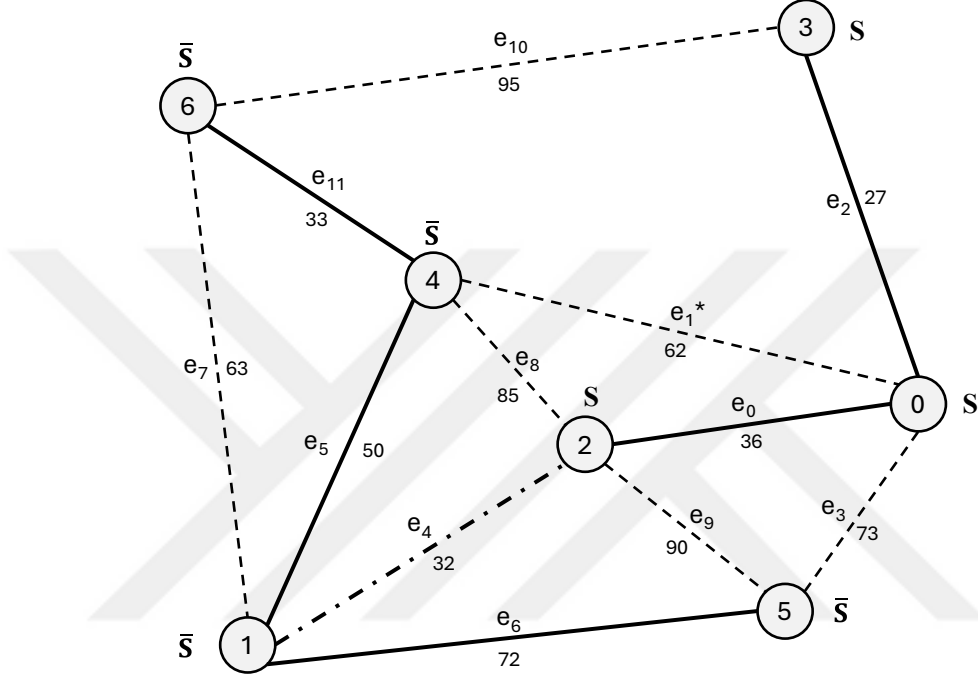


Figure 7.13. Edge replacement with breadth-first search on the sample graph.

After the two components of the tree are found, the endpoints of the free edges (which were sorted in increasing order according to their weights) are checked to find out whether they belong to the same component or not. The cut-set of the $(S, \bar{S})$ cut is given as $\delta_G(S) = \{e_1, e_3, e_8, e_9, e_{10}\}$, so all free edges are candidates to be replacement edges for e_4 . The free edge with the lowest weight is e_1 , hence the BFS procedure finishes.

The predecessor array to find replacement edges is illustrated in Figure 7.14. In this algorithm, the initial step is to label the root vertex of the tree (v_0) with 1. After the removal of e_4 , we check which endpoint is the predecessor of the other one. v_2 is the predecessor of v_1 , so the predecessor of v_1 is set as -1 (i.e., it becomes the root of the second component) and v_1 is labelled with 2. Then, the path from v_2 to the root

vertex is traversed using predecessor information, $\Pi_2 = v_2, v_0$, and the vertices in this path are labelled with 1.

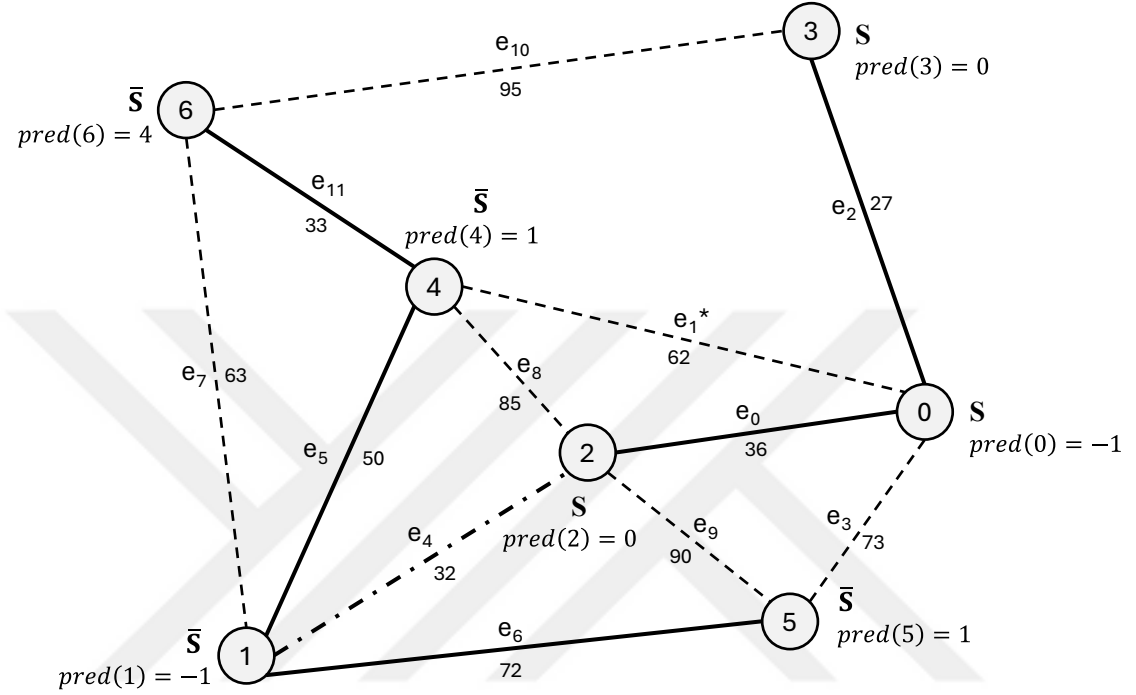


Figure 7.14. Edge replacement with predecessor array on the sample graph.

The paths from all other vertices to their corresponding root vertex are determined in similar fashion, but in order to prevent unnecessary iterations, the labelling process stops when a labelled vertex is encountered. The path from v_3 to its root vertex is $\Pi_2 = v_3, v_0$, so v_3 is labelled with 1. The path from v_4 to its root vertex is $\Pi_2 = v_4, v_1$, so v_4 is labelled with 2. The path from v_5 to its root vertex is $\Pi_2 = v_5, v_1$, so v_5 is labelled with 2. The path from v_6 to its root vertex is $\Pi_3 = v_6, v_4, v_1$, but v_4 has already been labelled with 2. As a result, v_6 is also labelled with 2 as soon as v_4 is encountered. After labelling is complete, the procedure is the same as the first algorithm (BFS).

The LCT algorithm to find replacement edges is illustrated in Figure 7.15. Here, the two connected components of the disconnected tree is stored as two LCTs with

different roots after the cut operation on $e_4 = \{v_1, v_2\}$: v_1 becomes the root of the first LCT, and v_2 becomes the root of the second tree. Again, by traversing the list of sorted free edges, it is investigated whether the roots of the two vertices that the free edge is in between are the same or not. The endpoints of e_1 are in two different LCTs: the LCT that v_4 belongs to has its root as v_1 , and the LCT that v_0 belongs to has its root as v_2 . Hence, e_1 is a replacement edge for e_4 , and no cycles are formed when it is added to the tree.

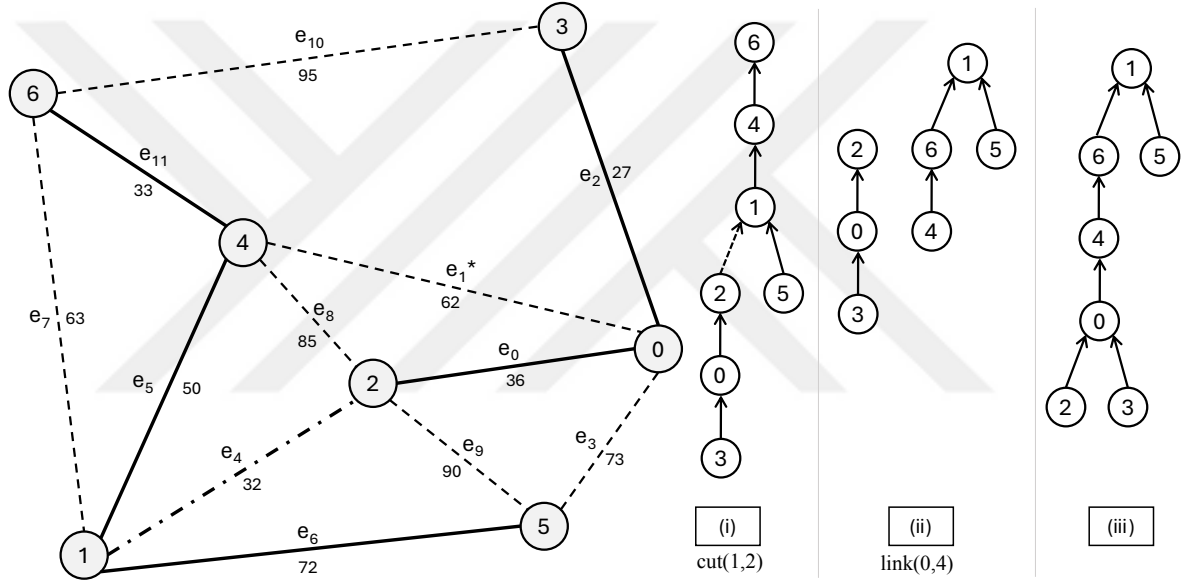


Figure 7.15. Edge replacement with link-cut tree on the sample graph.

The DFS algorithm to find replacement edges is depicted in Figure 7.16. In this algorithm, the vertices are labelled with IN, OUT numbers, and their parents are determined using DFS starting from the root vertex (v_0). It should be noted that the labelling considers the connected tree, i.e., before the removal of e_4 . Then using IN, OUT, and parent information, a cycle is identified which includes a free edge from the list of sorted free edges. Here, $e_1 = \{v_0, v_4\}$ is the free edge with the lowest weight ($c_4 = 62$). Starting from v_4 , its parent is v_1 , the parent of v_1 is v_2 , and the parent of v_2 is v_0 which is the other endpoint of e_1 . The cycle is identified as $\sigma_4 = v_4, v_1, v_2, v_0, v_4$. As e_4 (the removed edge) is encountered on σ_4 , e_1 becomes its replacement edge.

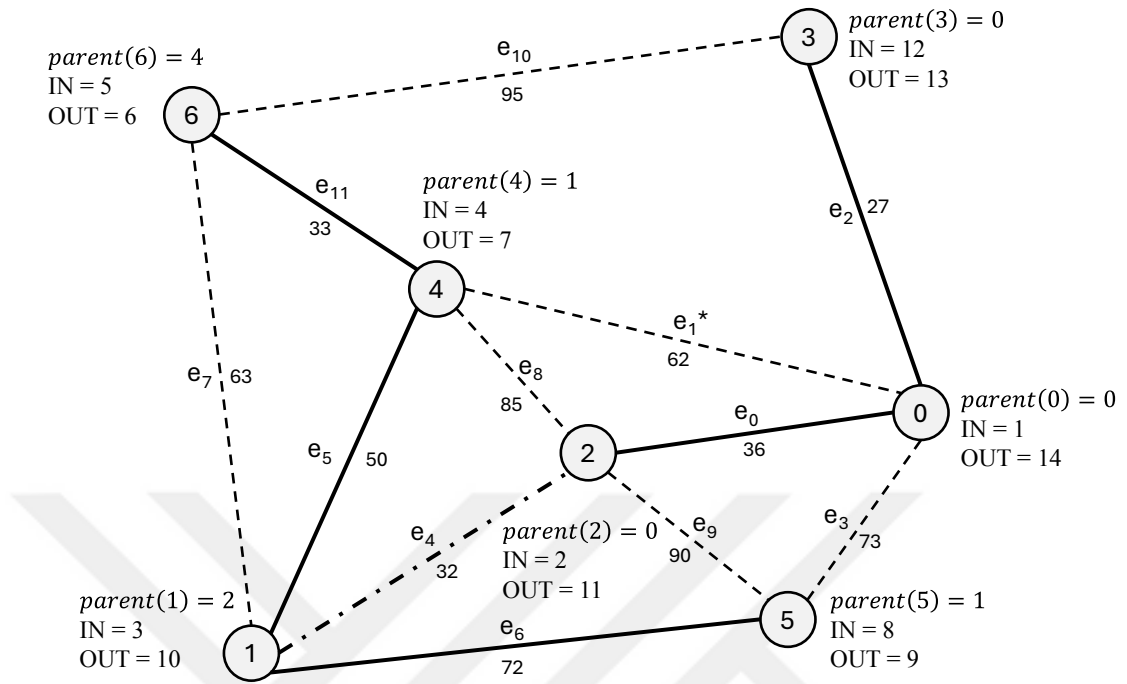


Figure 7.16. Edge replacement with depth-first search on the sample graph.

8. COMPUTATIONAL EXPERIMENTS

This chapter aims to summarize our findings from the results of computational experiments. Initially, test problem generator is introduced, and the reference formulation for benchmarking is determined. Then, the results of computational tests realized on existing [16,18] and newly generated test instances are summarized. Detailed results can be found in the Appendix.

8.1. Implementation Details

The implementation of the BB algorithms needs advanced data structures and fine-tuned parameters. They are outlined in order to provide insights on the reason for their use.

8.1.1. Data Structures

The graph is represented as an adjacency list to efficiently store vertex relationships. Each vertex maintains a list of its connected vertices, represented as pairs containing the neighboring vertex and the edge connecting these two vertices. This representation is optimized for operations like graph traversal and MST computations. An adjacency matrix is also used for fast lookups. An incidence list is used to track edges connected to each edge. Conflicts between edges are managed through a conflict graph, which is also represented as both an incidence list and an incidence matrix.

Initially, an MST is constructed using Prim's or Kruskal's algorithm, and stored as a predecessor array. To handle large graphs, memory pools are utilized for efficient allocation and deallocation of BB search node structures. This approach reduces overhead and ensures scalability. Each node in the BB search tree is represented as a structure, which contains the included and excluded edges in the current solution, its weight, and the number of violated conflicts.

The active node list $\mathcal{L}$ is implemented as a priority queue (min-heap), where BB search nodes are sorted by their estimated lower bounds (total weight of the relaxed solution). This structure allows efficient extraction of the next node to process during BB exploration. Priority queue is an ADS where its elements $v \in S$ are partially ordered according to some priorities. Each element $v \in S$ has an associated value $key(v)$ which denote the priority of v . In a min-priority queue, the element with a lower key indicates a higher priority. Min-priority queue supports primitive operations of addition, deletion and selection of the element with the smallest key. In the implementation of Prim's algorithm with priority queue, we store all vertices that are not in S in a min-priority queue Q . Priorities (keys) must be assigned to each vertex of G . For each vertex $v \in V(G)$, its key is denoted by $d(v)$, which is the minimum weight of any edge connecting v to a vertex in tree S . When there is no such edge, then we set $d(v) = \infty$. Also, for each vertex $v \in V(G)$, $parent(v)$ is the parent of v in the tree.

A heap can be used to implement a priority queue, which can be represented as a binary tree with a partial order such that the parent is before children and no prioritization exists between the children [60]. This is also known as heap property. A heap can be represented as a linear array A , where root is $A[1]$. To assign the tree vertices to array elements, given index i for some node, we set $parent(i) = \lfloor i/2 \rfloor$, $left(i) = 2i$, and $right(i) = 2i + 1$. In a min-heap, the value of a node is at least the value of its parent, and the smallest value is stored at the root, i.e., $A[parent(i)] \leq A[i]$.

Min-heap supports the functions *heapify*, *build-heap*, *min*, *insert*, *delete-min* and *decrease-key*. The function *heapify*(A, i) restores the partial order (i.e., heap property) in an array A at vertex i that may violate the partial order. The function *build-heap*(A) produces a heap from an unordered array A into the same array. The function *min*(A) returns the element of array A with the highest priority. To demonstrate these min-heap functions on a sample input array A , a min-heap binary tree is built, illustrated with Figure 8.1.

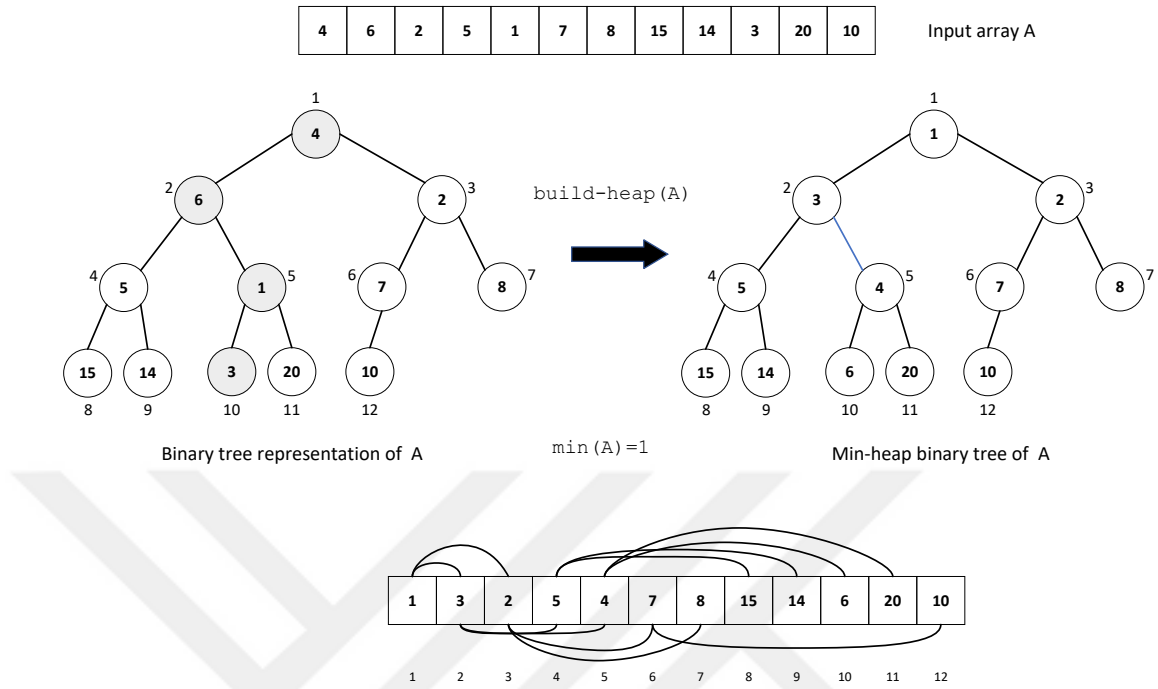


Figure 8.1. Building a min-heap binary tree.

The function $insert(S, u)$ inserts element u into set S and restores the heap property. The function $delete-min(S, u)$ deletes the element of S with the highest property, sets the pointer to u , and makes the memory restoration. Lastly, given a set S , a pointer to an element $u \in S$, and a weight $w \leq w_u$, the function $decrease-key(S, u, w)$ updates the weight of u by replacing it with w and makes the necessary changes in the set.

To store the tree and the set of included edges (which is a forest), a predecessor array is used in all our BB algorithms. The tree is stored as a predecessor array rooted at vertex 0, and the predecessor of the root vertex is set as -1, as explained and discussed in detail in Chapter 7. For the forest of included edges, the predecessor array is modified where the root of each component points to itself. An illustration on a graph with seven nodes and three components is illustrated with Figure 8.2. The first component consists of vertices v_0, v_1, v_3 , and its root is v_3 . The second component consists of vertices v_2, v_5, v_7 and is rooted at v_5 . The third component has two vertices, v_4, v_6 and it is rooted at v_4 .

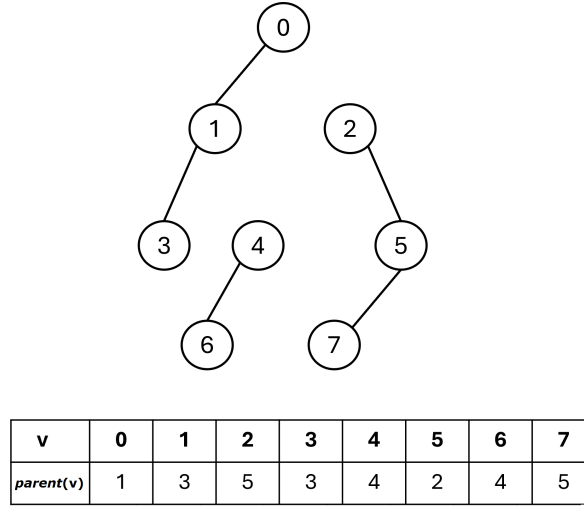


Figure 8.2. Predecessor array on the forest of included edges.

8.1.2. Parameters

The effectiveness of the algorithm heavily depends on carefully chosen parameters that guide edge selection, conflict resolution, and the efficiency of the BB process. The probing threshold β is expressed as a percentage of edges included and excluded in the current solution, and triggers probing depending on the depth of the BB search tree. When an edge is forced to be excluded from a conflict-free MST during probing, it must be checked whether this edge forms a cycle within the set of included edges or not before adding the edge. This might considerably increase the CPU time especially for dense graphs. In our computational experiments, it is observed that it does improve the solution significantly for some cases. Hence, we decided to keep it in our implementation of BB-E and BB-S, but revise the order of probing operations during the execution of BB-D and Rec-D due to extensive calls to probing functions within the code.

Free edges are sorted in nondecreasing order of their weights only in the beginning. They are used during tree restoration in order to speed up the process of finding replacement edges. Instead of checking whether there is an edge between two vertices and they belong to different components, this sorted array is traversed beginning from the edge with a weight larger than or equal to the weight of the deleted edge.

During BB-S, vertices of the conflict graph is sorted in increasing order of their degrees only once at the beginning, representing the number of conflicts that an edge is in conflict with other edges of the graph. This way, a maximal stable set is found at very low computational cost, compared to the sorting process applied at each iteration.

A time limit of 1800 seconds is determined for solution with a maximum number of 300 million BB nodes. We use Gurobi with default parameters while solving the test instances with STRONG and WEAK formulations in both single thread and multi thread settings.

8.2. Test Environment

All tests are done on a server running Microsoft Windows Server 2016 with Intel Xeon E5-2650 @ 2.20 GHz CPU and 288 GB RAM. C++ programming language (version 20) is used for the implementation [63].

MST relaxation is used as the initial relaxation strategy. Single-commodity flow based formulations are used to obtain optimal solutions coded in Python language (version 3.9.7) [64] and solved by Gurobi solver (version 10.0.3) [65]. The test instances are solved by Gurobi solver called from a driver program coded in Python with both STRONG and WEAK formulations in both multiple and single threaded settings.

8.2.1. Test Instances

There are two sets available and used during the previous studies. Since they are not fully capable for performance testing, random instances are also generated.

8.2.1.1. Existing Instances. A set of test instances have already been generated by Zhang et al. [16] (referenced as “Zhang’s dataset”), which is extended with additional test problems generated by Carrabs et al. [18] (referenced as “Carrabs’ dataset”). Both datasets are accessible online in [66].

Zhang’s dataset contains two types of instances, which are grouped as Type 1 and Type 2 instances [16]. Fourteen Type 1 instances are known to be infeasible, and the feasibility of the remaining nine Type 1 instances are unknown. The average graph density of Type 1 instances is $\bar{\Gamma} = 0.07$ and their average conflict density is $\bar{\Omega} = 0.03$. Both the graphs G and the conflict graphs C of Type 1 instances are relatively sparse. There are twenty-seven Type 2 instances, which are proven to be feasible. Their average graph and conflict density are $\bar{\Gamma} = 0.05$ and $\bar{\Omega} = 0.23$. The graphs G are again sparse in Type 2 instances, and the conflict graphs C are denser compared to Type 1 instances. As explained in [17] and [18], Type 2 instances in Zhang dataset are created in the form of disjoint components connected by means of bridges to ensure feasibility, which makes them easily solvable during preprocessing.

Carrabs’ dataset contains 180 test instances where $|V(G)| \in \{25, 50, 75, 100\}$; and $|E(G)|$ and $|E(C)|$ determined according to $\Gamma \in \{0.2, 0.3, 0.4\}$ and $\Omega \in \{0.01, 0.04, 0.07\}$. Their average graph and conflict density are $\bar{\Gamma} = 0.3$ and $\bar{\Omega} = 0.04$. Five test instances are created for each configuration. These instances consist of sparse graphs and extremely sparse conflict graphs. According to our preliminary observations they are harder to solve compared to instances with dense conflict graphs. We have not been able to obtain the codes that are used in the original study. As a result, we cannot replicate the results in our setting. In addition, the authors did not specify their hardware explicitly in their study, hence we do not have any option to use their results for benchmarking as it would not be a fair comparison.

8.2.1.2. Random Problems. In addition to these datasets, new test instances have been generated in C++ programming environment for benchmarking and performance analysis of the novel BB algorithm. Random graphs are created with edge weights between 20 and 80, an MST is found using Prim’s algorithm with given edge weights of 100. Additionally, a random conflict graph is created where conflicts are prevented between the edges of the determined MST. This way, at least one feasible solution for the random MSTC instance is guaranteed. The random MSTC instance generation algorithm is outlined in Figure 8.3.

Algorithm 21 MSTC Test Problem Instance Generator

Input: Number of vertices m , number of edges n , number of conflicts p , instance index s , lower limit l and upper limit u of edge weights, a maximum weight c_{max} for MST edges;

Output: A random graph $G = (V(G), E(G))$; and its conflict graph $C = (V(C), E(C))$;

```

1: begin
2: Set  $V(G), E(G), E(C) \leftarrow \emptyset$ ,  $seed \leftarrow s$ ,  $x, y \leftarrow 0$ ;
3: while  $|E(G)| \leq n$  do
4:   Create an edge  $e$ ;
5:   Set  $x, y \leftarrow \text{Uniform}(0, m-1)$ ,  $x \neq y$  and assign the endpoints of  $e$  with  $x$  and  $y$ ;
6:   if  $x$  and  $y$  are not adjacent and there is no other edge  $f$  between  $x, y$  then
7:     Set the weight of the edge,  $c_e \leftarrow \text{Uniform}(l, u)$ ;
8:   end if
9:   Add the edge  $e$  to graph  $G$ ;
10: end while
11: Find the minimum weight spanning tree  $T^* = (V(T^*), E(T^*))$  by Prim-Heap algorithm;
12: for  $e \in E(T^*)$  do
13:   Set  $c_e = c_{max}$ ;
14: end for
15: while  $|E(C)| < p$  do
16:   Set  $k, l \leftarrow \text{Uniform}(0, n-1)$ ,  $k \neq l$ ;
17:   if  $e_k, e_l \notin E(T^*)$  and there are no conflicts between  $e_k$  and  $e_l$  then
18:     Add the conflict between  $e_k$  and  $e_l$  to the conflict graph  $C$ ;
19:   end if
20: end while
21: Write the graph and the conflict graph to a file;
22: end

```

Figure 8.3. Test instance generator.

Three test sets are generated to be used for benchmarking and performance profiling. “Restricted Test Set” contains 144 new test instances where $|V(G)| \in \{20, 30, 40, 50\}$; and $|E(G)|$ and $|E(C)|$ are determined according to different graph density levels of $\Gamma \in \{0.4, 0.5, 0.7\}$, and conflict density levels of $\Omega \in \{0.1, 0.3, 0.5, 0.7\}$. This set is designed to evaluate the performance of the STRONG and WEAK formulations in single and multiple thread settings. There are a total of 144 instances, three test instances are created for each configuration. Graph and conflict densities cover a larger range from smaller to larger values than Zhang’s and Carrabs’ instances.

Furthermore, another set named “Restricted Test Set Plus” is developed which extends the Restricted Test Set of 144 instances with 72 new instances where $|V(G)| \in$

$\{75, 100\}$. All new instances have $\Gamma \in \{0.4, 0.5, 0.7\}$, and $\Omega \in \{0.1, 0.3, 0.5, 0.7\}$ with $|E(G)|$ and $|E(C)|$ being determined accordingly, resulting in a total number of 216 instances. Using this set, different implementations of finding replacement edges are compared in terms of their performance. The effect of infeasibility test, probing, and diving on the implementation with best data structure is also analyzed with this set.

Additionally, “Large Test Set” is created where $|V(G)|$ ranges from 500 to 1200 with an increment of 100; and $|E(G)|$ and $|E(C)|$ are determined according to graph and conflict densities $\Gamma = 0.03, \Omega = 0.7$, resulting in 8 instances. As these instances are harder to generate in terms of space requirement and generation time, they include very sparse graphs and dense conflict graphs. This set is used to compare different implementations of finding replacement edges in a more challenging setting, and to reach asymptotic complexities of the ADS in the hope of observing their contributions to the efficiency.

Lastly, “Restricted Test Set Plus” is extended to different graph density levels between 0.2 and 0.8 and conflict density levels between 0.1 and 0.8 with an increment of 0.1, while keeping $|V(G)|$ unchanged. The resulting test set is named as “Extended Random Set” which contains 335 instances, and it is used to demonstrate the effect of graph and conflict densities in the solution of MSTC instances. Then, benchmarking results of the best MSTC formulation in Gurobi and the best version of our BB algorithm is done regarding Zhang’s and Carrabs’ datasets, and our Extended Random Set.

8.2.2. Performance Measures

The performance metrics include the number of optimal solutions “Opt. Sols”, feasible solutions “Feas. Sols.”, infeasible solutions “Infeas. Sols.” (used only for Zhang’s Type 1 dataset since the rest of the instances are feasible), average number of explored nodes “Explored Nodes” and active nodes “Active Nodes”, average best upper bound “Best UB”, and average CPU time in seconds for all instances “CPU

Time” (average CPU time in seconds of the instances commonly solved to optimality are given in parenthesis). The number of optimal solutions are reported where the instances are solved optimally within the time limit of 1800 seconds. Besides, the number of feasible solutions are the ones where a feasible solution is found within 1800 seconds, but the optimal solution cannot be found. The average number of explored nodes indicate the average number of explored nodes in the solution space for a specific test set. The average number of active nodes are the average number of nodes present as the algorithm terminates. The average best upper bound is the average number of upper bound of test instances. Average CPU time is collected for all instances within a test set, and the ones which are solved commonly to optimality within 1800 seconds in all versions. The first type of average CPU times are given within parenthesis. These metrics are provided as tables in the following sections.

Performance profiles are very crucial for benchmarking and evaluating the performance of optimization algorithms [67]. They use a performance ratio where the solution time of an algorithm is divided over the best solution time of all algorithms as a performance measure. The cumulative distribution function for the performance ratio is obtained. They are more robust and provide visual information compared to summary tables. The performance profile can be plotted according to a factor $\tau \in \mathbb{R}$ of the best possible ratio. For $\tau = 0$, corresponding ordinate value denotes the percentage of the probability that the algorithm is the winner considering overall instances. However, the interval for τ can be quite large, and the complete performance data may not be captured. As an alternative, the performance ratio is scaled to logarithm of base two, and plotted against τ . This way, the plot is expected to display the performance of all algorithms. From a visual point of view, the profile closest to the left hand corner is the best.

8.3. The Determination of the Reference Formulations

Benchmark formulation is determined in this section, as our aim is to generate reference values by solving them using a widely used commercial MILP solver. Both

Zhang’s, Carrabs’ datasets, and our Restricted Test Set are used for benchmarking. The comparison of MSTC formulations for Type 1 instances are reported with Table 8.1. In the multiple thread setting, eight of the nine Type 1 instances with unknown feasibility are solved optimally and a feasible solution for the remaining instance is found with both formulations. Both STRONG-multi and WEAK-multi identify seven instances as infeasible.

Table 8.1. Comparison of MSTC formulations on Zhang’s Type 1 dataset.

Test Statistic	Single Thread		Multiple Thread	
	STRONG	WEAK	STRONG	WEAK
Opt. Sols.	7	7	8	8
Feas. Sols.	4	4	3	3
Infeas. Sols.	5	5	7	7
Explored Nodes	17107	26806	94261	179306
Active Nodes	6026	13206	34807	101305
Best UB	6078.81	6070.36	6073.91	6106.45
CPU (secs.)	55.46 (964.50)	77.45 (946.96)	59.33 (789.34)	146.82 (742.66)

Single thread setting performs worse for both formulations in terms of all performance metrics, and it finds only seven optimal solutions. The winner is STRONG as it is faster, however WEAK provides tighter upper bounds and explores more nodes in the search space. The comparison of MSTC formulations for Type 2 instances are given with in Table 8.2.

Table 8.2. Comparison of MSTC formulations on Zhang’s Type 2 dataset.

Test Statistic	Single Thread		Multiple Thread	
	STRONG	WEAK	STRONG	WEAK
Opt. Sols.	27	27	27	27
Feas. Sols.	0	0	0	0
Explored Nodes	6.19	3.44	6.07	4.11
Active Nodes	0.59	0.59	0.74	0.74
Best UB	29028.92	29028.92	29028.85	29028.85
CPU (secs.)	1.07 (1.07)	0.50 (0.50)	1.21 (1.21)	0.60 (0.60)

In both single and multiple thread settings, both formulations solve all 27 instances optimally in a very short time with WEAK slightly performing better compared to STRONG. This is the reason why both average CPU times within and without parenthesis are the same. Since these Type 2 instances are not reliable as standalone, the performance profile of the complete Zhang dataset is depicted with Figure 8.4.

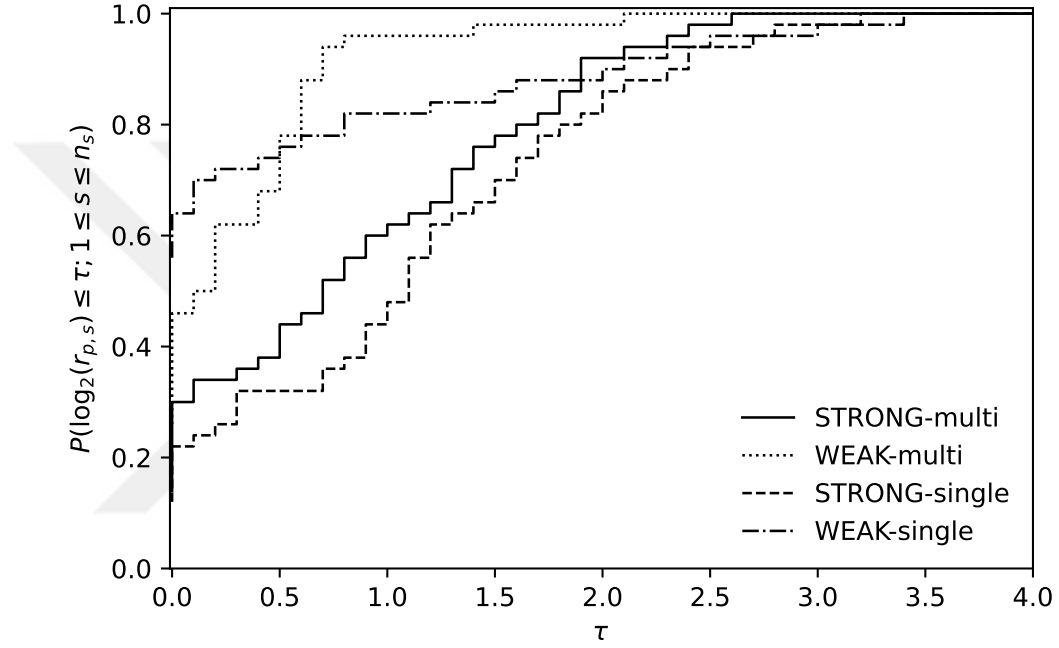


Figure 8.4. Performance profiles: MSTC formulations on Zhang's dataset.

At $\tau = 0$, WEAK-single performs the best, followed by WEAK-multi, and the STRONG formulations perform worse. When $\tau \geq 0.5$, WEAK-multi outperforms WEAK-single. As $\tau = 2$, nearly all instances are solved by WEAK-multi formulation and its performance ratio goes up to 0.98. The performance profiles indicate that the benchmark formulation for the complete Zhang's dataset should be chosen as WEAK-multi, which is followed by WEAK-single.

The comparison of MSTC formulations for Carrabs' dataset are included with Table 8.3. The multiple thread setting performs better in terms of the number of optimal solutions, number of explored nodes, and CPU time. There are 111 common optimal solutions in the multiple thread setting, and 108 common optimal solutions in

the single thread setting. These numbers are used for the average CPU times without parenthesis.

Table 8.3. Comparison of MSTC formulations on Carrabs' dataset.

Test Statistic	Single Thread		Multiple Thread	
	STRONG	WEAK	STRONG	WEAK
Opt. Sols.	108	108	117	111
Feas. Sols.	42	41	33	39
Explored Nodes	20390	50788	70905	281860
Active Nodes	7907	35389	28550	206147
Best UB	714.77	710.99	710.17	715.99
CPU (secs.)	19.97 (732.00)	32.11 (739.28)	30.13 (683.55)	42.47 (716.36)

The best performing formulation is the STRONG-multi as it has the highest number of optimally solved instances and the lowest CPU time in seconds. Notice that WEAK explore more average number of nodes compared to STRONG in both settings.

The performance profile of Carrabs' dataset is presented with Figure 8.5.

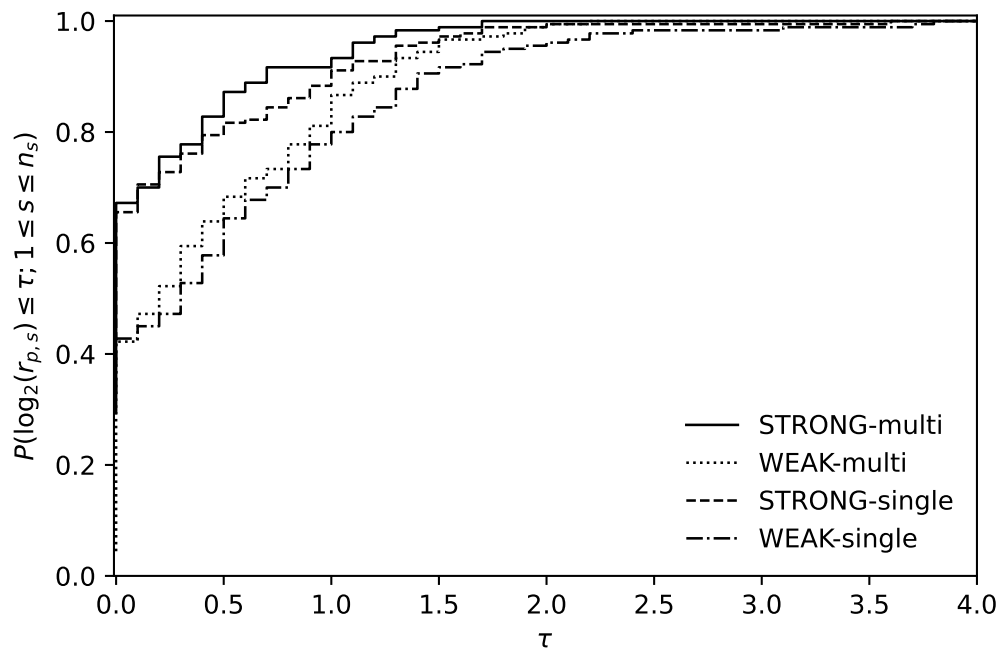


Figure 8.5. Performance profiles: MSTC formulations on Carrabs' dataset.

Here, both STRONG-multi and STRONG-single perform better than WEAK formulations at $\tau = 0$. As τ goes to 1, STRONG-multi performs the best, closely followed by STRONG-single, and WEAK-multi also begins to close the gap. For $\tau \geq 2$, all formulations except WEAK-single reach their highest performance ratio, which is very close to 1. STRONG-multi is the winner, being followed closely by STRONG-single. WEAK formulations do not perform well on Carrabs’ dataset.

We continue our analysis with the results computed on Restricted Test Set. The comparison of MSTC formulations for Restricted Test Set are given in Table 8.4.

Table 8.4. Comparison of MSTC formulations on random dataset.

Test Statistic	Single Thread		Multiple Thread	
	STRONG	WEAK	STRONG	WEAK
Opt. Sols.	111	111	119	119
Feas. Sols.	15	17	9	17
Explored Nodes	14422	59771	56926	216436
Active Nodes	6903	43024	29587	142620
Best UB	2633.37	2629.09	2687.44	2741.36
CPU (secs.)	86.64 (483.39)	111.38 (503.55)	54.05 (374.97)	74.53 (405.52)

There are 116 instances commonly solved optimally using multiple threads, and 109 instances commonly solved optimally using a single thread for both STRONG and WEAK. In both settings, WEAK finds a higher number of feasible solutions compared to STRONG, but the average CPU time in seconds is also larger. Furthermore, the average number of explored nodes and active nodes are higher for WEAK, which may be caused from the LP relaxation being weaker. Another implication of this fact is the average best upper bounds being lower than STRONG.

The performance profile of Restricted Test Set is described with Figure 8.6. It clearly indicates that the best performing formulation is STRONG-multi. It is followed by the WEAK-multi, but there is a significant gap between them.

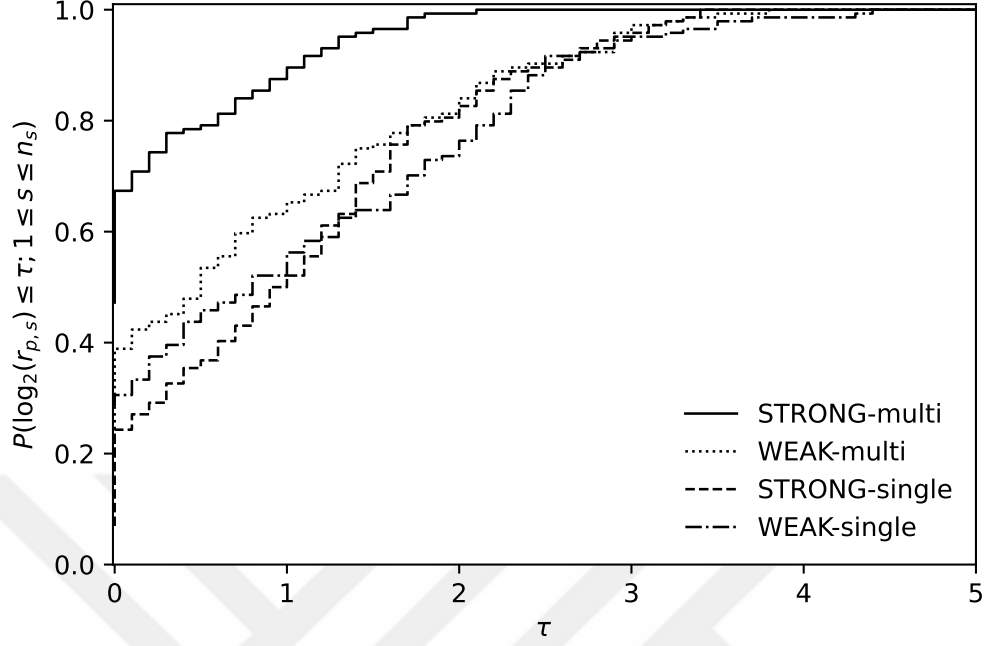


Figure 8.6. Performance profiles: MSTC formulations on random dataset.

Inspecting the results, WEAK-multi for Zhang’s dataset, and STRONG-multi for Carrabs’ and Restricted Test datasets are chosen as the benchmark reference formulations.

8.4. The Effect of Infeasibility Test and Probing

In this section, the effect of infeasibility test and probing is assessed with respect to particular test statistics and performance profiles. Performance profiles are plotted to visualize the performance of each algorithm in different tests.

Initially, we compare the plain versions of BB-E, BB-S, BB-D, and Rec-D on Restricted Test Set Plus. There are 152 optimal solutions found by BB-E, 135 optimal solutions found by Rec-D and BB-D, and 132 optimal solutions found by BB-S. A single feasible solution is found by BB-E and BB-D. There are 3 feasible solutions found by Rec-D, and 30 feasible solutions found by BB-S. The comparison of plain BB algorithms is given with Table 8.5.

Table 8.5. Comparison of plain BB algorithms.

Test Statistic	Plain BB Algorithms			
	BB-E	BB-S	Rec-D	BB-D
Opt. Sols.	152	132	135	135
Feas. Sols.	1	30	3	1
Explored Nodes	31900517	15947050	301904792	206467639
Active Nodes	13909241	15086190	15652021	5922888
Best UB	3965.76	4115.01	3720.76	3751.23
CPU (secs.)	6.11 (594.05)	35.91 (745.41)	86.02 (726.63)	97.11 (733.89)

BB-S also has the highest best UB among the others. In terms of average CPU time of all instances, BB-S performs the worst, followed by Rec-D and BB-D, and the best algorithm is BB-E. When we compare the average CPU times of the algorithms for 126 optimally solved instances in common, the worst performing algorithm is BB-D, followed by Rec-D. BB-S performs better, but the best performing algorithm is BB-E.

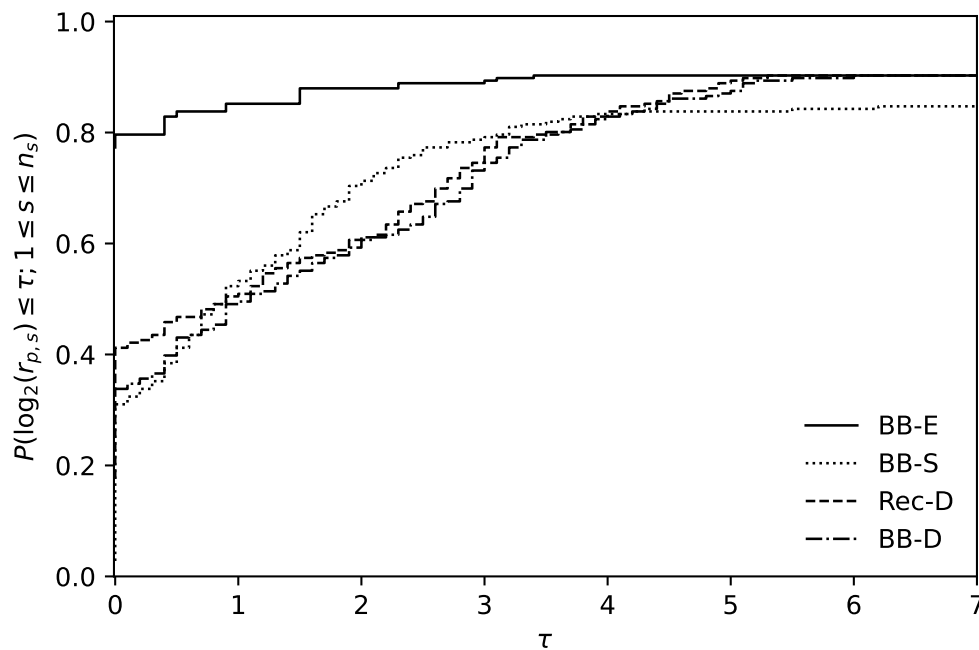


Figure 8.7. Performance profiles: plain BB algorithms.

The performance profile for plain BB algorithms is shown with in Figure 8.7. As indicated by the plot, the best performing algorithm is BB-E. After $\tau \geq 1$, BB-S

performs better than Rec-D and BB-D, which have very similar performances. Rec-D and BB-D perform better than BB-S when $\tau \geq 4$. For $\tau = 6$, BB-E, BB-D, and Rec-D algorithms stabilize at the performance ratio of 0.9. On the other hand, BB-S achieves a performance ratio of 0.8 even for $\tau \geq 6$.

We continue our analysis by comparing the performances of the recursive diving algorithm with BB-D, which are tested using Restricted Test Set Plus with a total number of 216 instances. Both plain versions of the algorithms and the versions with infeasibility test are used. The comparison of diving algorithms in Restricted Test Set Plus are incorporated with Table 8.6. Comparing the plain versions, both algorithms find 135 optimal solutions, but BB-D finds only one feasible solution, which is less than 3 feasible solutions found by Rec-D. CPU time of Rec-D is slightly lower than BB-D for the plain versions. Also, Rec-D explores more nodes than BB-D.

Table 8.6. Comparison of Rec-D and BB-D.

Test Statistic	Diving			
	Plain Rec-D	Plain BB-D	IT Rec-D	IT BB-D
Opt. Sols.	135	135	144	146
Feas. Sols.	3	1	10	12
Explored Nodes	301904792	206467639	179204371	39384260
Active Nodes	15652021	5922888	32869288	33116191
Best UB	3720.76	3751.23	3854.18	3842.63
CPU (secs.)	82.59 (726.63)	94.22 (733.89)	19.16 (637.81)	11.40 (618.3)

Looking at the optimal solutions found with the introduction of infeasibility tests, BB-D finds 146 optimal solutions, which is higher than 144 optimal solutions found by Rec-D. Also, BB-D finds 12 feasible solutions, compared to 10 feasible solutions found by the Rec-D. The average number of explored nodes by BB-D is less than the half of Rec-D. Average CPU time of BB-D is nearly 20 seconds less than the one of Rec-D. In addition, the average CPU time of 135 commonly solved optimal solutions display a remarkable decrease when infeasibility tests are introduced.

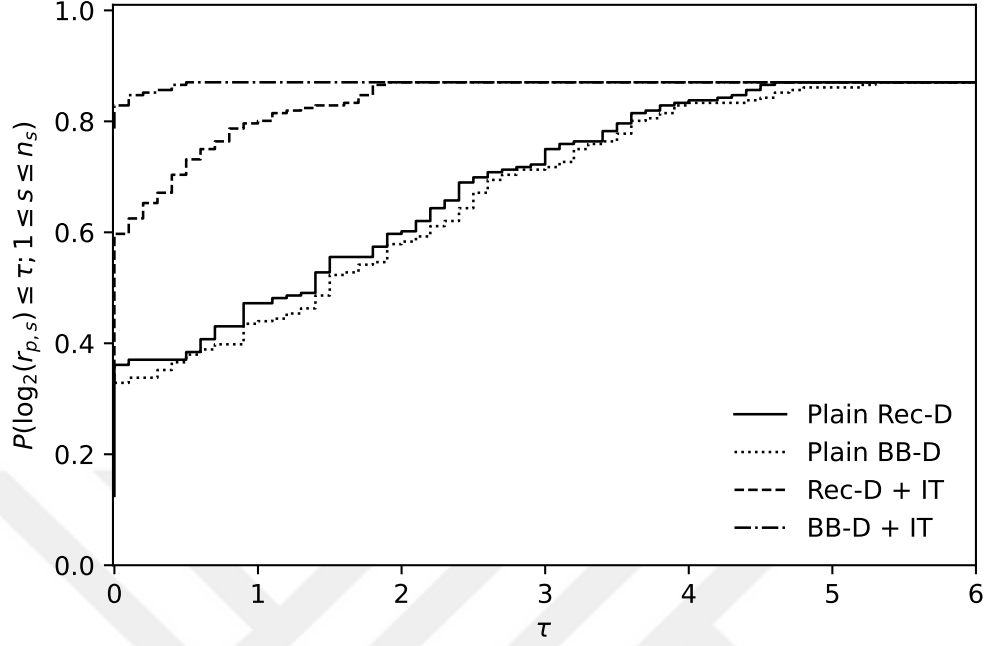


Figure 8.8. Performance profiles: the effect of IT on diving algorithms.

The performance profile for diving algorithms and the effect of infeasibility test is given in Figure 8.8. In the plot, we observe that the best performance is achieved with BB-D+IT, followed by Rec-D+IT, and the performances of plain BB-D and plain Rec-D are worse. In conclusion, BB-D should be used for further benchmarks as it performs better than Rec-D when infeasibility test is introduced.

As an improvement, the effect of probing on BB-D has to be observed. Four different scenarios that include infeasibility test are compared, BB-D without probing, BB-D with probing only at the root node, BB-D with and static probing, where probing is applied with $\beta = 0.8$, and lastly, BB-D with dynamic probing, where probing is applied depending on the conflict density Ω . If $\Omega < 0.15$, then probing is not applied ($\beta = 1$) as the conflict graph becomes very sparse. Probing would have a negative effect instead of any improvement for such cases, as our preliminary case studies suggest. If $\Omega > 0.15$, then probing is applied with $\beta = \Omega - 0.1$. We validate this setting by various test schemes with different conflict densities. The comparison of BB-D with IT and probing algorithms on Restricted Test Set Plus are reported with Table 8.7.

Applying probing only at the root node does not result in any significant improvement in CPU time, as outlined in the table. There are 148 optimal solutions found by IT+Root Pr, which is the same found by only IT. Both versions find 12 feasible solutions. IT+RootPr explores more nodes compared to IT only. For static probing, there are 153 optimal solutions and 15 feasible solutions found, which are higher than the previous algorithms.

Table 8.7. The effect of infeasibility tests and probing on BB-D.

Test Statistic	BB-D			
	IT	IT+ RootPr	IT + StaticPr	IT + DynamicPr
Opt. Sols.	146	146	153	156
Feas. Sols.	12	12	15	12
Explored Nodes	39384260	40692244	3608367	38669434
Active Nodes	33116191	34348685	2815985	33984094
Best UB	3842.63	3842.15	3901.39	4021.19
CPU (secs.)	51.73 (618.3)	51.42 (618.09)	47.32 (601.07)	12.88 (535.6)

The average CPU time (including the CPU times of instances not optimally solved in 1800 seconds) decreases compared to the previous versions, however the average CPU time of 146 commonly optimally solved instances decreases very slightly than the previous ones. This also verifies our preliminary tests in which we found out that applying probing for cases of sparse conflict graphs does not increase performance. Finally, the version with dynamic probing finds 156 optimal solutions, and 12 feasible solutions, with an average CPU time way less than the others.

The performance profiles for the effect of infeasibility tests and probing on BB-D is given in Figure 8.9. It shows that BB-D with the dynamic version of probing clearly outperforms the other versions in all settings. Using static probing and applying probing only at the root node have very similar performance. The worst performance is achieved by BB-D with only IT. We can conclude that if probing is applied, the dynamic version should be preferred.

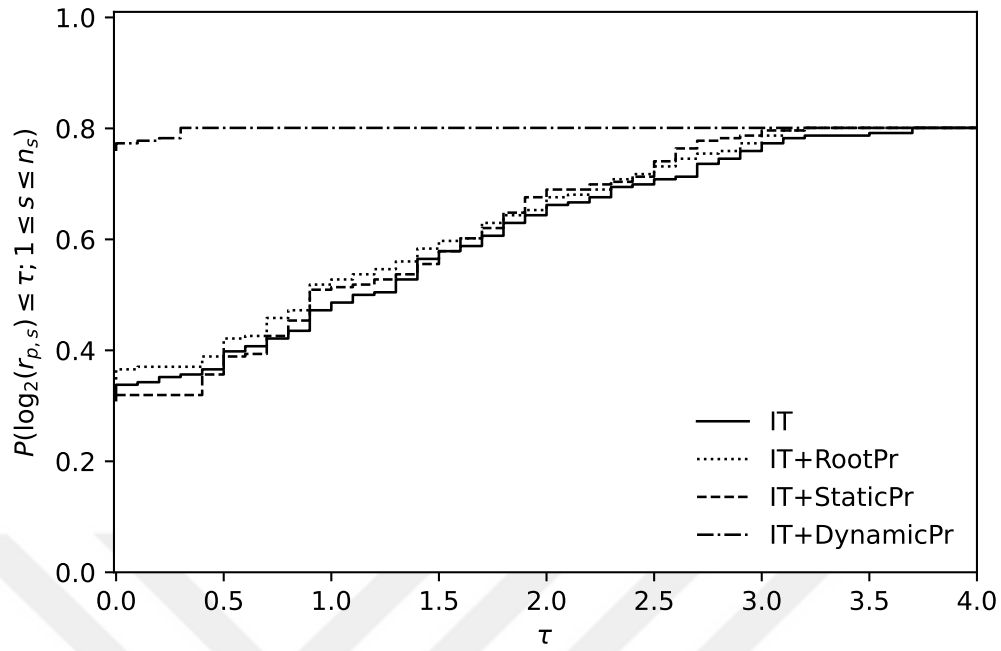


Figure 8.9. Performance profiles: the effect of both IT and Pr on BB-D.

The next step is to observe the effect of infeasibility tests and probing in our BB-E algorithm. Plain version, the version with infeasibility test, and the versions with both infeasibility test and probing are considered. Table 8.8 reports the number of optimal solutions, feasible solutions, average number of explored nodes and active nodes, and the average CPU time for the solution, for finding replacement edges and lower bound calculation in seconds.

Table 8.8. The effect of infeasibility tests and probing on BB-E.

Test Statistic	BB-E			
	Plain	IT	IT+StaticPr	IT+DynamicPr
Opt. Sols.	152	162	160	164
Feas. Sols.	1	0	2	1
Explored Nodes	31900517	30084450	24504034	22869194
Active Nodes	13909241	12847822	11405681	11070642
Best UB	3965.76	4202.83	4207.76	4215.52
CPU (secs.)	76.09 (594.05)	12.91 (497.02)	15.05 (502.98)	11.69 (496.12)
FindRep (secs.)	305.77	237.32	170.58	125.97
LB (secs.)	10.60	6.87	5.27	4.09
IT (secs.)	-	8.69	29.62	23.29
Pr (secs.)	-	-	117.12	195.40

Among 216 instances of the Restricted Test Set Plus, the plain version optimally solves 152 instances, whereas IT optimally solves 162 instances. IT+StaticPr and IT+DynamicPr optimally solve 160 and 164 instances, respectively. Both plain version and IT+DynamicPr find a feasible solution for a single instance, and IT+StaticPr finds feasible solutions for two instances. Average CPU time dramatically decreases with the introduction of infeasibility tests (also for the 151 commonly solved instances optimally), but probing has very little effect on average CPU time. Also, the average CPU time spent to find replacement edges takes 50% of total CPU time in the plain version, which decreases to 48% for IT, 35% for IT+StaticPr, and 25% for IT+DynamicPr. Lower bound calculation takes only 1-2% of total CPU time in all versions. On average, infeasibility test takes 2% of total CPU time in IT, 6% in IT+StaticPr, and 5% in IT+DynamicPr. The average CPU time spent for probing is 23% of total CPU time in static version, and 39% in dynamic version.

The performance profiles for the effect of infeasibility test and probing on BB-E is given with Figure 8.10.

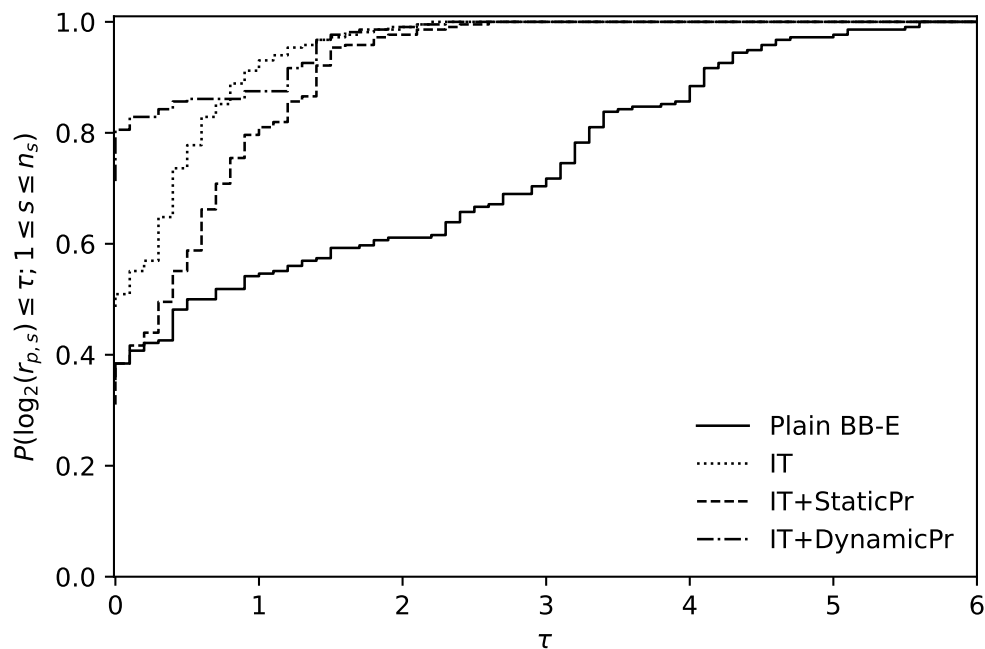


Figure 8.10. Performance profiles: the effect of both IT and Pr on BB-E.

For $\tau < 1$, the version with dynamic probing and infeasibility test performs better than the other algorithms. However, when $\tau \geq 1$, IT performs better than the algorithms with probing. The plain version clearly performs worse than the others. The effect of probing is not very significant on performance, but infeasibility test has a huge impact on the performance of BB-E.

Moving forward, the effect of infeasibility tests and probing on BB-S is analyzed, which is outlined in Table 8.9. The number of optimal solutions increases from 132 found by plain BB-S to 147 with the introduction of infeasibility test, rises further to 148 with static probing and reaches 154 with dynamic probing. Plain version finds 30 feasible solutions, and the version with IT finds 22 feasible solutions. Both IT+StaticPr and IT+DynamicPr find 21 feasible solutions. The average number of explored and active nodes decrease considerably with the addition of infeasibility test and probing. Also, the average best upper bound increases in IT and Pr.

Table 8.9. The effect of infeasibility tests and probing on BB-S.

Test Statistic	BB-S			
	Plain	IT	IT+StaticPr	IT+DynamicPr
Opt. Sols.	132	147	148	154
Feas. Sols.	30	22	21	21
Explored Nodes	15947050	12919165	8855387	12258155
Active Nodes	15086190	12167743	8424177	11621741
Best UB	4115.01	4158.02	4181.56	4226.12
CPU (secs.)	74.25 (745.41)	5.74 (635.25)	4.25 (595.27)	1.83 (557.37)
FindRep (secs.)	442.66	313.43	184.25	206.08
LB (secs.)	15.39	6.21	4.51	5.07
IT (secs.)	-	15.76	117.02	44.25
Pr (secs.)	-	-	227.31	170.84

Average CPU time for all instances and for 132 common optimal solutions found by all versions decrease with the introduction of IT, and decreases even more in both StaticPr and DynamicPr. This pattern is also clearly visible on the average CPU time

spent for finding replacement edges: It consumes 59% of total CPU time in the plain version, which reduces to 49% in IT, 31% in IT+StaticPr, and 37% in IT+DynamicPr. Simple lower bound calculation takes only 1-2% of total CPU time in all versions. Infeasibility test takes 3% of total CPU time in IT, which is 20% in IT+StaticPr and 8% in IT+DynamicPr. Static probing takes 38% of total CPU time, and dynamic probing takes 31% of total CPU time.

The performance profiles for demonstration of the effect of infeasibility test and probing on BB-S is depicted with Figure 8.11.

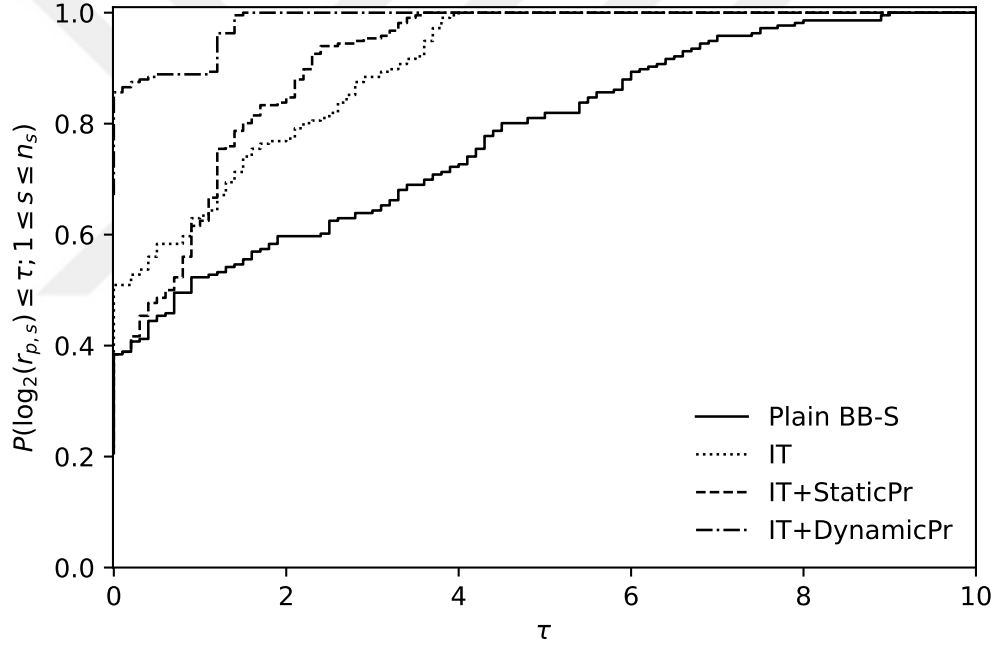


Figure 8.11. Performance profiles: the effect of both IT and Pr on BB-S.

The performance profile clearly demonstrates that BB-S with IT+DynamicPr performs the best. For $\tau \leq 1$, IT performs better than IT+StaticPr. After $\tau = 1$, static probing has a positive effect. Plain BB-S is outperformed by all other algorithms. In conclusion, the version that includes both infeasibility tests and dynamic probing is selected for benchmarking in all BB algorithms.

8.5. Performance of the Branch-and-Bound Algorithms

The next step is to conduct performance tests for benchmarking our best performing algorithm versus the best MSTC formulation using Zhang’s, Carrabs’ and our random datasets. The best MSTC formulation is determined as WEAK-multi for Zhang’s dataset, and STRONG-multi for Carrabs’ and Restricted Test Set. BB-E, BB-S, Rec-D, BB-D with IT and DynamicPr is compared with the reference formulation of the related dataset solved by Gurobi (version 10.0.3).

The performance comparison for 49 Zhang’s instances are included with Table 8.10. Among 49 instances, WEAK-multi finds the highest number of optimal solutions, which is followed by BB-E, BB-D and Rec-D. The lowest number of optimal solutions are found by BB-S. It finds seven feasible solutions, on the other hand Rec-D, BB-D and WEAK-multi find three feasible solutions. Also, WEAK-multi detects seven infeasible solutions, and BB-E finds six infeasible solutions. The other algorithms are unable to detect these infeasibilities.

Table 8.10. Performance of BB algorithms on Zhang’s dataset.

Test Statistic	Algorithms				
	WEAK-multi	BB-E	BB-S	Rec-D	BB-D
Opt. Sols.	35	33	27	30	30
Feas. Sols.	3	0	7	3	3
Infeas. Sols.	7	6	0	0	0
Explored Nodes	82483	29981641	20432343	469268556	250252900
Active Nodes	46601	12849245	19503806	200174042	169843605
Best UB	22393.42	24115.64	23918.15	24216.94	24216.94
CPU (secs.)	0.60 (341.95)	0.04 (463.76)	0.21 (828.13)	0.06 (750.67)	0.06 (757.77)

The number of explored nodes and active nodes are considerably higher in Rec-D and BB-D. The average best UB is achieved by BB-S, which is very close to the average best UB found by WEAK-multi. In terms of average CPU time, WEAK-multi has the lowest overall average CPU time, followed by BB-E. The other algorithms have relatively higher average CPU times, where BB-S performs the worst. Notice that

BB-D performs better compared to Rec-D. Among the 27 commonly solved instances optimally by all algorithms, the average CPU times are very similar. Here, WEAK-multi performs the worst, followed by BB-S, and BB-E performs the best.

The performance profiles for Zhang’s dataset is illustrated with Figure 8.12. The effect of infeasibility tests and probing is apparently visible in the performance profiles. Clearly, WEAK-multi is outperformed by BB-E. Also, BB-D and Rec-D display very similar performances. The worst performance is achieved by BB-S, as it stays way behind WEAK-multi. It should be denoted that the consistency and reliability of Zhang’s dataset, especially for Type 2 instances is open to debate, as they can be solved in CPU times below 1 seconds. They do not provide valuable information for performance profiling, and they can be misleading.

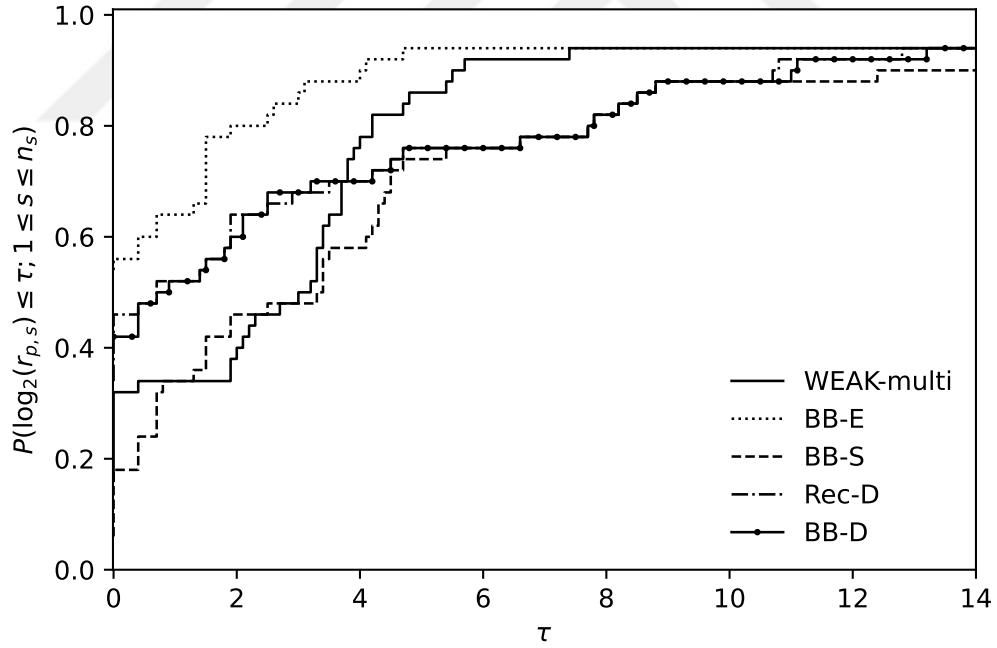


Figure 8.12. Performance profiles: Zhang’s dataset.

The performance comparison for 180 Carrabs’ instances are given in Table 8.11. STRONG-multi is able to solve 117 instances optimally, and it finds 33 feasible solutions. There are 111 optimal solutions found by BB-E, which is followed by 60 optimal solutions found via Rec-D and BB-D. Furthermore, Rec-D finds 68 feasible solutions,

and BB-D finds 73 feasible solutions. BB-S finds the lowest number of optimal solutions, and it finds the highest number of feasible solutions.

Table 8.11. Performance of BB algorithms on Carrabs' dataset.

Test Statistic	Algorithms				
	STRONG-multi	BB-E	BB-S	Rec-D	BB-D
Opt. Sols.	117	111	35	60	60
Feas. Sols.	33	0	106	68	73
Explored Nodes	70905	63117969	51175386	998272123	745542920
Active Nodes	28550	30823764	44107027	817126206	665727067
Best UB	756.07	593.5	876.25	652.56	666.54
CPU (secs.)	0.38 (683.55)	0.06 (745.15)	153.24 (1470.72)	0.03 (1222.56)	0.03 (1226.22)

The lowest best UB is provided by BB-E, followed by Rec-D and BB-D. STRONG-multi has a considerably higher best UB, and the highest best UB is provided by BB-S. BB-E explores more nodes compared to BB-S, and the average number of active nodes is less in BB-E than BB-S. The fastest algorithm is STRONG-multi, being closely followed by BB-E. BB-D and Rec-D have very similar overall average CPU times. The slowest algorithm is BB-S, also it is the slowest considering the 35 commonly optimally solved instances. All other BB algorithms are relatively faster than STRONG-multi for those instances.

The performance profiles for Carrabs' dataset is depicted with Figure 8.13. STRONG-multi is outperformed by BB-E. Other BB algorithms are unable to keep up with STRONG-multi, especially BB-S performs quite worse. Probing does not have any significant effect on this dataset, as one can observe from Figure 8.13. Recalling the sparse conflict density of the instances contained within this dataset, this is an expected result as dynamic probing is applied. Similar results are observed for the case of sparse conflict density among various problems such as knapsack problem with conflicts [68], maximum weight perfect matching problem with conflicts [69], maximum flow problem with conflicts [31], and minimum cost flow problem with conflicts [47].

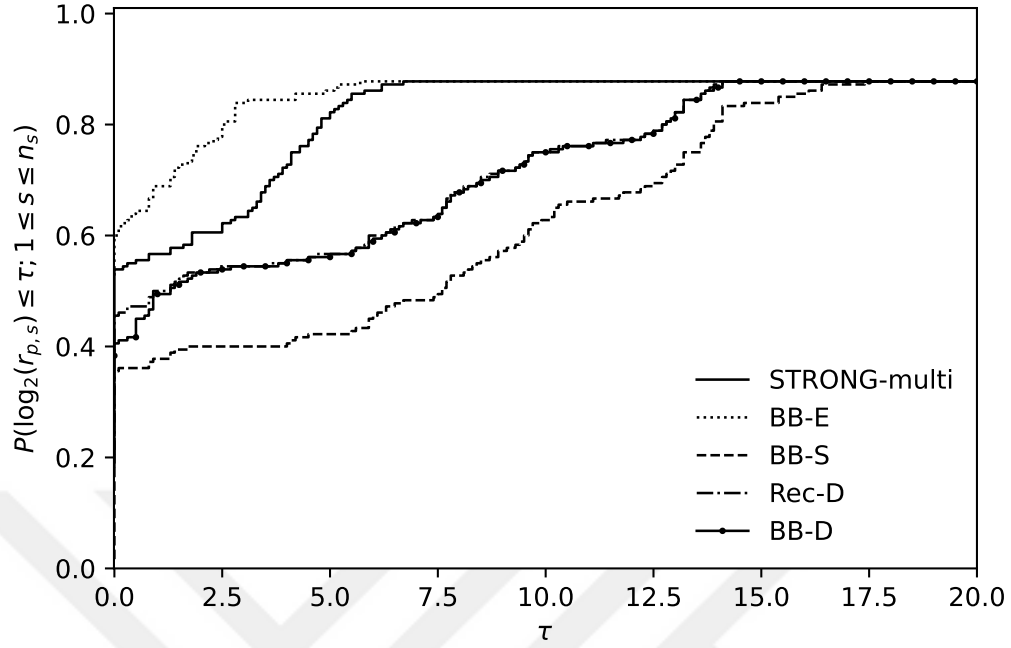


Figure 8.13. Performance profiles: Carrabs' dataset.

The performance comparison for 335 instances in the Extended Random Set are given in Table 8.12. All of our BB algorithms find higher number of optimal solutions compared to 277 optimal solutions found by STRONG-multi. The highest number of optimal solutions are found by BB-E algorithm with a number of 323, followed by BB-D and BB-S with 310 and 307 optimal solutions, respectively. Also, BB-S finds 24 feasible solutions, followed by BB-D with 18 feasible solutions. STRONG-multi finds 15 feasible solutions, Rec-D finds 13 feasible solutions, and there are only two feasible solutions found by BB-E.

Table 8.12. Performance of BB algorithms on random dataset.

Test Statistic	Algorithms				
	STRONG-multi	BB-E	BB-S	Rec-D	BB-D
Opt. Sols.	219	271	267	246	257
Feas. Sols.	14	0	16	7	10
Explored Nodes	30404	9840897	5206718	56701221	18456655
Active Nodes	13893	4685119	4960160	16343373	15640383
Best UB	3885.62	4510.65	4599.10	4294.49	4359.94
CPU (secs.)	91.69 (690.45)	0.38 (383.03)	7.34 (400.03)	16.56 (510.95)	2.22 (447.19)

Rec-D has the highest average number of explored nodes, followed by BB-D. This is another indicator of BB-D's being more efficient than Rec-D, which is apparent also from the average number of active nodes in both algorithms. BB-E explores more nodes than BB-S, but the average number of active nodes is higher for BB-S. The lowest best UB is achieved by STRONG-multi. BB algorithms display a similar pattern among each other, where diving algorithms yield in a lower UB than the others. The fastest algorithm is BB-E, which is followed by BB-S, BB-D, and Rec-D by looking at the average CPU times of all instances, and the average CPU times of 208 commonly solved instances optimally. STRONG-multi is outperformed by all of our BB algorithms.

The performance profiles for the Full Random Set is illustrated with Figure 8.14.

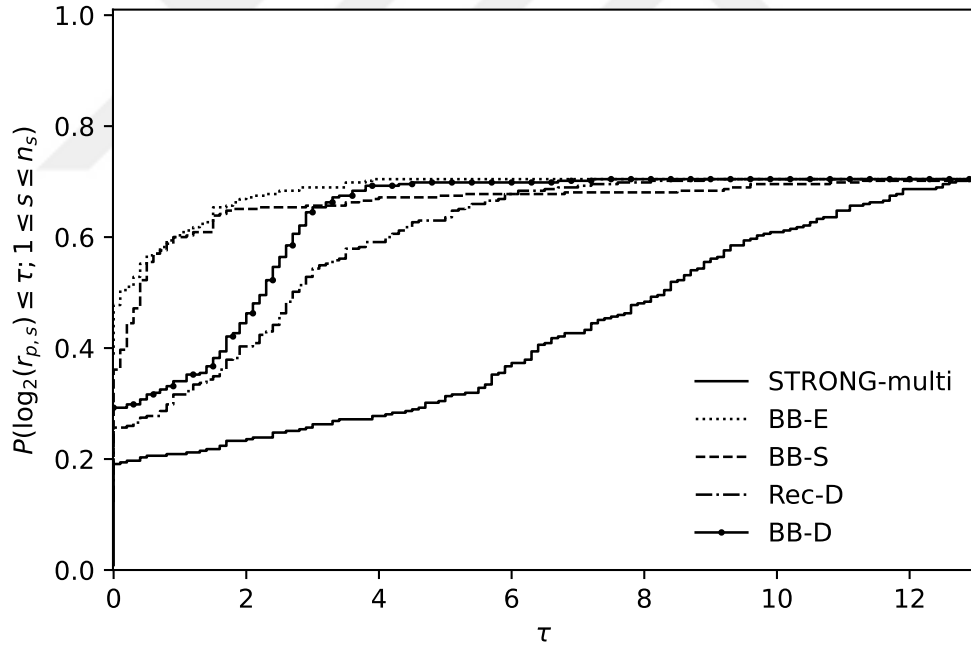


Figure 8.14. Performance profiles: random dataset.

For $\tau = 0$, the best performing algorithm is BB-E, followed closely by BB-S, then by BB-D and Rec-D, respectively. After $\tau \geq 3$, BB-S is outperformed by BB-D, which displays a very similar performance profile with BB-E. At $\tau = 8$, the performance ratios of BB-D, BB-E, and Rec-D reach to the upper bound of 0.7. However, BB-S reaches this upper bound at $\tau = 12$. STRONG-multi performs the worst for all τ values. All

BB algorithms with the addition of infeasibility test and probing clearly outperform STRONG-multi.

8.6. Performance of Edge Replacement Algorithms

Four different algorithms to find replacement edges after the spanning tree becomes disconnected due to deleted edges, which are previously discussed in Section 7.2 in detail, are tested on Restricted Test Set Plus. Plain versions without infeasibility test and probing are used as their client. BB-E is chosen for the implementation of edge replacement algorithms. The comparison of replacement edge algorithms is introduced with Table 8.13.

Table 8.13. Comparison of edge replacement algorithms on small and medium instances.

Test Statistic	Find Replacement Edge			
	BFS	Pred	LCT	DFS
Opt. Sols.	147	147	146	144
Feas. Sols.	0	0	0	0
Explored Nodes	18451530	15844731	18092603	15170273
Active Nodes	7119	7119	7119	7100
Best UB	4013.78	4013.78	4007.71	3995.32
CPU (secs.)	35.20 (615.98)	36.84 (617.17)	42.75 (626.08)	49.65 (633.14)

The best performing edge replacement algorithms are BFS, followed by Pred. Both algorithms find 147 optimal solutions, and their average CPU time in seconds are very close to each other. LCT performs worse, it solves 146 instances optimally in a longer average CPU time. The worst performing edge replacement algorithm is DFS, it only solves 144 instances optimally (which is commonly solved by all) and it has the longest average CPU time among all algorithms.

The performances profile for finding replacement edge algorithms in Restricted Test Set Plus is given in Figure 8.15. In the plot, we observe that the best performance

is achieved with BFS, being closely followed by predecessor array. DFS displays a very poor performance, and LCT performs in between.

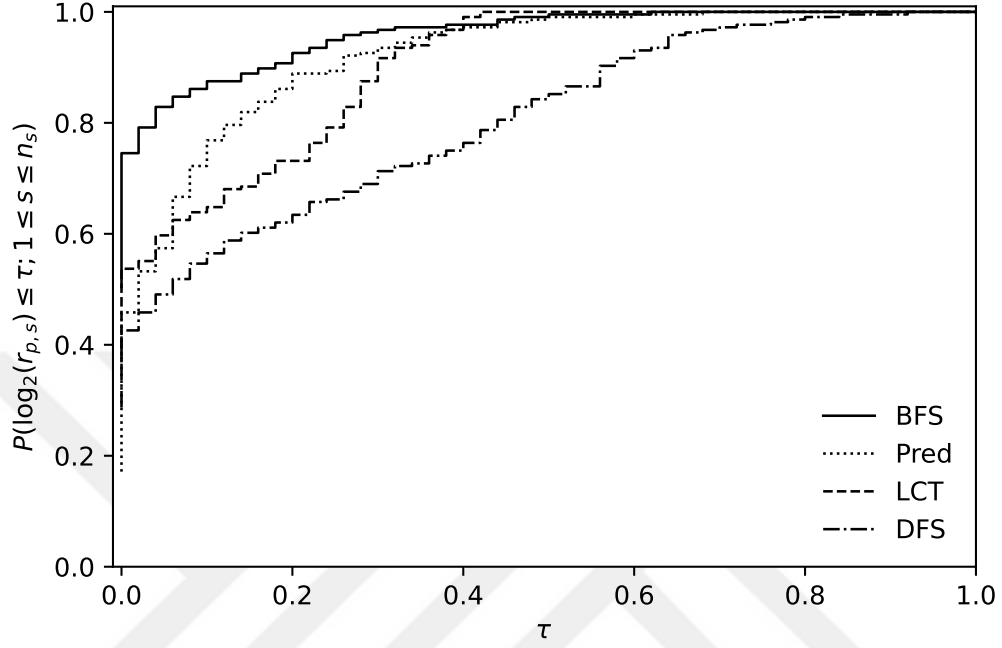


Figure 8.15. Performance profiles: the effect of edge replacement algorithms on small and medium instances.

To test the novel algorithms in a more challenging setting, Large Test Set is used. The comparison of finding replacement edge algorithms in this set is given in Table 8.14. Again, plain versions of the algorithms without infeasibility test and probing are used. This is done in the hope of reaching instance sizes large enough so that overheads due to used ADS become insignificant, and asymptotic worst case bound becomes effective.

Table 8.14. Comparison of edge replacement algorithms on large test instances.

Test Statistic	Find Replacement Edge			
	BFS	Pred	LCT	DFS
Opt. Sols.	7	7	7	7
Feas. Sols.	0	0	0	0
Explored Nodes	93033	89889	90930	91694
Active Nodes	16811	13666	14707	14081
Best UB	79900	79900	79900	79900
CPU (secs.)	372.73 (551.14)	410 (583.76)	488.72 (652.63)	472.89 (638.78)

All algorithms find optimal solutions for all instances except for the instance where $|V(G)| = 1200$. In terms of average CPU time, the best performing replacement edge algorithm is with BFS, followed by the one with predecessor array. DFS performs worse, and the worst performance is achieved by LCT. If we compare the number of explored nodes, BFS has the maximum, followed by DFS, and then LCT. Pred has the minimum number of explored nodes among others.

The performance profiles of edge replacement algorithms in Large Test Set is given in Figure 8.16. In the plot, again, we observe that the best performance is achieved with BFS, being closely followed by Pred.

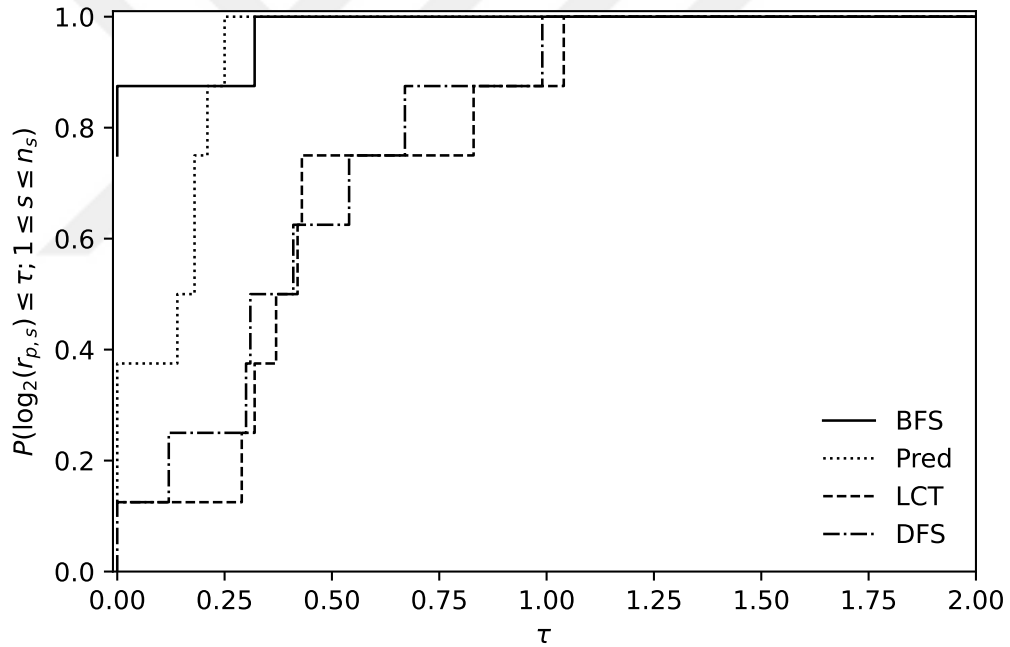


Figure 8.16. Performance profiles: the effect of edge replacement algorithms on large instances.

8.7. Performance of Cycle Detection Algorithms

The next step is to find the cycle-detection algorithm with the highest performance, to be used within diving algorithms. Four different algorithms to detect cycles are tested on Restricted Test Set Plus which contains a total number of 216 instances. The algorithms with infeasibility tests and probing only at the root node are used.

The comparison of cycle detection algorithms is given in Table 8.15. There is no significant difference in the performance of the cycle detection algorithms: all of them find 144 optimal solutions, and their average CPU times are very close to each other. BFS explores considerably more nodes compared to DSU and LCT, being closely followed by Pred. The lowest best UB is achieved with Pred, and it is able to find nine feasible solutions, whereas the other algorithms find only seven feasible solutions.

Table 8.15. Comparison of cycle detection algorithms.

Test Statistic	Cycle detection			
	DSU	BFS	LCT	Pred
Opt. Sols.	144	144	144	144
Feas. Sols.	7	7	7	9
Explored Nodes	119267444	168247760	124088166	166335840
Active Nodes	15820105	38077946	17903421	26784645
Best UB	3893.21	3853.94	3892.13	3865.54
CPU (secs.)	50.21 (633.49)	49.10 (632.75)	49.92 (633.28)	52.26 (634.85)

The performance profiles for cycle detection algorithms is given with Figure 8.17.

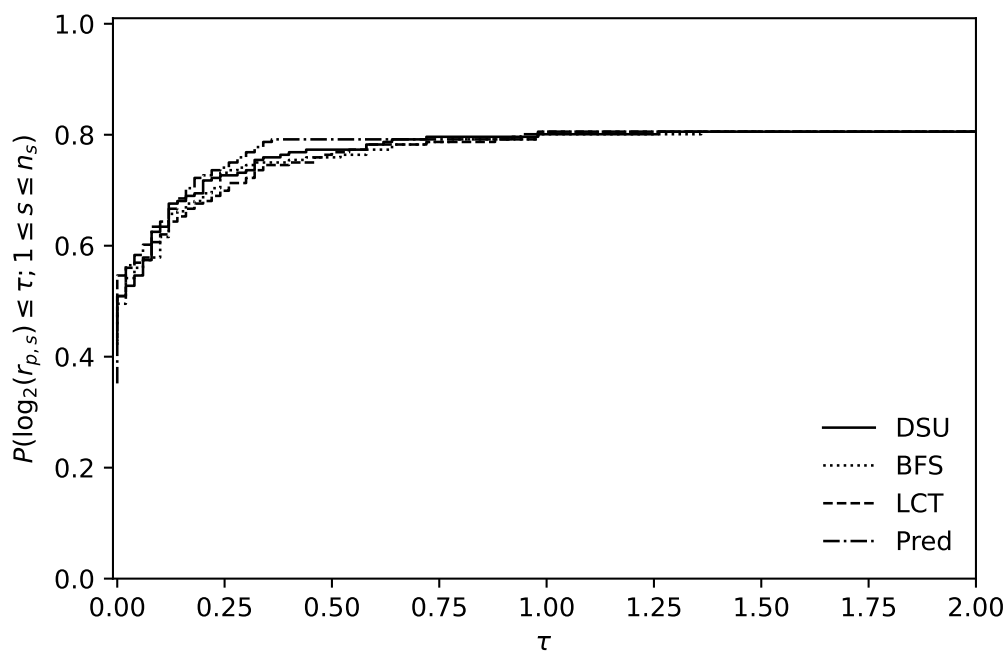


Figure 8.17. Performance profiles: cycle detection algorithms.

In the plot, we observe that the best performance is achieved with Pred, and the performances of the remaining algorithms are very close to each other. Reviewing the performances of different algorithms for finding replacement edges and cycle detection, the selection of predecessor array in BB algorithms is justified. We have implemented our codes using predecessor array, and the performance difference with BFS and disjoint sets is very negligible. Also, updating and traversing the predecessor array is very simple compared to more complex ADS.

8.8. The Effect of Graph and Conflict Densities

Moving forward, we are interested in comparing our BB algorithms with infeasibility tests and dynamic probing to STRONG-multi with respect to different number of vertices, graph densities, and conflict densities. The Extended Random Set is used for this comparison, which includes 335 random instances. Among these instances, 219 of them are solved optimally by STRONG-multi, and 271 of them are solved optimally by BB-E algorithm. BB-S is able to solve 267 instances optimally, whereas Rec-D and BB-D find 246 and 257 optimal solutions, respectively. Initially, the effect of $|V(G)| \in \{20, 30, 40, 50, 75, 100\}$ is outlined with Table 8.16.

Table 8.16. The number of proven optimal solutions: the effect of number of vertices.

$ V(G) $	STRONG-multi	BB-E	BB-S	Rec-D	BB-D
20	55	55	55	55	55
30	54	55	51	53	54
40	47	49	47	43	45
50	35	45	45	39	41
75	18	37	37	31	35
100	10	30	32	25	27
Total	219	271	267	246	257

For all algorithms, the number of optimal solutions decrease as the number of vertices increases. For small instances where $|V(G)| \in \{20, 30, 40\}$; there is nearly no

difference in the number of optimal solutions found by BB algorithms and STRONG-multi. However, when $|V(G)| = 50$, the difference is remarkable: both BB-E and BB-S find 45 optimal solutions, followed by 41 and 39 optimal solutions found by BB-D and Rec-D, respectively, but STRONG-multi only finds 35 solutions. After $|V(G)| \geq 75$, the difference between the number of optimal solutions found by STRONG-multi and our BB algorithms becomes larger. This can be explained by the increasing number of constraints in the formulation, resulting in STRONG-multi not being able to find an optimal solution within the time limit of 1800 seconds for several instances.

The reason for omitting the case of $\Gamma = 0.1$ in our analysis is that we would be unable to create a feasible instance where $|V(G)| = 20$ as we end up with $|E(G)| = 19$. We continue with demonstrating the effect of graph density between 0.2 and 0.8 with increments of 0.1, summarized with Table 8.17.

Table 8.17. The number of proven optimal solutions: the effect of graph density.

Γ	STRONG-multi	BB-E	BB-S	Rec-D	BB-D
0.2	44	45	45	42	43
0.3	38	43	42	41	41
0.4	34	41	41	37	38
0.5	30	38	37	34	37
0.6	27	36	36	33	34
0.7	23	35	34	30	33
0.8	23	33	32	29	31
Total	219	271	267	246	257

Looking at the table, we observe that the number of optimal solutions decrease for all settings as the graph becomes dense. When $\Gamma = 0.2$, the number of optimal solutions found by all algorithms are very similar. As the graph density increases ($\Gamma \geq 0.3$), all BB algorithms find a higher number of optimal solutions compared to STRONG-multi. For all values of Γ , BB-E finds the highest number of optimal solutions, and BB-S performs very similar. BB-D finds a slightly lower number of optimal solutions than BB-S, and Rec-D finds even lower number of optimal solutions.

For the setting where $|V(G)| = 20$ and $\Gamma = 0.2$, $\Omega = 0.8$ is not possible as it results with a number of conflicts being 562. It exceeds the maximum number of possible conflicts for this setting, since there can be at most 361 conflicts between non-tree edges and tree edges, and there can be at most 171 conflicts between non-tree edges, which add up to 532. There is no such problem for other settings. Table 8.18 depicts the effect of conflict density between 0.1 and 0.8 with increments of 0.1.

Table 8.18. The number of proven optimal solutions: the effect of conflict density.

Ω	STRONG-multi	BB-E	BB-S	Rec-D	BB-D
0.1	19	18	12	15	15
0.2	18	22	22	15	18
0.3	24	32	32	25	27
0.4	28	36	36	32	34
0.5	29	39	40	36	39
0.6	31	41	42	40	41
0.7	34	42	42	42	42
0.8	36	41	41	41	41
Total	219	271	267	246	257

We observe that the instances with lower conflict density are harder to solve in all settings. For $\Omega = 0.1$, STRONG-multi finds the highest number of optimal solutions compared to the BB algorithms. Notice that for this setting, BB-S performs the worst. This might be a consequence of the case $|U| = (|V(G)| - 1 - |I^t|)$ being encountered frequently in sparse conflict graphs, and in our implementation of BB-S algorithm, the branching edges for this case are not calculated from reduced costs as suggested in Proposition 5.4 due to increased computational effort. Still, it is possible to say that both BB-E and BB-S outperform STRONG-multi where $\Omega = 0.2$, and BB-D performs the same compared to STRONG-multi. Again, all our BB algorithms perform better than STRONG-multi where $\Omega > 0.3$, as they can solve more instances optimally. When $\Omega \geq 0.7$, all BB algorithms are able to solve all of the instances optimally.

9. CONCLUSION

In this study, four novel branch-and-bound algorithms are proposed to solve the MSTC problem. They use edge branching (BB-E), recursive diving (Rec-D), iterative diving (BB-D), and maximal stable sets (BB-S). The most promising algorithm so far is BB-E. The experimental results demonstrate substantial improvements over traditional BILP models, making the proposed methods a strong alternative for practical applications.

Infeasibility tests are very helpful to reduce problem size, and detect any infeasibility at a very low cost. Trivial lower bounds can be easily calculated for any subproblem, and play an important role to reduce the size of the active node list. Probing becomes useful notably for denser conflict graphs, but at deeper levels of the BB search tree.

A comprehensive analysis and code optimization is crucial for the proposed BB algorithms. The major bottleneck is the determination of the cut after a tree edge is deleted. More advanced ADS such as LCT have been tried. However, updating LCT with splay operations has an additional computational cost, and it does not yield any significant improvement. Another possible improvement is instead of finding replacement edges from elementary cuts, finding them by elementary cycles. Surprisingly, it resulted in the worst performance, as DFS is necessary to update the tree every time an edge is removed from the tree.

The findings of this thesis have far-reaching implications across several domains where optimization under conflict constraints is a critical requirement. In telecommunications, the proposed methods can optimize network configurations to prevent interference or conflicts among transmission channels. Similarly, in transportation and logistics, these algorithms can ensure efficient route planning while adhering to regulatory constraints or physical limitations. In energy distribution, MSTC solutions can

optimize power grid layouts by addressing conflicts such as overlapping power lines or limited resource availability.

The approaches developed in this thesis also contribute to the theoretical advancement of optimization techniques. By integrating innovative solution strategies and formulating the structural properties of conflict graphs, this research provides a framework that can be adapted to other combinatorial problems with similar constraints.

In conclusion, this thesis not only addresses the urgent challenges faced considering the solution of the MSTC problem but also provides a general framework for broader advancements in optimization under complex constraints. These contributions offer practical solutions to real-world problems and open new pathways for research in both applied and theoretical domains.

As future work, the proposed BB algorithms might be improved. There is still room for improvement on the MSTC problem, such as the integration of diving into other BB algorithms such as BB-E and BB-S. Integration of cutting planes with these branching rules, and using LP relaxations instead of MST relaxations are other potential future research directions. Future research can also focus on integrating parallel processing techniques with advanced approaches such as Benders' decomposition, branch-and-price, quantum optimization, and constraint programming in order to enhance performance. Finally, the MSTC problem can be extended to the case of multiple conflict graphs.

REFERENCES

1. Prim, R. C., “Shortest Connection Networks and Some Generalizations”, *The Bell System Technical Journal*, Vol. 36, No. 6, pp. 1389–1401, 1957.
2. Kruskal, J. B., “On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem”, *Proceedings of the American Mathematical Society*, Vol. 7, No. 1, pp. 48–50, 1956.
3. Graham, R. L. and P. Hell, “On the History of the Minimum Spanning Tree Problem”, *Annals of the History of Computing*, Vol. 7, No. 1, pp. 43–57, 1985.
4. Choi, Y., M. Khan, V. A. Kumar and G. Pandurangan, “Energy-optimal Distributed Algorithms for Minimum Spanning Trees”, *IEEE Journal on Selected Areas in Communications*, Vol. 27, No. 7, pp. 1297–1304, 2009.
5. Brimberg, J., P. Hansen, K.-W. Lin, N. Mladenović and M. Breton, “An Oil Pipeline Design Problem”, *Operations Research*, Vol. 51, No. 2, pp. 228–239, 2003.
6. Dutta, S., D. Patra, H. Shankar and P. Alok Verma, “Development of GIS Tool for the Solution of Minimum Spanning Tree Problem Using Prim’s Algorithm”, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XL-8, pp. 1105–1114, Copernicus GmbH, 2014.
7. Bazlamaçcı, C. F. and K. S. Hindi, “Minimum-weight Spanning Tree Algorithms: A Survey and Empirical Study”, *Computers & Operations Research*, Vol. 28, No. 8, pp. 767–785, 2001.
8. Galbiati, G., A. Morzenti and F. Maffioli, “On the Approximability of Some Maximum Spanning Tree Problems”, *Theoretical Computer Science*, Vol. 181, No. 1, pp. 107–118, 1997.

9. Chandy, K. M. and T. Lo, “The Capacitated Minimum Spanning Tree”, *Networks*, Vol. 3, No. 2, pp. 173–181, 1973.
10. Glover, F. and D. Klingman, “Finding Minimum Spanning Trees with a Fixed Number of Links at a Node”, B. Roy (Editor), *Combinatorial Programming: Methods and Applications*, Vol. 19, pp. 191–201, Springer, Dordrecht, 1975.
11. Narula, S. C. and C. A. Ho, “Degree-constrained Minimum Spanning Tree”, *Computers & Operations Research*, Vol. 7, No. 4, pp. 239–249, 1980.
12. Camerini, P. M., “The Min-max Spanning Tree Problem and Some Extensions”, *Information Processing Letters*, Vol. 7, No. 1, pp. 10–14, 1978.
13. Wolsey, L. A., *Integer Programming*, John Wiley & Sons, New York, 1998.
14. Samer, P., *Formulations and Exact Algorithms for the Minimum Spanning Tree Problem with Conflicting Edge Pairs*, Master’s Thesis, Federal University of Minas Gerais, 2014.
15. Darmann, A., U. Pferschy, J. Schauer and G. Woeginger, “Paths, Trees and Matchings under Disjunctive Constraints”, *Discrete Applied Mathematics*, Vol. 159, No. 16, pp. 1726–1735, 2011.
16. Zhang, R., S. Kabadi and A. Punnen, “The Minimum Spanning Tree Problem with Conflict Constraints and Its Variations”, *Discrete Optimization*, Vol. 8, No. 2, pp. 191–205, 2011.
17. Samer, P. and S. Urrutia, “A Branch and Cut Algorithm for Minimum Spanning Trees under Conflict Constraints”, *Optimization Letters*, Vol. 9, No. 1, pp. 41–55, 2015.
18. Carrabs, F., R. Cerulli, R. Pentaneglo and A. Raiconi, “Minimum Spanning Tree with Conflicting Edge Pairs: A Branch-and-cut Approach”, *Annals of Operations*

Research, Vol. 298, No. 1, pp. 65–78, 2021.

19. Carrabs, F. and M. Gaudio, “A Lagrangian Approach for the Minimum Spanning Tree Problem with Conflicting Edge Pairs”, *Networks*, Vol. 78, No. 1, pp. 32–45, 2021.
20. Samer, P. and D. Haugland, “Polyhedral Results and Stronger Lagrangean Bounds for Stable Spanning Trees”, *Optimization Letters*, Vol. 17, pp. 1317–1335, 2023.
21. Fischetti, M. and D. Pisinger, “Optimal Wind Farm Cable Routing: Modeling Branches and Offshore Transformer Modules”, *Networks*, Vol. 72, No. 1, pp. 42–59, 2018.
22. Ulku, I. and C. Alabas-Uslu, “Optimization of Cable Layout Designs for Large Offshore Wind Farms”, *International Journal of Energy Research*, Vol. 44, No. 8, pp. 6297–6312, 2020.
23. Letchford, A. N., F. Rossi and S. Smriglio, “The Stable Set Problem: Clique and Nodal Inequalities Revisited”, *Computers & Operations Research*, Vol. 123, p. 105024, 2020.
24. Berge, C., “Some Classes of Perfect Graphs”, *Six Papers on Graph Theory*, pp. 1–21, Indian Statistical Institute, Calcutta, 1963.
25. Lovász, L., “Normal Hypergraphs and the Perfect Graph Conjecture”, *Discrete Mathematics*, Vol. 2, No. 3, pp. 253–267, 1972.
26. Chvátal, V., “On Certain Polytopes Associated with Graphs”, *Journal of Combinatorial Theory, Series B*, Vol. 18, No. 2, pp. 138–154, 1975.
27. Grötschel, M., L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Second Edition, Springer, Berlin, 1988.

28. Carrabs, F., C. Cerrone and R. Pentangelo, “A Multiethnic Genetic Approach for the Minimum Conflict Weighted Spanning Tree Problem”, *Networks*, Vol. 74, No. 2, pp. 134–147, 2019.
29. Yamada, T., S. Kataoka and K. Watanabe, “Heuristic and Exact Algorithms for the Disjunctively Constrained Knapsack Problem”, *Information Processing Society of Japan Journal*, Vol. 43, No. 9, pp. 2864–2870, 2002.
30. Pferschy, U. and J. Schauer, “The Maximum Flow Problem with Conflict and Forcing Condition”, *International Conference on Network Optimization*, pp. 289–294, Springer, Berlin, Heidelberg, 2011.
31. Şuvak, Z., İ. K. Altinel and N. Aras, “Exact Solution Algorithms for the Maximum Flow Problem with Additional Conflict Constraints”, *European Journal of Operational Research*, Vol. 287, No. 2, pp. 410–437, 2020.
32. Altinel, İ. K., N. Aras, Z. Şuvak and Z. C. Taşkın, “Minimum Cost Noncrossing Flow Problem on Layered Networks”, *Discrete Applied Mathematics*, Vol. 261, pp. 2–21, 2019.
33. Jansen, K., “An Approximation Scheme for Bin Packing with Conflicts”, *Journal of Combinatorial Optimization*, Vol. 3, No. 4, pp. 363–377, 1999.
34. Cao, B., “Transportation Problem with Nonlinear Side Constraints: A Branch and Bound Approach”, *Zeitschrift für Operations Research*, Vol. 36, pp. 185–197, 1992.
35. Öncan, T., Z. Şuvak, M. H. Akyüz and İ. K. Altinel, “Assignment Problem with Conflicts”, *Computers & Operations Research*, Vol. 111, pp. 214–229, 2019.
36. Saffari, S. and Y. Fathi, “Set Covering Problem with Conflict Constraints”, *Computers & Operations Research*, Vol. 143, p. 105763, 2022.
37. Öncan, T., İ. K. Altinel and G. Laporte, “A Comparative Analysis of Several

- Asymmetric Traveling Salesman Problem Formulations”, *Computers & Operations Research*, Vol. 36, No. 3, pp. 637–654, 2009.
38. Magnanti, T. L. and L. A. Wolsey, “Optimal Trees”, *Handbooks in Operations Research and Management Science*, Vol. 7, pp. 503–615, North Holland, Amsterdam, 1995.
 39. Martin, R. K., “Using Separation Algorithms to Generate Mixed Integer Model Reformulations”, *Operations Research Letters*, Vol. 10, No. 3, pp. 119–128, 1991.
 40. Dar, M. A., A. Fischer, J. Martinovic and G. Scheithauer, “Integer Linear Programming Formulations for the Minimum Connectivity Inference Problem and Model Reduction Principles”, *Discrete Optimization*, Vol. 40, p. 100623, 2021.
 41. Floudas, C. A. and P. M. Pardalos, *Encyclopedia of Optimization*, Second Edition, Springer, Berlin, 2008.
 42. Padberg, M. W., “On the Facial Structure of Set Packing Polyhedra”, *Mathematical Programming*, Vol. 5, No. 1, pp. 199–215, 1973.
 43. Della Croce, F. and R. Tadei, “A Multi-KP Modeling for the Maximum-Clique Problem”, *European Journal of Operational Research*, Vol. 73, No. 3, pp. 555–561, 1994.
 44. Zhang, R., *Quadratic Bottleneck Problems: Algorithms, Complexity and Related Topics*, Ph.D. Thesis, Simon Fraser University, 2011.
 45. Cordone, R. and G. Passeri, “Heuristic and Exact Approaches to the Quadratic Minimum Spanning Tree Problem”, *Seventh Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, pp. 52–55, University of Milan, Gargano, Italy, 2008.
 46. Pamuk, B., T. Öncan and İ. K. Altınel, “An Efficient Branch-and-bound Algorithm

for the One-to-many Shortest Path Problem with Additional Disjunctive Conflict Constraints”, SSRN:4599025, 2023.

47. Şuvak, Z., İ. K. Altınel and N. Aras, “Minimum Cost Flow Problem with Conflicts”, *Networks*, Vol. 78, No. 4, pp. 421–442, 2021.
48. Balas, E. and C. S. Yu, “Finding a Maximum Clique in an Arbitrary Graph”, *SIAM Journal on Computing*, Vol. 15, No. 4, pp. 1054–1068, 1986.
49. Balas, E. and J. Xue, “Minimum Weighted Coloring of Triangulated Graphs, with Application to Maximum Weight Vertex Packing and Clique Finding in Arbitrary Graphs”, *SIAM Journal on Computing*, Vol. 20, No. 2, pp. 209–221, 1991.
50. Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
51. Gallai, T., “Maximum-minimum Sätze Über Graphen”, *Acta Mathematica Academiae Scientiarum Hungarica*, Vol. 9, pp. 395–434, 1958.
52. Gallai, T., “Über Extreme Punkt-und Kantenmengen”, *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Mathematica*, Vol. 2, pp. 133–138, 1959.
53. Tarjan, R. E., “Sensitivity Analysis of Minimum Spanning Trees and Shortest Path Trees”, *Information Processing Letters*, Vol. 14, No. 1, pp. 30–33, 1982.
54. Tarjan, R. E. and R. F. Werneck, “Dynamic Trees in Practice”, *Journal of Experimental Algorithmics (JEA)*, Vol. 14, pp. 4.5:1–4.5:21, 2010.
55. Hanauer, K., M. Henzinger and C. Schulz, “Recent Advances in Fully Dynamic Graph Algorithms – A Quick Reference Guide”, *ACM Journal of Experimental Algorithmics*, Vol. 27, pp. 1.11:1–1.11:45, 2022.

56. Bader, D. A. and P. Burkhardt, “A Simple and Efficient Algorithm for Finding Minimum Spanning Tree Replacement Edges”, *Journal of Graph Algorithms and Applications*, Vol. 26, No. 4, pp. 577–588, 2022.
57. Catteneo, G., P. Faruolo, U. F. Petrillo and G. F. Italiano, “Maintaining Dynamic Minimum Spanning Trees: An Experimental Study”, *Discrete Applied Mathematics*, Vol. 158, pp. 404–425, 2010.
58. Sleator, D. D. and R. E. Tarjan, “A Data Structure for Dynamic Trees”, *Journal of Computer and System Sciences*, Vol. 26, No. 3, pp. 362–391, 1983.
59. Gabow, H. N. and R. E. Tarjan, “A Linear-Time Algorithm for a Special Case of Disjoint Set Union”, *Journal of Computer and System Sciences*, Vol. 30, No. 2, pp. 209–221, 1985.
60. Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, Third Edition, MIT Press, Cambridge, Massachusetts, 2009.
61. Galil, Z. and G. F. Italiano, “Data Structures and Algorithms for Disjoint Set Union Problems”, *ACM Computing Surveys (CSUR)*, Vol. 23, No. 3, pp. 319–344, 1991.
62. Conchon, S. and J.-C. Filliâtre, “A Persistent Union-Find Data Structure”, *Proceedings of the ACM Workshop on ML*, pp. 37–46, Freiburg, Germany, 2007.
63. “C⁺⁺20 Reference Documentation”, ISO \ IEC, 2024, <https://en.cppreference.com/w/cpp/20>, accessed on January 15, 2025.
64. “Python 3.9 Documentation”, Python Software Foundation, 2021, <https://docs.python.org/3.9/>, accessed on January 15, 2025.
65. “Gurobi Optimizer v10.0 Documentation”, Gurobi Optimization, LLC, 2023, <https://docs.gurobi.com/10.0/>, accessed on January 15, 2025.

66. Carrabs, F., R. Cerulli, R. Pentaneglo, A. Raiconi, R. Zhang, S. Kabadi and A. Punnen, “Test Instances for the Minimum Spanning Tree with Conflict Constraints Problem”, 2020, <http://www.dipmat2.unisa.it/people/carrabs/www/DataSet/CMST.zip>, accessed on January 15, 2025.
67. Dolan, E. D. and J. J. Moré, “Benchmarking Optimization Software with Performance Profiles”, *Mathematical Programming*, Vol. 91, pp. 201–213, 2002.
68. Bettinelli, A., V. Cacchiani and E. Malaguti, “A Branch-and-bound Algorithm for the Knapsack Problem with Conflict Graph”, *INFORMS Journal on Computing*, Vol. 29, No. 3, pp. 457–473, 2017.
69. Akyüz, M. H., İ. K. Altınel and T. Öncan, “Maximum Weight Perfect Matching Problem with Additional Disjunctive Conflict Constraints”, *Networks*, Vol. 81, No. 4, pp. 465–489, 2023.

APPENDIX A: PROBLEM INSTANCES AND DETAILED RESULTS OF FORMULATIONS

This section introduces test instances for “Restricted Test Set” and detailed results of the solutions with STRONG and WEAK formulations in both single and multiple threaded settings. The tables include the following columns:

- ID: The ID of the MSTC instance.
- Type: Type of Zhang’s instances:
 - i. Type 1a, if known to be infeasible,
 - ii. Type 1b, if feasibility is unknown,
 - iii. Type 2, if known to be feasible.
- Rep: Replication index for instance generation.
- $|V(G)|$: Number of vertices in G .
- Γ : Graph density.
- Ω : Conflict density.
- Explored: Number of explored nodes until termination,
- Active: Number of active nodes during termination,
- Best UB: Best upper bound obtained until termination. Denoted with:
 - i. *, if a feasible solution not proven to be optimal is found until termination,
 - ii. -, if a feasible solution has not been found until termination.

Table A.1. Test instances and comparison of formulations: Zhang’s dataset.

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Type	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	2	100	0.06	0.24	1	1	7968	0.56	1	1	7968	0.13	1	1	7968	0.38	1	1	7968	0.14
1	2	100	0.06	0.28	1	1	8166	0.36	1	1	8166	0.13	1	1	8166	0.44	1	1	8166	0.15
2	1a	100	0.06	0.03	29160	0	-	63.96	53254	0	-	94.88	29229	0	-	441.15	18148	0	-	361.91
3	1b	100	0.06	0.01	6463	886	4041	6.24	2784	212	4041	8.65	1798	0	4041	6.50	1910	0	4041	6.62
4	2	100	0.06	0.19	1	0	7434	0.32	1	0	7434	0.14	1	0	7434	0.31	1	0	7434	0.09
5	1b	100	0.06	0.02	32767	0	5658	44.56	43289	0	5658	67.75	29681	0	5658	330.38	41498	0	5658	496.44
6	1b	100	0.10	0.01	2332	0	4275	3.98	1197	0	4275	4.85	842	119	4275	9.40	1209	0	4275	9.22
7	1a	100	0.10	0.10	7249	0	-	341.94	5972	136	-	87.66	3000	0	-	857.76	15149	0	-	730.81
8	2	100	0.10	0.20	141	0	12652	13.70	88	0	12652	8.97	130	0	12652	12.59	56	0	12652	7.73
9	1b	100	0.10	0.02	290893	0	5997	404.00	307229	0	5997	1079.64	94708	24810	5891*	1800.02	93924	33246	5865*	1800.03
10	2	100	0.10	0.25	1	1	11230	0.92	1	1	11230	0.54	1	0	11232	0.72	1	0	11232	0.39
11	2	100	0.10	0.30	1	1	11481	0.80	1	1	11481	0.46	1	0	11481	0.85	1	0	11481	0.39
12	1b	100	0.10	0.03	288416	105640	7023*	1800.87	360614	237168	6690*	1801.08	54543	28902	6725*	1800.03	90596	57211	6579*	1800.03
13	1a	100	0.10	0.05	341636	206577	8434	1800.76	2253948	1577878	7536	1800.26	31029	16371	7950	1800.02	178961	134952	7414	1800.02
14	2	200	0.02	0.17	1	1	17728	0.16	1	1	17728	0.14	1	0	17728	0.15	1	0	17728	0.15
15	2	200	0.02	0.21	1	1	18617	0.13	1	1	18617	0.12	1	0	18617	0.09	1	0	18617	0.07
16	2	200	0.02	0.26	1	1	19140	0.13	1	1	19140	0.12	1	0	19140	0.09	1	0	19140	0.08
17	1a	200	0.03	0.01	414160	152653	13456	1800.94	324716	116373	13418	1801.20	59675	26579	13304	1800.02	47818	17991	13293	1800.03
18	2	200	0.03	0.19	1	1	20716	0.72	1	0	20716	0.26	1	1	20716	0.65	1	1	20716	0.18
19	1a	200	0.03	0.02	157776	14410	18944	1800.80	50942	0	-	621.21	10371	4854	17301	1800.07	10636	3863	16901	1800.03
20	2	200	0.03	0.24	1	1	18025	1.04	1	1	18025	0.31	1	1	18025	0.84	1	1	18025	0.26
21	2	200	0.03	0.28	1	1	20864	0.88	1	1	20864	0.40	1	1	20864	0.79	1	1	20864	0.26
22	1a	200	0.03	0.03	17	0	-	12.30	2006	1055	-	55.09	17	0	-	14.92	615	354	-	63.90
23	1a	200	0.04	0.05	5633	0	-	814.50	11559	0	-	420.35	1838	244	37018	1800.02	8111	1041	34434	1800.02
24	1a	200	0.04	0.01	268146	127767	20953	1800.42	196584	77293	20974	1800.58	30325	10359	20700	1800.05	30673	12550	20727	1800.02
25	2	200	0.04	0.20	1	1	39895	1.18	1	1	39895	0.59	1	1	39895	0.91	1	1	39895	0.44
26	1a	200	0.04	0.02	151040	96621	26942	1800.31	271080	196135	26025	1800.21	14435	11171	26312	1800.02	31318	19110	25649	1800.02
27	2	200	0.04	0.25	1	1	37671	1.49	1	1	37671	0.48	1	0	37671	1.20	1	0	37671	0.48
28	2	200	0.04	0.29	1	1	38798	1.72	1	1	38798	0.50	1	1	38798	1.38	1	1	38798	0.49
29	1a	200	0.04	0.03	42525	24364	32621	1800.16	117173	53236	30190	1800.83	4726	3043	31163	1800.02	15831	9390	29551	1800.05
30	2	300	0.02	0.24	0	0	76345	2.21	0	0	76345	0.52	0	0	76345	1.85	0	0	76345	0.35
31	2	300	0.02	0.29	0	0	78880	2.14	0	0	78880	0.45	0	0	78880	2.13	0	0	78880	0.39
32	1a	300	0.02	0.03	115	0	-	131.38	2562	1433	-	148.41	217	0	-	499.35	702	324	-	206.92
33	1a	300	0.02	0.01	88750	63423	51666	1800.31	73685	49519	51657	1800.11	14085	11070	51502	1800.03	11809	9675	50796	1800.02
34	2	300	0.02	0.19	1	1	71562	1.90	1	1	71562	0.56	1	1	71562	1.45	1	1	71562	0.36
35	1a	300	0.02	0.02	19195	8223	62772	1800.23	35517	18526	58981	1800.22	2211	1018	60872	1800.04	7491	3685	58275	1800.02
36	2	300	0.01	0.17	0	0	43721	0.13	0	0	43721	0.09	0	0	43721	0.13	0	0	43721	0.06
37	2	300	0.01	0.21	1	1	44267	0.25	1	1	44267	0.13	1	0	44267	0.22	1	0	44267	0.14
38	2	300	0.01	0.25	1	1	43071	0.30	1	1	43071	0.20	15	0	43071	0.38	15	0	43071	0.27
39	1a	300	0.02	0.01	11207	0	-	111.23	2345	1054	-	74.71	1532	0	-	181.78	683	266	-	73.93
40	2	300	0.02	0.19	1	1	43125	0.18	1	1	43125	0.20	1	1	43125	0.15	1	1	43125	0.12
41	2	300	0.02	0.23	1	1	42292	0.22	1	1	42292	0.22	1	1	42292	0.26	1	1	42292	0.13
42	2	300	0.02	0.28	1	1	44114	0.22	1	1	44114	0.20	1	1	44114	0.17	1	1	44114	0.13
43	1b	50	0.16	0.01	622	0	708	2.47	1053	0	708	2.68	232	0	708	1.53	803	85	708	2.17
44	2	50	0.16	0.20	1	0	1636	0.39	1	0	1636	0.17	1	0	1636	0.33	1	0	1636	0.15
45	1b	50	0.16	0.02	2115	0	770	2.63	870	0	770	1.92	1000	49	770	4.01	1441	0	770	3.58
46	2	50	0.16	0.25	1	1	2043	0.23	1	1	2043	0.11	1	1	2043	0.20	1	1	2043	0.09
47	2	50	0.16	0.29	1	1	2338	0.26	1	1	2338	0.10	1	1	2338	0.24	1	1	2338	0.11
48	1b	50	0.16	0.03	1991	0	917	3.50	1615	0	917	3.15	1190	0	917	6.21	1367	0	917	5.51
49	1b	50	0.16	0.05	5788	0	1324	7.26	4043	0	1324	5.79	6783	0	1324	30.26	5854	0	1324	18.68
Average			0.06	0.14	43363	16012	23870	363.75	82483	46601	23789	341.95	7873	2772	24005	444.25	12333	6075	23799	435.87

Table A.2. Test instances and comparison of formulations: Carrabs' dataset.

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Rep	$V[G]$	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
1	1	100	0.30	0.01	10421	0	1079	36.58	8415	0	1079	28.34	6611	0	1079	92.16	7068	0	1079	68.00
2	2	100	0.30	0.01	2240	0	1056	8.72	2540	0	1056	7.32	1301	0	1056	12.17	4077	4	1056	13.09
3	3	100	0.30	0.01	1	2	1059	2.69	155	0	1059	6.09	81	0	1059	5.72	166	1	1059	4.05
4	4	100	0.30	0.01	234	0	1046	5.25	229	0	1046	6.68	361	0	1046	5.73	387	0	1046	6.78
5	5	100	0.30	0.01	18598	0	1072	22.83	6610	0	1072	32.24	3361	0	1072	26.85	2682	0	1072	32.31
6	1	100	0.30	0.04	60778	41864	1284*	1800.69	98305	56506	1346*	1800.60	11939	8371	1292*	1800.04	30911	19112	1441*	1800.04
7	2	100	0.30	0.04	38184	27463	1301*	1800.58	96690	55630	1348*	1800.35	11701	9374	1364*	1800.08	24911	16563	1347*	1800.03
8	3	100	0.30	0.04	37870	29491	1299*	1800.67	126202	81460	1311*	1800.94	17156	9691	1299*	1800.04	30320	17761	1338*	1800.04
9	4	100	0.30	0.04	36085	27225	1291*	1800.53	95506	63739	1324*	1800.81	12578	9839	1320*	1800.03	24034	13472	1322*	1800.04
10	5	100	0.30	0.04	46699	24584	1269*	1800.45	99323	57955	1312*	1800.98	11080	7969	1278*	1800.07	27446	15483	1311*	1800.03
11	1	100	0.30	0.07	36391	26499	-	1801.32	282863	229080	-	1800.22	5208	4071	-	1800.03	61212	57688	-	1800.03
12	2	100	0.30	0.07	36505	26812	-	1801.00	438478	400275	-	1800.21	4848	3623	-	1800.03	27627	20054	-	1800.03
13	3	100	0.30	0.07	36945	26810	-	1802.20	493275	451654	-	1800.09	4624	3578	-	1800.03	36018	24966	-	1800.03
14	4	100	0.30	0.07	35467	25911	-	1800.58	448351	408966	-	1800.22	4644	3408	-	1800.04	32964	23165	-	1800.03
15	5	100	0.30	0.07	36351	26499	-	1804.37	415335	376659	-	1800.19	5415	4188	-	1800.05	31153	22241	-	1800.03
16	1	100	0.40	0.07	26022	19905	-	1800.62	188154	171454	-	1800.39	3833	2884	-	1800.05	19654	17854	-	1800.03
17	2	100	0.40	0.07	30168	22663	-	1800.72	55473	41073	-	1800.37	3427	2553	-	1800.05	23988	22383	-	1800.03
18	3	100	0.40	0.07	21757	17690	-	1800.59	47044	33045	-	1800.25	3119	2330	-	1800.05	19941	18118	-	1800.03
19	4	100	0.40	0.07	28359	21608	-	1800.78	35784	27850	-	1800.70	3895	2847	-	1800.07	18862	17331	-	1800.03
20	5	100	0.40	0.07	28200	21423	-	1800.49	36247	27418	-	1800.31	3541	2647	-	1800.05	19380	17811	-	1800.03
21	1	100	0.40	0.01	639	0	1024	9.25	49	0	1024	6.30	165	0	1024	5.50	1	0	1024	4.13
22	2	100	0.40	0.01	89	0	1025	4.48	1	0	1025	4.19	1	0	1025	3.21	1	0	1025	3.00
23	3	100	0.40	0.01	149504	0	1028	56.19	126058	0	1028	69.38	118603	49	1028	120.49	108785	1	1028	148.70
24	4	100	0.40	0.01	1	0	1031	7.95	111	0	1031	7.77	32	5	1031	6.92	158	8	1031	3.83
25	5	100	0.40	0.01	1	0	1036	4.68	1	2	1036	4.54	1	1	1036	5.27	1	0	1036	8.87
26	1	100	0.40	0.04	36128	29212	1242*	1803.26	36421	26067	1247*	1800.48	9840	7806	1217*	1800.07	12176	9813	1240*	1800.04
27	2	100	0.40	0.04	35807	27141	1212*	1800.38	37360	31041	1217*	1800.49	10236	8342	1207*	1800.15	13131	7007	1228*	1800.03
28	3	100	0.40	0.04	36914	28672	1224*	1803.10	42578	32298	1216*	1800.51	10238	7439	1230*	1800.08	15309	11970	1270*	1800.03
29	4	100	0.40	0.04	37220	29317	1206*	1802.56	37031	30210	1247*	1800.71	10238	7646	1237*	1800.21	13658	9368	1245*	1800.03
30	5	100	0.40	0.04	35941	28382	1194*	1802.09	36453	28622	1175*	1800.88	10202	8163	1183*	1800.21	14878	11310	1230*	1800.04
31	1	100	0.20	0.04	136415	77885	1409*	1800.75	877529	726881	1537*	1800.50	38066	33293	1623*	1800.05	57097	40649	1417*	1800.04
32	2	100	0.20	0.04	168819	95103	1513*	1800.61	887501	757081	1639*	1800.85	25916	13466	1483*	1800.04	109615	87352	1540*	1800.06
33	3	100	0.20	0.04	138319	83126	1438*	1800.55	593218	437742	1438*	1800.59	39674	25084	1546*	1800.04	193466	169992	-	1800.03
34	4	100	0.20	0.04	127512	75273	1399*	1800.60	955737	747080	1493*	1800.32	45201	24893	1494*	1800.04	88458	64461	1482*	1800.03
35	5	100	0.20	0.04	162575	84936	1515*	1800.84	463029	407336	1611*	1800.50	39307	26339	1670*	1800.04	97737	74317	1502*	1800.04
36	1	100	0.20	0.07	35677	26177	-	1800.29	1135617	1000102	-	1800.26	6465	5634	-	1800.03	128537	119127	-	1800.02
37	2	100	0.20	0.07	35956	27114	-	1800.30	950840	848308	-	1800.40	6266	5251	-	1800.04	111603	104158	-	1800.02
38	3	100	0.20	0.07	36523	26869	-	1800.26	1090916	974175	-	1800.24	6462	5502	-	1800.02	136409	127303	-	1800.03
39	4	100	0.20	0.07	36125	26812	-	1800.27	1218858	1086660	-	1800.21	6706	5837	-	1800.05	139169	129400	-	1800.02
40	5	100	0.20	0.07	36029	26822	-	1800.63	931250	824544	-	1800.22	6690	5578	-	1800.02	82975	67836	-	1800.03
41	1	100	0.20	0.01	14153	0	1119	19.85	11938	0	1119	19.67	4855	0	1119	14.74	6787	0	1119	19.18
42	2	100	0.20	0.01	9613	0	1137	30.59	10559	0	1137	26.32	7148	0	1137	66.77	7127	0	1137	28.44
43	3	100	0.20	0.01	6566	0	1113	28.70	7319	0	1113	15.63	1780	113	1113	13.02	3502	1	1113	13.09
44	4	100	0.20	0.01	211	0	1110	4.18	202	0	1110	4.64	931	176	1110	12.55	1053	152	1110	12.78
45	5	100	0.20	0.01	136	0	1090	8.37	485	0	1090	7.50	58	0	1090	2.50	478	0	1090	6.33
46	1	25	0.40	0.04	1	1	321	0.33	1	2	321	0.96	133	0	321	0.41	162	2	321	0.93
47	2	25	0.40	0.04	1	1	317	0.37	1	0	317	0.37	1	1	317	0.26	1	0	317	0.39
48	3	25	0.40	0.04	1	0	284	0.16	1	1	284	0.16	1	0	284	0.12	1	0	284	0.18
49	4	25	0.40	0.04	1	0	311	0.29	1	0	311	0.69	1	0	311	0.24	1	0	311	0.61
50	5	25	0.40	0.04	1	1	290	0.18	1	0	290	0.30	1	0	290	0.17	1	0	290	0.27

Table A.2 . Test instances and comparison of formulations: Carrabs' dataset. (cont..)

Instances			STRONG-multi				WEAK-multi				STRONG-single				WEAK-single			
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active
51	1	25	0.40	0.07	229	0	329	1.25	279	0	329	1.93	413	0	329	0.99	462	0
52	2	25	0.40	0.07	1	1	339	0.89	1	2	339	1.25	46	0	339	0.74	248	0
53	3	25	0.40	0.07	1	2	368	1.51	140	0	368	1.46	20	0	368	0.68	428	0
54	4	25	0.40	0.07	1	1	311	0.56	1	1	311	0.56	1	0	311	0.55	1	0
55	5	25	0.40	0.07	1	1	321	0.46	1	0	321	0.94	1	0	321	0.59	79	0
56	1	25	0.40	0.01	1	1	282	0.11	1	1	282	0.13	1	1	282	0.09	1	1
57	2	25	0.40	0.01	1	1	294	0.54	1	1	294	0.26	1	0	294	0.30	1	0
58	3	25	0.40	0.01	1	0	284	0.12	1	0	284	0.14	1	0	284	0.09	1	0
59	4	25	0.40	0.01	12	0	281	0.59	12	0	281	0.57	1	1	281	0.60	1	1
60	5	25	0.40	0.01	1	0	292	0.20	1	0	292	0.15	1	0	292	0.13	1	0
61	1	25	0.20	0.07	1	0	385	0.19	1	1	385	0.25	7	0	385	0.40	1	1
62	2	25	0.20	0.07	1	0	432	0.13	1	0	432	0.15	1	0	432	0.12	1	0
63	3	25	0.20	0.07	58	0	458	0.34	45	0	458	0.60	69	0	458	0.43	258	0
64	4	25	0.20	0.07	1	1	400	0.20	1	1	400	0.41	1	2	400	0.21	51	0
65	5	25	0.20	0.07	1	0	420	0.23	1	0	420	0.45	1	0	420	0.25	1	1
66	1	25	0.20	0.01	1	1	347	0.13	1	1	347	0.15	1	0	347	0.18	1	0
67	2	25	0.20	0.01	1	1	353	0.12	1	1	353	0.14	1	1	353	0.12	1	1
68	3	25	0.20	0.01	1	1	346	0.08	1	1	346	0.09	1	1	346	0.06	1	1
69	4	25	0.20	0.01	1	0	336	0.11	1	0	336	0.13	1	0	336	0.09	1	0
70	5	25	0.20	0.01	1	1	389	0.11	1	1	389	0.12	1	0	389	0.14	1	0
71	1	25	0.20	0.04	1	2	381	0.47	1	2	381	0.53	34	0	381	0.33	40	0
72	2	25	0.20	0.04	35	0	390	0.35	1	0	390	0.41	51	0	390	0.33	51	0
73	3	25	0.20	0.04	1	0	372	0.20	1	0	372	0.29	1	0	372	0.14	1	0
74	4	25	0.20	0.04	1	0	357	0.13	1	0	357	0.16	1	0	357	0.27	1	0
75	5	25	0.20	0.04	1	0	406	0.25	1	0	406	0.16	1	0	406	0.22	1	0
76	1	25	0.30	0.04	1	0	305	0.58	1	1	305	0.48	1	0	305	0.46	1	1
77	2	25	0.30	0.04	1	0	339	0.13	1	0	339	0.16	1	0	339	0.11	1	0
78	3	25	0.30	0.04	1	0	344	0.13	1	0	344	0.24	1	0	344	0.09	1	0
79	4	25	0.30	0.04	1	1	329	0.53	1	0	329	0.63	1	0	329	0.35	1	0
80	5	25	0.30	0.04	1	2	326	0.66	1	2	326	0.80	21	0	326	0.80	66	0
81	1	25	0.30	0.07	1	1	349	0.18	1	1	349	0.73	1	2	349	0.22	15	0
82	2	25	0.30	0.07	275	0	385	1.24	374	0	385	1.72	331	0	385	1.11	410	0
83	3	25	0.30	0.07	1	1	335	0.25	1	1	335	0.32	1	1	335	0.18	1	1
84	4	25	0.30	0.07	133	0	348	0.71	115	0	348	0.92	351	0	348	0.78	247	0
85	5	25	0.30	0.07	1	0	357	0.29	1	0	357	0.39	1	0	357	0.21	40	0
86	1	25	0.30	0.01	1	0	299	0.23	1	1	299	0.24	1	0	299	0.46	23	0
87	2	25	0.30	0.01	1	0	297	0.30	1	0	297	0.32	1	1	297	0.23	1	1
88	3	25	0.30	0.01	1	0	318	0.17	1	0	318	0.19	1	0	318	0.18	1	0
89	4	25	0.30	0.01	1	2	311	0.30	1	1	311	0.37	1	2	311	0.32	1	1
90	5	25	0.30	0.01	1	1	306	0.11	1	1	306	0.12	1	0	306	0.08	1	0
91	1	50	0.20	0.04	573	0	678	2.89	414	0	678	2.65	571	48	678	3.78	882	35
92	2	50	0.20	0.04	320	0	681	2.17	514	0	681	3.15	1028	64	681	6.42	1371	0
93	3	50	0.20	0.04	4269	0	709	6.34	10433	0	709	10.58	3717	0	709	16.21	7797	0
94	4	50	0.20	0.04	1	1	639	1.38	924	0	639	2.92	467	0	639	2.77	655	82
95	5	50	0.20	0.04	2945	0	681	5.77	5020	385	681	10.82	2518	0	681	12.55	2255	0
96	1	50	0.20	0.07	322529	0	816	672.74	490653	0	816	757.20	264295	17637	816*	1800.03	301451	111135
97	2	50	0.20	0.07	60689	0	835	74.82	206503	0	835	269.72	81532	0	835	967.98	136589	0
98	3	50	0.20	0.07	245228	0	810	547.61	1436979	94448	810*	1800.23	172140	20994	812*	1800.03	311109	121348
99	4	50	0.20	0.07	64275	0	832	106.73	232989	0	832	257.88	54724	0	832	428.15	158528	0
100	5	50	0.20	0.07	6550	0	769	16.75	53438	0	769	47.86	5568	0	769	32.28	25627	0
101	1	50	0.20	0.01	1	0	619	1.44	313	0	619	3.61	541	429	619	1.49	528	303

Table A.2 . Test instances and comparison of formulations: Carrabs' dataset. (cont.)

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
102	2	50	0.20	0.01	1	2	604	0.69	27	16	604	0.86	86	2	604	0.98	25	0	604	0.97
103	3	50	0.20	0.01	1	0	634	0.23	1	1	634	0.26	1	0	634	0.14	1	0	634	0.24
104	4	50	0.20	0.01	1	0	616	1.01	27	19	616	1.22	10	1	616	0.63	347	0	616	1.18
105	5	50	0.20	0.01	1	0	595	1.25	1141	51	595	2.37	1	0	595	0.49	0	0	595	1.41
106	1	50	0.30	0.04	2431	0	612	8.17	4471	0	612	19.99	3170	0	612	17.92	4907	0	612	27.99
107	2	50	0.30	0.04	3222	0	615	7.84	13150	0	615	21.92	2387	0	615	14.99	8265	0	615	36.84
108	3	50	0.30	0.04	1381	0	587	5.00	3204	0	587	6.12	1165	22	587	8.15	1108	132	587	11.57
109	4	50	0.30	0.04	4237	4	634	9.80	9017	0	634	21.12	5819	0	634	30.31	9561	0	634	47.05
110	5	50	0.30	0.04	30	2	643	2.73	1411	18	643	5.59	452	3	643	3.86	577	264	643	6.95
111	1	50	0.30	0.07	560991	50530	726*	1800.40	597909	382477	767*	1800.74	138336	45199	729*	1800.03	192579	112860	737*	1800.04
112	2	50	0.30	0.07	224713	0	744	719.77	614921	265554	758*	1800.72	141515	64540	764*	1800.03	211072	141882	786*	1800.04
113	3	50	0.30	0.07	487250	68062	758*	1800.87	617048	341704	786*	1800.78	130091	41692	758*	1800.03	202314	127908	767*	1800.03
114	4	50	0.30	0.07	543536	79658	697*	1801.15	621323	321811	706*	1801.13	139724	65912	711*	1800.03	221541	158564	729*	1800.04
115	5	50	0.30	0.07	794437	236044	769*	1800.63	603178	353840	781*	1800.75	150075	66155	770*	1800.03	198414	128821	781*	1800.04
116	1	50	0.30	0.01	1	0	570	0.47	1	0	570	0.96	1	0	570	0.50	1	0	570	0.64
117	2	50	0.30	0.01	387	0	561	3.07	307	0	561	2.55	563	162	561	3.02	218	0	561	1.85
118	3	50	0.30	0.01	1	0	573	0.44	1	1	573	0.51	1	0	573	0.39	1	0	573	0.50
119	4	50	0.30	0.01	1	0	560	0.72	1	0	560	0.68	1	0	560	0.62	1	0	560	1.22
120	5	50	0.30	0.01	1591	0	549	2.65	1350	0	549	3.44	1110	134	549	2.89	809	0	549	1.76
121	1	50	0.40	0.01	1	0	548	1.40	1	0	548	1.45	1	0	548	0.57	1	0	548	0.88
122	2	50	0.40	0.01	1	0	530	1.53	1	2	530	2.01	23	0	530	1.12	12	3	530	1.70
123	3	50	0.40	0.01	1	0	549	0.81	1	0	549	0.86	22	0	549	0.76	1	0	549	0.70
124	4	50	0.40	0.01	4905	0	540	8.37	5804	0	540	5.05	2828	0	540	7.46	1732	117	540	5.89
125	5	50	0.40	0.01	1	0	540	1.28	1	1	540	1.47	1	1	540	0.38	1	0	540	0.98
126	1	50	0.40	0.04	322	0	594	6.35	1804	0	594	7.60	338	0	594	5.98	1733	0	594	16.16
127	2	50	0.40	0.04	14439	0	579	29.95	12373	0	579	32.96	8524	0	579	47.32	16543	0	579	82.98
128	3	50	0.40	0.04	1	1	589	3.87	1	2	589	6.18	37	0	589	4.12	344	0	589	4.33
129	4	50	0.40	0.04	2105	0	577	8.59	3552	0	577	10.69	1965	0	577	10.79	1540	0	577	10.59
130	5	50	0.40	0.04	662	0	592	5.52	1270	0	592	6.61	624	284	592	13.60	2815	0	592	23.53
131	1	50	0.40	0.07	301827	103158	665*	1806.32	742372	429933	667*	1800.23	104928	55888	673*	1800.03	154298	91783	673*	1800.03
132	2	50	0.40	0.07	325960	44991	640*	1801.36	717415	330150	646*	1800.15	109707	48097	653*	1800.05	160046	109050	664*	1800.03
133	3	50	0.40	0.07	186175	0	672	944.73	874936	496480	688*	1800.16	144646	21826	672*	1800.03	172123	96370	684*	1800.03
134	4	50	0.40	0.07	211641	0	677	1001.02	598458	0	677	1643.15	108975	40402	689*	1800.03	149696	97897	700*	1800.03
135	5	50	0.40	0.07	173526	0	657	692.97	695248	449320	682*	1800.41	128280	53155	669*	1800.03	146118	90436	672*	1800.03
136	1	75	0.40	0.04	118625	19279	856*	1800.56	290585	185759	867*	1800.27	38885	9176	854*	1800.06	59467	31828	858*	1800.03
137	2	75	0.40	0.04	140390	50741	848*	1800.64	204648	117552	846*	1801.08	29955	11402	847*	1800.04	46360	27634	849*	1800.03
138	3	75	0.40	0.04	124592	67110	896*	1800.62	210620	141355	879*	1800.54	32547	15000	875*	1800.04	46135	29484	884*	1800.03
139	4	75	0.40	0.04	130102	67363	883*	1800.56	215699	140805	871*	1800.35	28825	10691	862*	1800.03	48726	31364	877*	1800.03
140	5	75	0.40	0.04	137116	74668	881*	1800.92	219546	147158	874*	1800.41	29219	14137	869*	1800.05	42335	24896	862*	1800.03
141	1	75	0.40	0.07	195089	171378	-	1800.42	847840	766540	-	1800.23	29436	20553	-	1800.02	147426	137511	-	1800.03
142	2	75	0.40	0.07	201586	179905	-	1800.75	942020	850345	-	1800.28	27409	19270	-	1800.03	130062	112482	-	1800.02
143	3	75	0.40	0.07	195058	172303	-	1800.58	894947	803858	-	1800.14	26229	18671	-	1800.02	149921	138858	-	1800.03
144	4	75	0.40	0.07	147320	102901	-	1800.49	1010672	913527	-	1800.23	25294	18189	-	1800.03	130724	112889	-	1800.02
145	5	75	0.40	0.07	184579	163128	-	1800.79	852241	769392	-	1800.21	28319	19928	-	1800.02	127354	109077	-	1800.03
146	1	75	0.40	0.01	627	0	787	2.76	1	0	787	3.23	1	0	787	1.25	1	0	787	2.86
147	2	75	0.40	0.01	475	0	785	6.50	233	0	785	5.71	430	0	785	4.38	1111	0	785	6.58
148	3	75	0.40	0.01	1	0	783	2.05	1	2	783	1.48	12	0	783	1.34	1	0	783	1.81
149	4	75	0.40	0.01	52	0	784	3.59	46	0	784	2.64	70	0	784	1.75	178	0	784	1.66
150	5	75	0.40	0.01	132	0	797	4.46	63	26	797	2.92	1	2	797	2.21	1	2	797	3.01
151	1	75	0.20	0.07	364399	299314	-	1800.19	2271509	1872641	-	1800.56	30935	19267	-	1800.02	288306	244588	-	1800.03

Table A.2 . Test instances and comparison of formulations: Carrabs' dataset. (cont.)

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
152	2	75	0.20	0.07	330656	244499	-	1800.33	2308020	1902241	-	1800.19	36929	22624	-	1800.03	317295	270715	-	1800.03
153	3	75	0.20	0.07	232333	167610	-	1800.18	2608777	2170996	-	1800.17	30281	19576	-	1800.02	298644	255890	-	1800.03
154	4	75	0.20	0.07	246976	174244	-	1800.43	2277312	1835353	-	1800.12	33159	21129	-	1800.02	360416	313310	-	1800.03
155	5	75	0.20	0.07	276667	200579	-	1800.51	2434159	1984003	-	1800.34	33332	20373	-	1800.02	331219	278098	-	1800.02
156	1	75	0.20	0.01	1578	52	868	4.43	42	19	868	2.51	117	0	868	2.26	155	2	868	1.72
157	2	75	0.20	0.01	245	0	871	3.67	1868	0	871	4.96	696	242	871	4.95	737	185	871	5.21
158	3	75	0.20	0.01	2602	0	838	4.15	59	34	838	3.60	560	3	838	1.94	258	0	838	1.71
159	4	75	0.20	0.01	3155	353	855	8.62	2359	0	855	5.37	1004	0	855	3.02	1308	224	855	6.82
160	5	75	0.20	0.01	182	0	857	1.91	4335	3296	857	7.61	202	0	857	2.13	752	427	857	6.50
161	1	75	0.20	0.04	263043	0	1047	1605.37	555249	370990	1078*	1801.10	62512	24380	1048*	1800.03	132762	97565	1088*	1800.03
162	2	75	0.20	0.04	280497	0	1036	1725.38	512393	344744	1069*	1801.07	76702	39714	1056*	1800.03	111618	71784	1049*	1800.03
163	3	75	0.20	0.04	259872	105605	1037*	1801.36	495243	231565	1033*	1800.93	68681	32619	1028*	1800.03	137472	83520	1053*	1800.04
164	4	75	0.20	0.04	164090	0	992	986.22	446235	0	992	1167.13	66570	17399	997*	1800.03	121146	62069	1001*	1800.03
165	5	75	0.20	0.04	294983	115259	1000*	1801.37	532338	304707	1008*	1800.77	74145	39459	1004*	1800.03	121741	73278	1002*	1800.03
166	1	75	0.30	0.04	176318	37474	889*	1800.83	434697	261969	897*	1800.30	45451	20763	897*	1800.03	73458	50860	917*	1800.03
167	2	75	0.30	0.04	194918	76179	928*	1800.62	358617	211110	924*	1800.34	43190	18798	926*	1800.05	70890	45252	945*	1800.03
168	3	75	0.30	0.04	166193	65793	895*	1800.53	371075	247114	901*	1800.34	50625	21930	892*	1800.06	75959	42218	889*	1800.04
169	4	75	0.30	0.04	170006	77761	917*	1800.57	315336	211266	916*	1800.32	44245	17280	907*	1800.09	71450	47540	919*	1800.03
170	5	75	0.30	0.04	225912	6016	896*	1800.45	322911	135308	900*	1800.42	47116	14465	896*	1800.10	79971	47743	901*	1800.03
171	1	75	0.30	0.07	236164	180557	-	1800.42	1382101	1211497	-	1800.41	31800	21531	-	1800.03	188381	163606	-	1800.02
172	2	75	0.30	0.07	220159	167754	-	1800.24	1405939	1234901	-	1800.17	30014	20418	-	1800.02	167150	144835	-	1800.03
173	3	75	0.30	0.07	221109	166551	-	1800.25	1532912	1343437	-	1800.14	31192	21121	-	1800.02	195074	170261	-	1800.02
174	4	75	0.30	0.07	221367	167942	-	1800.55	1506354	1330533	-	1800.33	30288	20781	-	1800.02	261282	239807	-	1800.03
175	5	75	0.30	0.07	206469	154830	-	1800.61	1319662	1155109	-	1800.39	30966	21409	-	1800.03	186467	162053	-	1800.03
176	1	75	0.30	0.01	48	0	798	3.46	274	0	798	4.38	244	0	798	2.02	710	474	798	8.70
177	2	75	0.30	0.01	26494	0	821	28.38	32279	0	821	36.03	30565	1	821	59.00	22142	0	821	39.62
178	3	75	0.30	0.01	2178	0	816	6.69	5225	0	816	14.03	1118	193	816	7.62	863	406	816	12.26
179	4	75	0.30	0.01	177	46	820	5.07	1309	76	820	7.63	923	167	820	3.77	1258	21	820	4.72
180	5	75	0.30	0.01	255	37	815	2.17	114	0	815	4.67	130	7	815	2.55	1	0	815	1.81
Average			0.30	0.04	70905	28550	710	683.55	281860	206147	716	716.36	20390	7907	715	732.00	50787	35389	711	739.28

Table A.3. Test instances and comparison of formulations: random dataset.

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Rep	$V(G)$	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	1	20	0.70	0.30	900	0	1900	3.23	6793	0	1900	13.36	965	0	1900	10.67	975	153	1900	10.92
1	2	20	0.70	0.30	866	0	1900	3.55	6323	0	1900	11.78	1155	0	1900	10.92	956	175	1900	11.76
2	3	20	0.70	0.30	1027	0	1900	3.40	6878	0	1900	11.91	1033	0	1900	6.69	940	193	1900	7.27
3	1	20	0.70	0.50	1	1	1900	0.30	1	1	1900	1.99	1	1	1900	0.56	1	1	1900	2.34
4	2	20	0.70	0.50	1	1	1900	0.38	1	1	1900	1.33	1	1	1900	0.50	1	1	1900	1.48
5	3	20	0.70	0.50	1	1	1900	0.36	1	1	1900	2.28	1	1	1900	0.45	1	1	1900	1.48
6	1	20	0.70	0.70	0	0	1900	0.17	0	0	1900	0.20	0	0	1900	0.38	0	0	1900	0.14
7	2	20	0.70	0.70	0	0	1900	0.19	0	0	1900	0.17	0	0	1900	0.31	0	0	1900	0.16
8	3	20	0.70	0.70	0	0	1900	0.19	0	0	1900	0.16	0	0	1900	0.31	0	0	1900	0.11
9	1	20	0.70	0.10	23	0	750	0.81	55	0	750	3.75	37	0	750	1.76	34	0	750	2.02
10	2	20	0.70	0.10	309	0	777	1.31	316	0	777	5.35	251	0	777	2.06	898	0	777	2.31
11	3	20	0.70	0.10	1	2	769	0.83	139	0	769	2.31	1	1	769	2.19	277	0	769	1.25
12	1	20	0.40	0.50	0	0	1900	0.06	0	0	1900	0.16	0	0	1900	0.14	0	0	1900	0.11
13	2	20	0.40	0.50	0	0	1900	0.05	0	0	1900	0.13	0	0	1900	0.16	0	0	1900	0.09
14	3	20	0.40	0.50	0	0	1900	0.06	0	0	1900	0.11	0	0	1900	0.11	0	0	1900	0.08
15	1	20	0.40	0.70	0	0	1900	0.05	0	0	1900	0.16	0	0	1900	0.14	0	0	1900	0.13
16	2	20	0.40	0.70	0	0	1900	0.06	0	0	1900	0.13	0	0	1900	0.13	0	0	1900	0.09
17	3	20	0.40	0.70	0	0	1900	0.05	0	0	1900	0.13	0	0	1900	0.14	0	0	1900	0.09
18	1	20	0.40	0.10	1	1	1022	0.14	1	1	1022	1.61	1	0	1022	0.89	1	1	1022	0.83
19	2	20	0.40	0.10	1	1	964	0.14	1	1	964	1.08	1	1	964	0.41	1	1	964	0.68
20	3	20	0.40	0.10	135	0	1190	0.49	309	0	1190	1.69	141	0	1190	1.13	390	0	1190	1.08
21	1	20	0.40	0.30	0	0	1900	0.05	0	0	1900	0.16	0	0	1900	0.16	0	0	1900	0.11
22	2	20	0.40	0.30	0	0	1900	0.05	0	0	1900	0.14	0	0	1900	0.14	0	0	1900	0.11
23	3	20	0.40	0.30	1	1	1900	0.11	1	1	1900	0.69	1	1	1900	0.19	1	1	1900	0.53
24	1	20	0.50	0.30	1	1	1883	0.44	1	1	1883	3.36	16	0	1883	2.16	186	0	1883	2.23
25	2	20	0.50	0.30	1	1	1900	0.49	1	1	1900	1.28	13	0	1900	1.65	77	0	1900	1.81
26	3	20	0.50	0.30	1	1	1900	0.34	1	1	1900	3.42	11	0	1900	1.44	146	0	1900	1.89
27	1	20	0.50	0.50	0	0	1900	0.08	0	0	1900	0.19	0	0	1900	0.20	0	0	1900	0.13
28	2	20	0.50	0.50	0	0	1900	0.08	0	0	1900	0.13	0	0	1900	0.19	0	0	1900	0.09
29	3	20	0.50	0.50	0	0	1900	0.08	0	0	1900	0.14	0	0	1900	0.16	0	0	1900	0.11
30	1	20	0.50	0.70	0	0	1900	0.08	0	0	1900	0.14	0	0	1900	0.19	0	0	1900	0.11
31	2	20	0.50	0.70	0	0	1900	0.08	0	0	1900	0.13	0	0	1900	0.19	0	0	1900	0.09
32	3	20	0.50	0.70	0	0	1900	0.09	0	0	1900	0.14	0	0	1900	0.16	0	0	1900	0.11
33	1	20	0.50	0.10	1	2	893	0.48	1	2	893	3.70	32	0	893	1.75	32	0	893	1.94
34	2	20	0.50	0.10	1	1	879	0.41	52	0	879	3.49	109	0	879	1.17	261	0	879	1.05
35	3	20	0.50	0.10	1	1	924	0.20	1	2	924	1.61	1	0	924	1.22	1	0	924	1.19
36	1	30	0.40	0.70	0	0	2900	0.22	0	0	2900	0.19	0	0	2900	0.47	0	0	2900	0.17
37	2	30	0.40	0.70	0	0	2900	0.31	0	0	2900	0.17	0	0	2900	0.47	0	0	2900	0.14
38	3	30	0.40	0.70	0	0	2900	0.33	0	0	2900	0.20	0	0	2900	0.39	0	0	2900	0.16
39	1	30	0.40	0.10	264	0	1350	1.34	1250	0	1350	5.77	297	0	1350	2.86	722	8	1350	4.36
40	2	30	0.40	0.10	2726	0	1383	2.83	8393	0	1383	7.58	2024	0	1383	15.67	15467	0	1383	57.00
41	3	30	0.40	0.10	3763	0	1476	3.25	20008	0	1476	8.38	3604	0	1476	16.31	10169	0	1476	21.97
42	1	30	0.40	0.30	1	1	2900	0.27	576	0	2900	5.83	33	0	2900	8.22	1171	0	2900	5.41
43	2	30	0.40	0.30	37	0	2900	4.78	1445	0	2900	5.03	49	0	2900	11.22	940	0	2900	5.47
44	3	30	0.40	0.30	37	0	2900	4.63	1101	0	2900	5.04	53	0	2900	10.98	879	0	2900	5.75
45	1	30	0.40	0.50	0	0	2900	0.23	0	0	2900	0.19	0	0	2900	0.64	0	0	2900	0.13
46	2	30	0.40	0.50	0	0	2900	0.28	0	0	2900	0.10	0	0	2900	0.77	0	0	2900	0.16
47	3	30	0.40	0.50	0	0	2900	0.25	0	0	2900	0.10	0	0	2900	0.69	0	0	2900	0.14
48	1	30	0.50	0.50	1	1	2900	0.67	1	1	2900	9.86	1	1	2900	1.44	1	1	2900	3.38
49	2	30	0.50	0.50	1	1	2900	0.84	1	1	2900	3.48	1	1	2900	1.56	1	1	2900	3.45
50	3	30	0.50	0.50	1	1	2900	0.92	1	1	2900	3.50	1	1	2900	1.52	1	1	2900	3.34

Table A.3 . Test instances and comparison of formulations: random dataset. (cont.)

Instances			STRONG-multi				WEAK-multi				STRONG-single				WEAK-single			
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active
51	1	30	0.50	0.70	0	0	2900	0.41	0	0	2900	0.30	0	0	2900	1.00	0	0
52	2	30	0.50	0.70	0	0	2900	0.50	0	0	2900	0.11	0	0	2900	0.99	0	0
53	3	30	0.50	0.70	0	0	2900	0.52	0	0	2900	0.14	0	0	2900	0.97	0	0
54	1	30	0.50	0.10	1714	0	1274	2.50	4824	0	1274	13.54	1305	0	1274	16.53	6492	0
55	2	30	0.50	0.10	19628	0	1265	13.96	71853	0	1265	28.44	15972	0	1265	71.74	63629	0
56	3	30	0.50	0.10	1494	0	1222	2.14	3119	0	1222	2.94	1546	0	1222	9.83	5548	0
57	1	30	0.50	0.30	797	0	2900	21.08	4178	1503	2900	34.85	793	0	2900	31.88	1126	273
58	2	30	0.50	0.30	777	0	2900	18.72	2182	935	2900	16.59	783	0	2900	26.95	992	164
59	3	30	0.50	0.30	879	0	2900	15.31	2420	1046	2900	19.42	865	0	2900	32.49	1321	174
60	1	30	0.70	0.30	12269	0	2900	110.74	17037	0	2900	164.21	12902	0	2900	1172.66	47770	0
61	2	30	0.70	0.30	13047	0	2900	136.78	16523	0	2900	130.86	12658	0	2900	1125.50	45623	0
62	3	30	0.70	0.30	17457	0	2900	113.27	16409	0	2900	109.40	17047	0	2900	775.98	27466	0
63	1	30	0.70	0.50	91	0	2900	35.19	1471	0	2900	38.15	101	0	2900	33.67	1102	379
64	2	30	0.70	0.50	89	0	2900	30.22	3175	0	2900	24.96	89	0	2900	27.27	1496	851
65	3	30	0.70	0.50	91	0	2900	32.86	3711	0	2900	25.08	129	0	2900	34.66	1677	929
66	1	30	0.70	0.70	0	0	2900	1.19	0	0	2900	0.63	0	0	2900	1.24	0	0
67	2	30	0.70	0.70	0	0	2900	1.00	0	0	2900	0.42	0	0	2900	1.13	0	0
68	3	30	0.70	0.70	0	0	2900	0.94	0	0	2900	0.36	0	0	2900	1.09	0	0
69	1	30	0.70	0.10	5600	0	1078	6.42	28440	0	1078	60.94	4886	0	1078	29.80	19567	0
70	2	30	0.70	0.10	6611	0	1096	9.72	28355	0	1096	16.49	5715	0	1096	39.91	28399	0
71	3	30	0.70	0.10	2754	0	1131	4.89	3812	0	1131	3.94	1480	0	1131	12.62	18077	1877
72	1	40	0.40	0.30	789	0	3900	65.22	2564	996	3900	85.99	859	0	3900	80.58	1218	161
73	2	40	0.40	0.30	305	0	3900	41.91	2283	960	3900	55.09	235	0	3900	26.91	931	165
74	3	40	0.40	0.30	1009	0	3900	53.45	2610	863	3900	58.08	893	0	3900	82.92	2347	45
75	1	40	0.40	0.50	1	1	3900	1.91	1	1	3900	15.23	1	1	3900	1.91	1	1
76	2	40	0.40	0.50	0	0	3900	1.08	1	0	3900	3.16	0	0	3900	1.63	1	0
77	3	40	0.40	0.50	1	1	3900	2.09	1	1	3900	8.69	1	1	3900	2.19	1	1
78	1	40	0.40	0.70	0	0	3900	0.66	0	0	3900	0.33	0	0	3900	0.95	0	0
79	2	40	0.40	0.70	0	0	3900	0.86	0	0	3900	0.26	0	0	3900	1.06	0	0
80	3	40	0.40	0.70	0	0	3900	0.75	0	0	3900	0.34	0	0	3900	1.13	0	0
81	1	40	0.40	0.10	334235	0	1846	720.56	904655	486154	1940*	1801.57	267839	49047	1846*	1800.11	305352	186314
82	2	40	0.40	0.10	291507	0	1883	556.56	1790686	0	1883	1109.84	206136	40240	1892*	1800.17	249910	93837
83	3	40	0.40	0.10	191156	0	1753	415.23	494118	0	1753	492.71	140811	47219	1824*	1800.23	399211	162487
84	1	40	0.50	0.30	8342	0	3900	262.73	6725	0	3900	163.67	6069	0	3900	1142.86	9153	0
85	2	40	0.50	0.30	5686	0	3900	202.26	7048	0	3900	118.75	7637	0	3900	973.61	10642	0
86	3	40	0.50	0.30	9016	0	3900	244.47	6603	0	3900	146.23	7759	0	3900	1525.20	13821	0
87	1	40	0.50	0.50	1	1	3900	37.99	1509	0	3900	55.67	1	1	3900	40.41	1269	0
88	2	40	0.50	0.50	1	1	3900	27.50	1999	1225	3900	69.85	1	1	3900	27.40	1012	569
89	3	40	0.50	0.50	1	1	3900	38.07	2367	0	3900	52.90	23	0	3900	48.69	1074	704
90	1	40	0.50	0.70	0	0	3900	1.33	0	0	3900	0.58	0	0	3900	1.28	0	0
91	2	40	0.50	0.70	0	0	3900	1.19	0	0	3900	0.56	0	0	3900	1.36	0	0
92	3	40	0.50	0.70	0	0	3900	1.20	0	0	3900	0.48	0	0	3900	1.33	0	0
93	1	40	0.50	0.10	499169	0	1704*	1801.79	717752	355924	1703*	1800.35	135843	58158	1695*	1800.12	253053	155902
94	2	40	0.50	0.10	515709	0	1728*	1801.71	748857	337721	1728*	1801.51	131167	67713	1759*	1800.17	491044	303860
95	3	40	0.50	0.10	414042	0	1608	1108.08	694807	386134	1659*	1801.72	125694	68924	1696*	1800.18	277256	167986
96	1	40	0.70	0.70	1	1	3900	5.81	1	1	3900	23.77	1	1	3900	4.81	1	1
97	2	40	0.70	0.70	1	1	3900	4.70	1	1	3900	21.32	1	1	3900	5.03	1	1
98	3	40	0.70	0.70	1	1	3900	6.15	1	1	3900	40.92	1	1	3900	5.34	1	1
99	1	40	0.70	0.10	458043	0	1388*	1800.65	749349	418204	1408*	1800.26	92385	48737	1431*	1800.16	201268	146090
100	2	40	0.70	0.10	412278	0	1451*	1800.71	895109	565877	1510*	1800.22	79102	40695	1501*	1800.20	189216	135484

Table A.3 . Test instances and comparison of formulations: random dataset. (cont.)

Instances				STRONG-multi				WEAK-multi				STRONG-single				WEAK-single				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
101	3	40	0.70	0.10	210697	0	1397	878.54	925111	308754	1429*	1800.27	76752	34793	1452*	1800.19	189353	59213	1401*	1800.02
102	1	40	0.70	0.30	84140	28907	-	1800.50	1110952	772951	-	1801.08	9830	3502	-	1800.06	307341	224737	-	1800.02
103	2	40	0.70	0.30	37638	17590	3900*	1800.36	125797	24997	3900*	1800.32	8994	3039	-	1800.13	497115	28200	3900*	1800.02
104	3	40	0.70	0.30	46844	21907	3900*	1800.34	1371248	913877	-	1800.21	9485	3304	-	1800.14	288804	205234	-	1800.02
105	1	40	0.70	0.50	459	0	3900	212.78	3190	1342	3900	544.50	459	0	3900	323.92	3293	1536	3900	763.44
106	2	40	0.70	0.50	449	0	3900	281.23	2793	1268	3900	503.09	487	0	3900	331.72	3951	1954	3900	952.84
107	3	40	0.70	0.50	471	0	3900	261.49	3100	1193	3900	439.67	501	0	3900	348.92	3938	1980	3900	953.01
108	1	50	0.40	0.10	688774	585404	-	1800.24	2399745	1942943	-	1800.63	93461	78714	-	1800.11	608643	527107	-	1800.02
109	2	50	0.40	0.10	721676	595956	-	1800.25	3399324	2750250	-	1800.26	65824	46073	-	1800.11	772911	666439	-	1800.03
110	3	50	0.40	0.10	717054	580030	-	1800.24	3359534	2703666	-	1800.21	81141	66219	-	1800.15	702507	593921	-	1800.03
111	1	50	0.40	0.30	6609	0	4900	580.04	12127	0	4900	403.09	4669	1139	4900*	1800.06	11595	1602	4900*	1800.02
112	2	50	0.40	0.30	6614	0	4900	539.81	3885	809	4900	164.47	4231	748	4900*	1800.14	13321	0	4900	503.43
113	3	50	0.40	0.30	2695	0	4900	195.53	7757	0	4900	259.22	4661	792	4900*	1800.20	3300	0	4900	467.22
114	1	50	0.40	0.50	1	1	4900	23.83	603	0	4900	103.54	1	1	4900	21.25	727	434	4900	114.45
115	2	50	0.40	0.50	1	1	4900	20.22	1089	0	4900	86.71	1	1	4900	19.69	772	487	4900	111.28
116	3	50	0.40	0.50	1	1	4900	23.55	467	0	4900	105.74	1	1	4900	21.99	688	423	4900	118.25
117	1	50	0.40	0.70	0	0	4900	1.88	0	0	4900	0.89	0	0	4900	1.56	0	0	4900	0.91
118	2	50	0.40	0.70	0	0	4900	1.53	0	0	4900	0.83	0	0	4900	1.78	0	0	4900	0.58
119	3	50	0.40	0.70	0	0	4900	1.95	0	0	4900	0.84	0	0	4900	1.73	0	0	4900	0.80
120	1	50	0.50	0.70	0	0	4900	2.35	0	0	4900	2.31	0	0	4900	2.56	0	0	4900	2.92
121	2	50	0.50	0.70	0	0	4900	2.78	0	0	4900	1.73	0	0	4900	2.72	0	0	4900	1.70
122	3	50	0.50	0.70	0	0	4900	3.25	0	0	4900	2.03	0	0	4900	2.78	0	0	4900	2.86
123	1	50	0.50	0.10	539449	475361	-	1800.65	2356539	1981020	-	1800.31	86916	76367	-	1800.04	470016	410316	-	1800.02
124	2	50	0.50	0.10	502458	432364	-	1800.49	2114533	1735086	-	1800.18	61183	44094	-	1800.07	553434	491151	-	1800.02
125	3	50	0.50	0.10	500150	425247	-	1800.50	2470678	2089266	-	1800.21	71485	61349	-	1800.14	536900	471221	-	1800.02
126	1	50	0.50	0.30	29305	10791	4900*	1800.63	37010	0	4900	1435.61	3493	1741	4900*	1800.11	25886	12688	4900*	1800.02
127	2	50	0.50	0.30	36117	7038	-	1800.65	637189	0	4900	1570.53	3320	932	-	1800.08	103926	70065	-	1800.03
128	3	50	0.50	0.30	30743	11888	4900*	1800.70	83757	32232	4900*	1800.21	3311	1180	-	1800.15	29638	15098	4900*	1800.03
129	1	50	0.50	0.50	227	0	4900	317.25	3448	1504	4900	961.99	265	0	4900	377.56	3462	2317	4900*	1800.03
130	2	50	0.50	0.50	259	0	4900	361.74	2587	1427	4900	545.72	275	0	4900	381.37	4427	2969	4900*	1800.07
131	3	50	0.50	0.50	233	0	4900	341.84	2322	1422	4900	375.42	267	0	4900	390.94	3555	2002	4900	1670.34
132	1	50	0.70	0.30	21224	9322	-	1800.45	59465	36899	4900*	1801.15	2147	1169	-	1800.05	103122	91158	-	1800.02
133	2	50	0.70	0.30	28463	13313	-	1800.30	107533	64962	4900*	1800.94	1967	1450	-	1800.12	156822	139871	-	1800.02
134	3	50	0.70	0.30	25480	10721	-	1800.39	107306	59875	4900*	1801.53	2112	1537	-	1800.12	149311	133310	-	1800.03
135	1	50	0.70	0.50	5394	1525	-	1801.14	6514	91	4900	1727.15	968	755	-	1800.10	2962	2194	-	1800.07
136	2	50	0.70	0.50	3785	1169	-	1801.37	5774	529	4900*	1800.54	838	661	-	1800.41	3016	2047	-	1800.03
137	3	50	0.70	0.50	3779	1246	-	1801.27	9001	4	4900*	1800.52	893	698	-	1800.13	2936	2123	-	1800.03
138	1	50	0.70	0.70	1	1	4900	180.00	1	1	4900	87.51	1	1	4900	134.78	759	0	4900	386.77
139	2	50	0.70	0.70	1	1	4900	166.37	1	1	4900	83.48	1	1	4900	147.84	1021	0	4900	432.09
140	3	50	0.70	0.70	1	1	4900	176.93	575	0	4900	623.22	1	1	4900	141.11	771	0	4900	437.37
141	1	50	0.70	0.10	187990	164938	-	1800.80	1353267	1130968	2107*	1800.46	73217	65415	-	1800.04	247192	199086	2016*	1800.04
142	2	50	0.70	0.10	164524	107372	1932*	1800.97	1367237	1086210	2026*	1800.67	59340	39824	1958*	1800.15	196044	168189	2053*	1800.04
143	3	50	0.70	0.10	358871	315199	-	1800.53	425110	336098	2128*	1800.58	55712	37837	1962*	1800.15	338333	305732	-	1800.03
Average			0.53	0.40	56926	29587	2852	374.97	216436	142619	2942	406	14422	6903	2815	483.39	59771	43024	2842	503.55

APPENDIX B: DETAILED RESULTS OF BRANCH-AND-BOUND ALGORITHMS

This section introduces test instances for “Full Random Set” and detailed results of the solutions in plain setting, infeasibility test, and infeasibility test with probing with STRONG-multi formulation and BB-E. The tables include the following columns:

- ID: The ID of the MSTC instance.
- Type: Type of Zhang’s instances:
 - i. Type 1a, if known to be infeasible,
 - ii. Type 1b, if feasibility is unknown,
 - iii. Type 2, if known to be feasible.
- Rep: Replication index for instance generation.
- $|V(G)|$: Number of vertices in G .
- Γ : Graph density.
- Ω : Conflict density.
- Explored: Number of explored nodes until termination,
- Active: Number of active nodes during termination,
- Best UB: Best upper bound obtained until termination. Denoted with:
 - i. *, if a feasible solution not proven to be optimal is found until termination,
 - ii. -, if a feasible solution has not been found until termination.

Table B.1 . Plain branch-and-bound results: random dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	f^*	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
55	2	30	0.50	0.10	1266478	538014	1265	5.61	169327826	161407505	1299*	1800.04	135573008	134519503	1265	76.16	135572995	134519498	1265	98.96
56	3	30	0.50	0.10	152549	59155	1222	1.20	164592578	154089424	1261*	1800.04	12769654	12767281	1222	7.39	12769642	12767269	1222	9.25
57	1	30	0.50	0.30	136505	0	2900	1.82	101755	0	2900	6.39	8883013	0	2900	11.96	8883012	0	2900	13.74
58	2	30	0.50	0.30	115014	0	2900	1.64	116554	0	2900	9.31	10148450	0	2900	13.14	10148449	0	2900	15.48
59	3	30	0.50	0.30	154950	0	2900	1.81	134696	0	2900	8.51	11879293	0	2900	15.26	11879292	0	2900	18.24
60	1	30	0.70	0.30	2516692	0	2900	23.20	3273506	0	2900	222.73	390964637	0	2900	530.19	390964636	0	2900	674.98
61	2	30	0.70	0.30	2806080	0	2900	26.13	3073506	0	2900	203.21	338782558	0	2900	461.31	338782557	0	2900	587.05
62	3	30	0.70	0.30	2915618	0	2900	27.41	3245898	0	2900	221.15	336618756	0	2900	458.55	336618755	0	2900	564.88
63	1	30	0.70	0.50	16228	0	2900	0.33	4047	0	2900	0.75	320545	0	2900	0.80	320544	0	2900	0.92
64	2	30	0.70	0.50	15714	0	2900	0.31	3861	0	2900	0.74	297699	0	2900	0.76	297698	0	2900	0.86
65	3	30	0.70	0.50	17015	0	2900	0.34	4014	0	2900	0.76	274269	0	2900	0.74	274268	0	2900	0.84
66	1	30	0.70	0.70	1587	0	2900	0.04	127	0	2900	0.05	7982	0	2900	0.03	7981	0	2900	0.04
67	2	30	0.70	0.70	1645	0	2900	0.04	126	0	2900	0.05	7648	0	2900	0.03	7647	0	2900	0.04
68	3	30	0.70	0.70	1462	0	2900	0.04	118	0	2900	0.05	7463	0	2900	0.03	7462	0	2900	0.04
69	1	30	0.70	0.10	676381	188709	1078	3.71	137095356	134685153	1231*	1800.04	48769636	48620063	1078	31.13	48769606	48620036	1078	42.23
70	2	30	0.70	0.10	424441	183648	1096	3.16	137799038	135233645	1238*	1800.04	110964464	106067848	1096	71.66	110964432	106067817	1096	98.72
71	3	30	0.70	0.10	307410	79970	1131	2.55	132065663	129777417	1332*	1800.05	89534138	5347806	1131	64.93	89534121	5347789	1131	172.41
72	1	40	0.40	0.30	285794	0	3900	3.80	253711	0	3900	36.14	67828861	0	3900	115.56	67828860	0	3900	133.50
73	2	40	0.40	0.30	236108	0	3900	3.50	231097	0	3900	35.04	68832904	0	3900	116.64	68832903	0	3900	133.77
74	3	40	0.40	0.30	357264	0	3900	5.58	291817	0	3900	38.61	66063774	0	3900	113.68	66063773	0	3900	129.73
75	1	40	0.40	0.50	3936	0	3900	0.14	703	0	3900	0.22	113211	0	3900	0.40	113210	0	3900	0.41
76	2	40	0.40	0.50	3892	0	3900	0.15	694	0	3900	0.24	111421	0	3900	0.39	111420	0	3900	0.43
77	3	40	0.40	0.50	3295	0	3900	0.13	707	0	3900	0.29	109060	0	3900	0.39	109059	0	3900	0.41
78	1	40	0.40	0.70	615	0	3900	0.03	57	0	3900	0.04	5221	0	3900	0.02	5220	0	3900	0.02
79	2	40	0.40	0.70	579	0	3900	0.04	54	0	3900	0.04	5254	0	3900	0.03	5253	0	3900	0.03
80	3	40	0.40	0.70	533	0	3900	0.03	53	0	3900	0.04	5228	0	3900	0.02	5227	0	3900	0.03
81	1	40	0.40	0.10	267807062	127504579	-	1800	104727692	102909392	2331*	1800.05	2586256284	0	-	1800	1624086350	0	-	1800
82	2	40	0.40	0.10	261240840	103471292	1886*	1800.06	105539509	103432432	2153*	1800.07	2354433668	0	-	1800	1277389906	0	-	1800
83	3	40	0.40	0.10	208010876	87311027	1753	1431.53	114673419	112168109	2084*	1800.07	2581066505	0	-	1800	1327865029	0	-	1800
84	1	40	0.50	0.30	2696437	0	3900	35.25	2730393	0	3900	414.49	786854709	0	3900	1444.81	786854708	0	3900	1654.61
85	2	40	0.50	0.30	3095671	0	3900	40.92	2368611	0	3900	368.77	706800624	0	3900	1256.44	706800623	0	3900	1466.71
86	3	40	0.50	0.30	3179082	0	3900	48.05	2936087	0	3900	445.99	758618015	0	3900	1368.79	758618014	0	3900	1617.62
87	1	40	0.50	0.50	14491	0	3900	0.33	3152	0	3900	1.02	454621	0	3900	1.35	454620	0	3900	1.36
88	2	40	0.50	0.50	20970	0	3900	0.46	3192	0	3900	0.95	447693	0	3900	1.30	447692	0	3900	1.45
89	3	40	0.50	0.50	17985	0	3900	0.56	3245	0	3900	0.93	462775	0	3900	1.35	462774	0	3900	1.48
90	1	40	0.50	0.70	1732	0	3900	0.05	115	0	3900	0.05	10752	0	3900	0.05	10751	0	3900	0.05
91	2	40	0.50	0.70	1556	0	3900	0.04	132	0	3900	0.06	10815	0	3900	0.05	10814	0	3900	0.05
92	3	40	0.50	0.70	1911	0	3900	0.05	122	0	3900	0.05	10918	0	3900	0.05	10917	0	3900	0.06
93	1	40	0.50	0.10	255794748	127043776	-	1800	97138669	96164394	2195*	1800.06	2454073190	559381344	1855*	1800	1415204938	0	-	1800
94	2	40	0.50	0.10	243025288	120063315	-	1800.01	98462642	97435682	2091*	1800.06	2289415128	0	-	1800	1339995186	0	-	1800
95	3	40	0.50	0.10	260375276	129104442	-	1800.01	100664779	99572974	1974*	1800.05	1892531280	0	-	1800	1043888196	0	-	1800
96	1	40	0.70	0.70	5262	0	3900	0.11	305	0	3900	0.14	43939	0	3900	0.18	43938	0	3900	0.21
97	2	40	0.70	0.70	4888	0	3900	0.12	288	0	3900	0.16	43354	0	3900	0.18	43353	0	3900	0.23
98	3	40	0.70	0.70	5320	0	3900	0.11	311	0	3900	0.15	44471	0	3900	0.19	44470	0	3900	0.21
99	1	40	0.70	0.10	241279303	120610027	-	1800	80869760	80399627	1886*	1800.06	2309016947	668888374	1443*	1800	1037050118	0	-	1800
100	2	40	0.70	0.10	235376526	117666112	-	1800	79905005	79447156	2004*	1800.07	2144334663	1782032663	1441*	1800	1049798047	0	-	1800
101	3	40	0.70	0.10	215500954	97614336	1397	1625.93	83210733	82674085	1754*	1800.07	2044334663	1782032663	1441*	1800	117111412	908809873	1441*	1800
102	1	40	0.70	0.30	91724547	0	3900	1306.06	29708795	28082839	-	1800.01	903590726	0	-	1800	717605498	0	-	1800
103	2	40	0.70	0.30	113959055	0	3900	1657.18	29079074	27346780	-	1800.01	924762139	0	-	1800	717630837	0	-	1800
104	3	40	0.70	0.30	107113990	0	3900	1519.53	29971213	28284396	-	1800	878616797	0	-	1800	691370992	0	-	1800
105	1	40	0.70	0.50	134134	0	3900	2.60	28776	0	3900	8.31	4762669	0	3900	12.77	4762668	0	3900	15.26
106	2	40	0.70	0.50	144183	0	3900	2.70	26997	0	3900	8.14	4770563	0	3900	12.78	4770562	0	3900	14.64
107	3	40	0.70	0.50	128652	0	3900	2.50	26677	0	3900	7.68	4770438	0	3900	12.73	4770437	0	3900	15.50
108	1	50	0.40	0.10	204118562	101620217	-	1800	58342906	57938174	-	1800	1563811681	0	-	1800	913412352	0	-	1800
109	2	50	0.40	0.10	207089563	102690111	-	1800.01	63557824	63085488	-	1800	1773340039	0	-	1800	1094158968	0	-	1800

Table B.1 . Plain branch-and-bound results: random dataset. (cont.)

Instances				BB-E			BB-S			Rec-D			BB-D		
ID	Rep	V(G)	Γ	Ω	Explored	Active	Best UB	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
110	3	50	0.40	0.10	209575733	104203266	-	59772096	59340348	-	1800.01	1590338402	0	-	1800
111	1	50	0.40	0.30	4506222	0	4900	3618830	0	4900	1038.78	840198358	0	-	1800
112	2	50	0.40	0.30	2816720	0	4900	3303611	0	4900	988.07	796114125	0	-	1800
113	3	50	0.40	0.30	4915464	0	4900	3587434	0	4900	1051.71	800201016	0	-	1800
114	1	50	0.40	0.50	22963	0	4900	3372	0	4900	1.45	901332	0	4900	2.81
115	2	50	0.40	0.50	16585	0	4900	3067	0	4900	1.40	837472	0	4900	2.84
116	3	50	0.40	0.50	21485	0	4900	3404	0	4900	1.45	840333	0	4900	2.68
117	1	50	0.40	0.70	2340	0	4900	142	0	4900	0.06	16851	0	4900	0.08
118	2	50	0.40	0.70	1510	0	4900	136	0	4900	0.06	16691	0	4900	0.09
119	3	50	0.40	0.70	2241	0	4900	143	0	4900	0.07	16724	0	4900	0.09
120	1	50	0.50	0.70	5186	0	4900	255	0	4900	0.12	39514	0	4900	0.17
121	2	50	0.50	0.70	3945	0	4900	255	0	4900	0.12	38563	0	4900	0.20
122	3	50	0.50	0.70	6207	0	4900	254	0	4900	0.12	39087	0	4900	0.20
123	1	50	0.50	0.10	199059910	99491950	-	54354245	54106863	-	1800	1508228501	0	-	1800
124	2	50	0.50	0.10	192984506	96442895	-	55524334	55241736	-	1800.01	1415894410	0	-	1800
125	3	50	0.50	0.10	200918811	100402696	-	56510876	56245017	-	1800.01	1286848045	0	-	1800
126	1	50	0.50	0.30	53233745	0	4900	1046.32	16473042	-	1800	814549344	0	-	1800
127	2	50	0.50	0.30	44535586	0	4900	15805648	14759889	-	1800.01	739532701	0	-	1800
128	3	50	0.50	0.30	45768460	0	4900	16420496	15431856	4900*	1800.07	748481233	0	-	1800
129	1	50	0.50	0.50	103526	0	4900	14628	0	4900	6.85	4705732	0	4900	14.48
130	2	50	0.50	0.50	83742	0	4900	13428	0	4900	6.21	4309192	0	4900	13.90
131	3	50	0.50	0.50	93367	0	4900	13631	0	4900	6.59	4468111	0	4900	16.09
132	1	50	0.70	0.30	80856428	33007464	-	25276537	25035400	-	1800.01	668119961	0	-	1800
133	2	50	0.70	0.30	81562746	33161527	-	25121044	24875901	-	1800.01	641449842	0	-	1800
134	3	50	0.70	0.30	82054223	33266138	-	23691400	23468047	-	1800.01	633689792	0	-	1800
135	1	50	0.70	0.50	862915	0	4900	182460	0	4900	96.08	51100647	0	4900	185.50
136	2	50	0.70	0.50	830547	0	4900	177770	0	4900	101.52	50338793	0	4900	177.13
137	3	50	0.70	0.50	893686	0	4900	194576	0	4900	108.42	51180621	0	4900	186.86
138	1	50	0.70	0.70	13249	0	4900	787	0	4900	0.66	273101	0	4900	1.23
139	2	50	0.70	0.70	10707	0	4900	811	0	4900	0.67	263024	0	4900	1.22
140	3	50	0.70	0.70	13473	0	4900	839	0	4900	0.82	269668	0	4900	1.25
141	1	50	0.70	0.10	156027010	78013340	-	45882990	45750459	-	1800.01	1219358342	0	-	1800
142	2	50	0.70	0.10	149431206	74715375	-	44758248	44624071	2840*	1800.08	1168524783	0	-	1800
143	3	50	0.70	0.10	148077666	74038789	-	45244985	45113522	-	1800.01	1285889400	0	-	1800
144	1	75	0.40	0.30	44343500	15654491	-	11974106	11885442	-	1800.01	482327567	0	-	1800
145	2	75	0.40	0.30	44452750	15947438	-	12018628	11929644	-	1800.02	475788194	0	-	1800
146	3	75	0.40	0.30	44650109	16396841	-	11798626	11709280	7400*	1800.12	469128342	0	-	1800
147	1	75	0.40	0.50	589271	0	7400	68483	0	7400	105.24	71485350	0	7400	343.42
148	2	75	0.40	0.50	445643	0	7400	68015	0	7400	108.22	70952123	0	7400	339.31
149	3	75	0.40	0.50	609927	0	7400	70021	0	7400	107.27	71667885	0	7400	340.78
150	1	75	0.40	0.70	12210	0	7400	504	0	7400	0.60	276423	0	7400	1.73
151	2	75	0.40	0.70	15185	0	7400	542	0	7400	0.60	268883	0	7400	1.77
152	3	75	0.40	0.70	14109	0	7400	529	0	7400	0.60	266888	0	7400	1.67
153	1	75	0.40	0.10	107047259	53519996	-	22914671	22870405	-	1800.03	670202135	0	-	1800
154	2	75	0.40	0.10	104395973	52196363	-	21808601	21767070	-	1800.01	693925094	0	-	1800
155	3	75	0.40	0.10	105471623	52735038	-	23428693	23382984	-	1800.01	646943495	0	-	1800
156	1	75	0.50	0.30	41404351	19351663	-	12323163	12280127	-	1800.04	412059421	0	-	1800
157	2	75	0.50	0.30	41257772	19422054	-	12239468	12198391	-	1800.03	407292525	0	-	1800
158	3	75	0.50	0.30	43178586	20404987	-	12367025	12325103	-	1800.03	417805586	0	-	1800
159	1	75	0.50	0.50	2366862	0	7400	460541	0	7400	712.59	332827855	0	-	1800
160	2	75	0.50	0.50	2745506	0	7400	482658	0	7400	732.64	338253834	0	-	1800
161	3	75	0.50	0.50	2976777	0	7400	499874	0	7400	752.90	338492292	0	-	1800
162	1	75	0.50	0.70	32995	0	7400	1110	0	7400	2.48	1172110	0	7400	7.65
163	2	75	0.50	0.70	22098	0	7400	1102	0	7400	2.35	1187565	0	7400	7.65
164	3	75	0.50	0.70	29031	0	7400	1199	0	7400	2.42	1182182	0	7400	7.56

Table B.1 . Plain branch-and-bound results: random dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D						
ID	Rep	V	G	T	Ω	Explored	Active	Best UB	UB	CPU(s)	Explored	Active	Best UB	UB	CPU(s)	Explored	Active	Best UB	UB	CPU(s)		
165	1	75	0.50	0.10	100700299	50350138	-	1800	-	1800.02	20692414	20661902	-	1800.02	1800.02	625015561	0	-	1800.02	407855649	0	1800
166	2	75	0.50	0.10	99179860	49589930	-	1800	-	1800.02	20382244	20353668	-	1800.02	1800.02	539186502	0	-	1800.02	364646330	0	1800
167	3	75	0.50	0.10	102052324	51026164	-	1800	-	1800.02	21280318	21250631	-	1800.02	1800.02	586793887	0	-	1800.02	389521475	0	1800
168	1	75	0.70	0.70	152895	0	7400	8.88	7400	26.33	14490	0	7400	26.33	7332281	7332280	7332281	7400	55.23	7332280	7400	62.96
169	2	75	0.70	0.70	141027	0	7400	7.81	7400	24.63	14713	0	7400	24.63	7343960	7343959	7343960	7400	55.31	7343959	7400	63.69
170	3	75	0.70	0.70	130210	0	7400	7.16	7400	25.50	14418	0	7400	25.50	7315200	7315199	7315200	7400	55.46	7315199	7400	62.81
171	1	75	0.70	0.10	92528810	46264407	-	1800	-	1800.05	18261325	18244653	-	1800.05	1800.05	454327851	0	-	1800.04	269977686	0	1800
172	2	75	0.70	0.10	93107872	46553938	-	1800	-	1800.01	19155758	19138701	-	1800.01	1800.01	492978501	0	-	1800.04	310075579	0	1800
173	3	75	0.70	0.10	93278916	46639460	-	1800	-	1800.11	184466215	18449396	-	1800.11	1800.11	524478778	0	-	1800.03	339350693	0	1800
174	1	75	0.70	0.30	40054063	19952512	-	1800	-	1800.04	11073074	11054921	-	1800.04	1800.04	383401151	0	-	1800.03	298899094	0	1800
175	2	75	0.70	0.30	37848650	18862032	-	1800	-	1800.03	10807167	10789485	-	1800.03	1800.03	371803047	0	-	1800.03	285926805	0	1800
176	3	75	0.70	0.30	37438308	18655270	-	1800	-	1800.09	11176419	11157813	-	1800.09	1800.09	357778693	0	-	1800.06	278378987	0	1800
177	1	75	0.70	0.50	30090500	1841637	-	1800	-	1800.11	4732391	4667233	7400*	1800.11	1800.11	297029026	0	-	1800.03	250446410	0	1800
178	2	75	0.70	0.50	30179221	1449966	-	1800	-	1800.01	46706554	4640592	-	1800.01	1800.03	296269937	0	-	1800.03	252841627	0	1800
179	3	75	0.70	0.50	30064127	1574888	-	1800	-	1800.01	4726714	4659612	-	1800.01	1800.03	296253093	0	-	1800.03	252166283	0	1800
180	1	100	0.40	0.70	49905	0	9900	5.58	9900	9.89	1694	0	9900	9.89	3882523	3882522	3882523	9900	47.05	3882522	9900	40.25
181	2	100	0.40	0.70	49017	0	9900	5.10	9900	7.22	1641	0	9900	7.22	3845337	3845336	3845337	9900	35.25	3845336	9900	38.94
182	3	100	0.40	0.70	49708	0	9900	4.70	9900	7.14	1658	0	9900	7.14	3831943	3831942	3831943					

Table B.2. The effect of IT and Pr on Rec-D using random dataset.

Instances				IT				IT+RootPr				IT+StaticPr				IT+DynamicPr				
ID	Rep	V(G)	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	1	20	0.70	0.30	339919	0	1900	0.78	339919	0	1900	0.73	333201	0	1900	0.82	12050	0	1900	0.44
1	2	20	0.70	0.30	348724	0	1900	0.71	348724	0	1900	0.67	341102	0	1900	0.75	12943	0	1900	0.47
2	3	20	0.70	0.30	344984	0	1900	0.76	344984	0	1900	0.72	337826	0	1900	0.63	11981	0	1900	0.44
3	1	20	0.70	0.50	3206	0	1900	0.01	2918	0	1900	0.01	2908	0	1900	0.01	110	0	1900	0.01
4	2	20	0.70	0.50	3213	0	1900	0.01	2249	0	1900	0.01	2236	0	1900	0.01	103	0	1900	0.00
5	3	20	0.70	0.50	3218	0	1900	0.01	2649	0	1900	0.01	2639	0	1900	0.01	108	0	1900	0.01
6	1	20	0.70	0.70	395	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
7	2	20	0.70	0.70	438	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
8	3	20	0.70	0.70	359	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
9	1	20	0.70	0.10	13821	13537	750	0.02	13821	13537	750	0.06	13728	13457	750	0.06	13821	13537	750	0.02
10	2	20	0.70	0.10	13787	13776	777	0.02	13787	13776	777	0.03	13787	13776	777	0.03	13787	13776	777	0.02
11	3	20	0.70	0.10	4983	4979	769	0.01	4983	4979	769	0.01	4982	4978	769	0.01	4983	4979	769	0.01
12	1	20	0.40	0.50	101	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
13	2	20	0.40	0.50	153	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
14	3	20	0.40	0.50	181	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
15	1	20	0.40	0.70	69	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
16	2	20	0.40	0.70	69	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
17	3	20	0.40	0.70	69	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
18	1	20	0.40	0.10	1604	1577	1022	0.00	1604	1577	1022	0.01	1595	1571	1022	0.01	1604	1577	1022	0.00
19	2	20	0.40	0.10	673	652	964	0.00	673	652	964	0.00	673	652	964	0.00	673	652	964	0.00
20	3	20	0.40	0.10	7790	7499	1190	0.01	7790	7499	1190	0.02	7764	7495	1190	0.02	7790	7499	1190	0.01
21	1	20	0.40	0.30	660	0	1900	0.00	27	0	1900	0.00	17	0	1900	0.00	0	0	1900	0.00
22	2	20	0.40	0.30	1719	0	1900	0.00	490	0	1900	0.00	481	0	1900	0.00	45	0	1900	0.00
23	3	20	0.40	0.30	2152	0	1900	0.00	1651	0	1900	0.00	1643	0	1900	0.00	54	0	1900	0.00
24	1	20	0.50	0.30	14065	1	1883	0.03	13680	1	1883	0.03	13666	1	1883	0.03	242	1	1883	0.01
25	2	20	0.50	0.30	11198	0	1900	0.02	8961	0	1900	0.02	8946	0	1900	0.02	135	0	1900	0.01
26	3	20	0.50	0.30	13510	0	1900	0.03	10941	0	1900	0.02	10928	0	1900	0.02	193	0	1900	0.01
27	1	20	0.50	0.50	625	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
28	2	20	0.50	0.50	391	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
29	3	20	0.50	0.50	543	0	1900	0.00	1	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
30	1	20	0.50	0.70	87	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
31	2	20	0.50	0.70	89	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
32	3	20	0.50	0.70	88	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
33	1	20	0.50	0.10	11927	11084	893	0.02	11927	11084	893	0.05	11862	11031	893	0.05	11927	11084	893	0.02
34	2	20	0.50	0.10	3980	3868	879	0.00	3980	3868	879	0.01	3939	3837	879	0.01	3980	3868	879	0.00
35	3	20	0.50	0.10	2726	2643	924	0.00	2726	2643	924	0.01	2716	2634	924	0.01	2726	2643	924	0.00
36	1	30	0.40	0.70	184	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
37	2	30	0.40	0.70	161	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
38	3	30	0.40	0.70	195	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
39	1	30	0.40	0.10	733950	626878	1350	1.24	733950	626878	1350	1.01	696702	607582	1350	5.85	733950	626878	1350	0.96
40	2	30	0.40	0.10	8474986	7805915	1383	13.87	8474986	7805915	1383	12.04	8404059	7720307	1383	33.66	8474986	7805915	1383	13.25
41	3	30	0.40	0.10	8714260	7768922	1476	12.36	8714260	7768922	1476	14.65	8586474	7618213	1476	35.96	8714260	7768922	1476	13.41
42	1	30	0.40	0.30	121079	0	2900	0.32	109960	0	2900	0.30	109943	0	2900	0.28	787	0	2900	0.05
43	2	30	0.40	0.30	136744	0	2900	0.37	136744	0	2900	0.38	136730	0	2900	0.34	1450	0	2900	0.08
44	3	30	0.40	0.30	127411	0	2900	0.35	127411	0	2900	0.33	127398	0	2900	0.32	1091	0	2900	0.07
45	1	30	0.40	0.50	2257	0	2900	0.01	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
46	2	30	0.40	0.50	2430	0	2900	0.01	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
47	3	30	0.40	0.50	2690	0	2900	0.01	3	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
48	1	30	0.50	0.50	9160	0	2900	0.03	7195	0	2900	0.03	7181	0	2900	0.03	178	0	2900	0.02
49	2	30	0.50	0.50	8650	0	2900	0.03	7283	0	2900	0.03	7271	0	2900	0.03	182	0	2900	0.02
50	3	30	0.50	0.50	9127	0	2900	0.03	8264	0	2900	0.04	8247	0	2900	0.03	184	0	2900	0.02
51	1	30	0.50	0.70	823	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
52	2	30	0.50	0.70	791	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
53	3	30	0.50	0.70	737	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
54	1	30	0.50	0.10	5697957	5493513	1274	5.61	5697957	5493513	1274	5.39	5646162	5445839	1274	19.59	5697957	5493513	1274	6.88

Table B.2 . The effect of IT and Pr on Rec-D using random dataset. (cont.)

Instances				IT				IT+RootPr				IT+StaticPr				IT+DynamicPr			
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)
55	2	30	0.50	0.10	78901303	76913482	1265	76.24	69.66	77452780	75713128	1265	381.90	78901303	76913482	1265	68.13		
56	3	30	0.50	0.10	9336747	9030478	1222	7.57	7.28	9237569	8955580	1222	48.08	9336747	9030478	1222	7.74		
57	1	30	0.50	0.30	1461519	0	2900	2.64	2.52	1457707	0	2900	2.94	14170	0	2900	1.14		
58	2	30	0.50	0.30	1826386	0	2900	2.96	3.01	1819417	0	2900	3.46	18948	0	2900	1.28		
59	3	30	0.50	0.30	2346758	0	2900	4.15	4.69	2336925	0	2900	3.99	238192	0	2900	1.50		
60	1	30	0.70	0.30	80270963	0	2900	141.19	142.90	64114732	0	2900	189.00	1319785	0	2900	72.01		
61	2	30	0.70	0.30	73121065	0	2900	127.67	175.51	58886927	0	2900	170.90	1293990	0	2900	69.56		
62	3	30	0.70	0.30	75940418	0	2900	131.77	170.93	60701112	0	2900	182.03	1397220	0	2900	75.97		
63	1	30	0.70	0.50	112336	0	2900	0.34	0.37	110606	0	2900	0.35	462	0	2900	0.11		
64	2	30	0.70	0.50	99064	0	2900	0.33	0.31	96556	0	2900	0.31	314	0	2900	0.11		
65	3	30	0.70	0.50	108406	0	2900	0.35	0.34	107243	0	2900	0.35	384	0	2900	0.11		
66	1	30	0.70	0.70	6232	0	2900	0.03	0.01	2055	0	2900	0.01	203	0	2900	0.01		
67	2	30	0.70	0.70	5361	0	2900	0.02	0.00	115	0	2900	0.00	115	0	2900	0.00		
68	3	30	0.70	0.70	5676	0	2900	0.02	0.00	245	0	2900	0.00	159	0	2900	0.00		
69	1	30	0.70	0.10	30009468	29238960	1078	31.26	32.35	28243422	27925051	1078	330.68	30009468	29238960	1078	31.59		
70	2	30	0.70	0.10	82465523	81542989	1096	82.48	85.42	80427406	79837690	1096	622.09	82465523	81542989	1096	80.07		
71	3	30	0.70	0.10	4141632	4098498	1131	4.50	4.47	4053401	4035469	1131	16.10	4141632	4098498	1131	4.31		
72	1	40	0.40	0.30	7034131	0	3900	14.31	14.26	7023252	0	3900	14.31	32174	0	3900	3.23		
73	2	40	0.40	0.30	5889733	0	3900	12.04	12.46	5883593	0	3900	12.93	22399	0	3900	2.99		
74	3	40	0.40	0.30	7908785	0	3900	15.73	16.07	7895963	0	3900	16.37	35975	0	3900	4.32		
75	1	40	0.40	0.50	18326	0	3900	0.08	0.06	13557	0	3900	0.06	253	0	3900	0.05		
76	2	40	0.40	0.50	16466	0	3900	0.07	0.05	10237	0	3900	0.05	234	0	3900	0.02		
77	3	40	0.40	0.50	19191	0	3900	0.08	0.07	16671	0	3900	0.07	265	0	3900	0.05		
78	1	40	0.40	0.70	1099	0	3900	0.01	0.00	0	0	3900	0.00	0	0	3900	0.00		
79	2	40	0.40	0.70	1088	0	3900	0.01	0.00	0	0	3900	0.00	0	0	3900	0.00		
80	3	40	0.40	0.70	1502	0	3900	0.01	0.00	0	0	3900	0.00	0	0	3900	0.00		
81	1	40	0.40	0.10	1057593806	885046982	1857*	1800	1800	67022490	49013353	1857*	1800	1095690425	919301688	1848*	1800		
82	2	40	0.40	0.10	922181915	89434059	2172*	1800	1800	371426232	0	1800	1800	946390106	103289980	2172*	1800		
83	3	40	0.40	0.10	1076485982	883402994	1813*	1800	1800	37174579	12182026	1813*	1800	1128431482	928236257	1813*	1800		
84	1	40	0.50	0.30	97284024	0	3900	212.64	211.93	91639064	0	3900	233.03	966442	0	3900	68.76		
85	2	40	0.50	0.30	86544687	0	3900	187.04	195.83	82093767	0	3900	210.19	968273	0	3900	67.24		
86	3	40	0.50	0.30	106201803	0	3900	230.22	242.64	99619074	0	3900	264.44	1141382	0	3900	83.59		
87	1	40	0.50	0.50	103410	0	3900	0.39	0.37	98965	0	3900	0.37	350	0	3900	0.14		
88	2	40	0.50	0.50	115835	0	3900	0.50	0.40	109480	0	3900	0.41	348	0	3900	0.15		
89	3	40	0.50	0.50	121214	0	3900	0.45	0.44	118985	0	3900	0.44	351	0	3900	0.14		
90	1	40	0.50	0.70	5868	0	3900	0.02	0.00	0	0	3900	0.00	0	0	3900	0.00		
91	2	40	0.50	0.70	6543	0	3900	0.03	0.00	0	0	3900	0.00	0	0	3900	0.00		
92	3	40	0.50	0.70	6766	0	3900	0.03	0.00	0	0	3900	0.00	0	0	3900	0.00		
93	1	40	0.50	0.10	971758909	824573520	1809*	1800	1800	29355894	16971321	1855*	1800	1003172988	852111533	1809*	1800		
94	2	40	0.50	0.10	837459214	351356140	2025*	1800	1800	51672915	15744940	2054*	1800	830855120	348477299	2025*	1800		
95	3	40	0.50	0.10	770016135	0	3900	1800	1800	28403655	0	3900	1800	771214954	0	3900	1800		
96	1	40	0.70	0.70	25533	0	3900	0.11	0.11	21551	0	3900	0.10	468	0	3900	0.07		
97	2	40	0.70	0.70	25201	0	3900	0.11	0.11	22513	0	3900	0.11	477	0	3900	0.07		
98	3	40	0.70	0.70	25574	0	3900	0.12	0.10	21509	0	3900	0.10	465	0	3900	0.07		
99	1	40	0.70	0.10	1106163865	1061265446	1443*	1800	1800	22339637	20336544	1599*	1800	1117588453	1072118168	1443*	1800		
100	2	40	0.70	0.10	1046118447	989462420	1507*	1800	1800	25500261	2530370	1596*	1800	1062845937	1005755311	1507*	1800		
101	3	40	0.70	0.10	1138762299	1097167984	1441*	1800	1800	26209950	26080438	1519*	1800	1161141939	1119153384	1441*	1800		
102	1	40	0.70	0.30	763307455	0	3900	1800	1800	147695273	0	3900	1800	24214044	0	3900	1800		
103	2	40	0.70	0.30	765840461	0	3900	1800	1800	165713516	0	3900	1800	23638416	0	3900	1800		
104	3	40	0.70	0.30	764037860	0	3900	1800	1800	155258642	0	3900	1800	21842401	0	3900	1800		
105	1	40	0.70	0.50	1247413	0	3900	3.84	3.88	1171327	0	3900	3.89	26467	0	3900	1.63		
106	2	40	0.70	0.50	1223321	0	3900	3.79	3.75	1159140	0	3900	4.10	23854	0	3900	1.56		
107	3	40	0.70	0.50	1214689	0	3900	3.75	3.74	1143861	0	3900	3.76	25348	0	3900	1.45		
108	1	50	0.40	0.10	727297800	0	1800	1800	1800	47574896	0	1800	1800	710470312	0	1800	1800		
109	2	50	0.40	0.10	729452453	0	1800	1800	1800	85270444	0	1800	1800	735380636	0	1800	1800		

Table B.2 . The effect of IT and Pr on Rec-D using random dataset. (cont.)

Instances				IT			IT+RootPr			IT+StaticPr			IT+DynamicPr						
ID	Rep	V(G)	f	Ω	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)
110	3	50	0.40	0.10	624661208	0	-	-	1800	601995430	0	-	-	1800	624438718	0	-	-	1800
111	1	50	0.40	0.30	170853669	0	4900	4900	433.06	170853669	0	4900	433.34	4900	1215974	0	4900	108.00	
112	2	50	0.40	0.30	151960975	0	4900	4900	395.84	151960975	0	4900	410.37	4900	806420	0	4900	74.10	
113	3	50	0.40	0.30	168221510	0	4900	4900	438.98	168221510	0	4900	455.99	4900	1155219	0	4900	104.80	
114	1	50	0.40	0.50	160878	0	4900	4900	0.64	155919	0	4900	0.62	4900	439	0	4900	0.19	
115	2	50	0.40	0.50	124291	0	4900	4900	0.49	108870	0	4900	0.46	4900	436	0	4900	0.18	
116	3	50	0.40	0.50	149357	0	4900	4900	0.58	143228	0	4900	0.57	4900	438	0	4900	0.21	
117	1	50	0.40	0.70	9626	0	4900	4900	0.05	0	0	4900	0.00	4900	0	0	4900	0.00	
118	2	50	0.40	0.70	6720	0	4900	4900	0.03	0	0	4900	0.00	4900	0	0	4900	0.00	
119	3	50	0.40	0.70	8115	0	4900	4900	0.04	0	0	4900	0.00	4900	0	0	4900	0.00	
120	1	50	0.50	0.70	26459	0	4900	4900	0.13	2421	0	4900	0.01	4900	344	0	4900	0.01	
121	2	50	0.50	0.70	24372	0	4900	4900	0.12	415	0	4900	0.01	4900	276	0	4900	0.01	
122	3	50	0.50	0.70	26617	0	4900	4900	0.13	4578	0	4900	0.03	4900	387	0	4900	0.02	
123	1	50	0.50	0.10	655556379	0	-	-	1800	654130021	0	-	-	1800	633252803	0	-	1800	
124	2	50	0.50	0.10	633198085	0	-	-	1800	610102175	0	-	-	1800	637967770	0	-	1800	
125	3	50	0.50	0.10	664917370	0	-	-	1800	632741855	0	-	-	1800	667509776	0	-	1800	
126	1	50	0.50	0.30	677964167	0	-	-	1800	674513222	0	-	-	1800	12217386	0	-	1800	
127	2	50	0.50	0.30	652038474	0	-	-	1800	631075074	0	-	-	1800	10360764	0	-	1800	
128	3	50	0.50	0.30	651127708	0	-	-	1800	621415721	0	-	-	1800	10772083	0	-	1800	
129	1	50	0.50	0.50	1090647	0	4900	4900	3.67	1090647	0	4900	3.52	4900	4.07	0	4900	0.54	
130	2	50	0.50	0.50	925550	0	4900	4900	3.21	925550	0	4900	3.54	4900	3.57	0	4900	0.50	
131	3	50	0.50	0.50	1008195	0	4900	4900	3.51	1008195	0	4900	3.52	4900	3.51	0	4900	0.54	
132	1	50	0.70	0.30	615074749	0	-	-	1800.01	611741120	0	-	-	1800.01	11975012	0	-	1800	
133	2	50	0.70	0.30	596085302	0	-	-	1800.01	571756219	0	-	-	1800.01	11400388	0	-	1800	
134	3	50	0.70	0.30	592143247	0	-	-	1800.01	569462339	0	-	-	1800.01	10811344	0	-	1800	
135	1	50	0.70	0.50	9460401	0	4900	4900	37.27	9460401	0	4900	37.50	4900	135896	0	4900	30.17	
136	2	50	0.70	0.50	9200147	0	4900	4900	38.49	9200147	0	4900	39.06	4900	135444	0	4900	33.89	
137	3	50	0.70	0.50	9766807	0	4900	4900	40.13	9766807	0	4900	40.05	4900	137124	0	4900	43.24	
138	1	50	0.70	0.70	72812	0	4900	4900	0.45	71908	0	4900	0.45	4900	800	0	4900	0.37	
139	2	50	0.70	0.70	73112	0	4900	4900	0.42	72515	0	4900	0.46	4900	803	0	4900	0.38	
140	3	50	0.70	0.70	73610	0	4900	4900	0.46	73325	0	4900	0.46	4900	807	0	4900	0.36	
141	1	50	0.70	0.10	540910541	280827233	2115*	2115*	1800	540573561	280627758	2115*	2115*	1800	482758851	246523749	2115*	1800	
142	2	50	0.70	0.10	460798892	0	-	-	1800.01	444675540	0	-	-	1800.01	459863827	0	-	1800	
143	3	50	0.70	0.10	586636699	414650125	2035*	2035*	1800	565219011	396740513	2035*	2035*	1800	588547680	415905373	2035*	1800	
144	1	75	0.40	0.30	460477435	0	-	-	1800.01	456924919	0	-	-	1800.01	8755469	0	-	1800	
145	2	75	0.40	0.30	453082190	0	-	-	1800.01	458409675	0	-	-	1800.01	8360009	0	-	1800	
146	3	75	0.40	0.30	45890691	0	-	-	1800.01	455753421	0	-	-	1800.01	8474373	0	-	1800	
147	1	75	0.40	0.50	8806492	0	7400	7400	47.26	8806492	0	7400	46.33	7400	22991	0	7400	3.87	
148	2	75	0.40	0.50	8852514	0	7400	7400	47.84	8852514	0	7400	46.36	7400	25373	0	7400	4.21	
149	3	75	0.40	0.50	8891221	0	7400	7400	47.13	8891221	0	7400	49.27	7400	28554	0	7400	5.06	
150	1	75	0.40	0.70	104918	0	7400	7400	0.78	78721	0	7400	0.57	7400	920	0	7400	0.32	
151	2	75	0.40	0.70	106653	0	7400	7400	0.79	78982	0	7400	0.56	7400	927	0	7400	0.32	
152	3	75	0.40	0.70	104737	0	7400	7400	0.78	80483	0	7400	0.57	7400	927	0	7400	0.32	
153	1	75	0.40	0.10	554839516	0	-	-	1800.01	553207915	0	-	-	1800.01	536369541	0	-	1800	
154	2	75	0.40	0.10	513975808	0	-	-	1800.01	514030210	0	-	-	1800.01	457323086	0	-	1800	
155	3	75	0.40	0.10	533798944	0	-	-	1800.01	535200127	0	-	-	1800.01	502855827	0	-	1800	
156	1	75	0.50	0.30	421465210	0	-	-	1800.02	424523086	0	-	-	1800.01	3372954	0	-	1800	
157	2	75	0.50	0.30	427441341	0	-	-	1800.01	422145984	0	-	-	1800.01	3487117	0	-	1800	
158	3	75	0.50	0.30	335109034	0	-	-	1800.02	423022956	0	-	-	1800.02	4154632	0	-	1800	
159	1	75	0.50	0.50	43520423	0	7400	7400	227.58	38520423	0	7400	237.03	7400	251.91	0	7400	180.49	
160	2	75	0.50	0.50	41745906	0	7400	7400	246.82	41745906	0	7400	246.82	7400	353265	0	7400	196.25	
161	3	75	0.50	0.50	43018118	0	7400	7400	255.56	43018118	0	7400	258.94	7400	364382	0	7400	201.79	
162	1	75	0.50	0.70	194964	0	7400	7400	1.67	194964	0	7400	1.71	7400	1302	0	7400	1.56	
163	2	75	0.50	0.70	196369	0	7400	7400	1.66	194188	0	7400	1.74	7400	1306	0	7400	1.44	
164	3	75	0.50	0.70	196115	0	7400	7400	1.67	194104	0	7400	1.74	7400	1306	0	7400	1.44	

Table B.2 . The effect of IT and Pr on Rec-D using random dataset. (cont.)

Instances				IT				IT+RootPr				IT+StaticPr				IT+DynamicPr			
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)	Explored	Active	Best	UB	CPU(s)
165	1	75	0.50	0.10	5188760558	0	-	-	1800.02	499682465	0	-	-	1800.02	11993844	0	-	-	1800.02
166	2	75	0.50	0.10	525026420	0	-	-	1800.02	507620378	0	-	-	1800.02	13411589	0	-	-	1800.02
167	3	75	0.50	0.10	508105573	0	-	-	1800.02	472783693	0	-	-	1800.02	12797459	0	-	-	1800.02
168	1	75	0.70	0.70	771216	0	7400	7400	6.90	771216	0	7400	7400	6.87	434845	0	7400	7400	5.37
169	2	75	0.70	0.70	783368	0	7400	7400	6.93	783368	0	7400	7400	6.92	434903	0	7400	7400	5.34
170	3	75	0.70	0.70	764091	0	7400	7400	6.77	764091	0	7400	7400	6.73	434747	0	7400	7400	5.34
171	1	75	0.70	0.10	477214173	0	-	-	1800.04	476338800	0	-	-	1800.04	10949112	0	-	-	1800.05
172	2	75	0.70	0.10	41989765	0	-	-	1800.04	419372910	0	-	-	1800.04	10435940	0	-	-	1800.04
173	3	75	0.70	0.10	454520961	0	-	-	1800.04	453271023	0	-	-	1800.04	13573120	0	-	-	1800.04
174	1	75	0.70	0.30	421834347	0	-	-	1800.03	4117297112	0	-	-	1800.03	3831810	0	-	-	1800.03
175	2	75	0.70	0.30	412675625	0	-	-	1800.03	411825149	0	-	-	1800.03	3561262	0	-	-	1800.03
176	3	75	0.70	0.30	414551627	0	-	-	1800.04	414730450	0	-	-	1800.03	3581405	0	-	-	1800.03
177	1	75	0.70	0.50	339157383	0	-	-	1800.03	332812431	0	-	-	1800.03	14246626	0	-	-	1800
178	2	75	0.70	0.50	340341013	0	-	-	1800.03	311281720	0	-	-	1800.03	13435311	0	-	-	1788.28
179	3	75	0.70	0.50	342037753	0	-	-	1800.03	291804080	0	-	-	1800.06	12718718	0	-	-	1734.19
180	1	100	0.40	0.70	407595	0	9900	9900	9.30	401732	0	9900	9900	8.43	401523	0	9900	9900	5.86
181	2	100	0.40	0.70	408079	0	9900	9900	9.60	404380	0	9900	9900	6.43	404106	0	9900	9900	6.42
182	3	100	0.40	0.70	407565	0	9900	9900	7.71	402368	0	9900	9900	5.78	404106	0	9900	9900	5.46
183	1	100	0.40	0.10	347011119	0	-	-	1800.04	323843277	0	-	-	1800.04	7705901	0	-	-	1800.04
184	2	100	0.40	0.10	366033111	0	-	-	1800.05	347301627	0	-	-	1800.05	7695052	0	-	-	1800.04
185	3	100	0.40	0.10	380461413	0	-	-	1800.05	363371966	0	-	-	1800.09	7500903	0	-	-	1800.04
186	1	100	0.40	0.30	303788797	0	-	-	1800.04	290055130	0	-	-	1800.05	8170722	0	-	-	1800.04
187	2	100	0.40	0.30	305991354	0	-	-	1800.03	296176773	0	-	-	1800.04	11337324	0	-	-	1800.03
188	3	100	0.40	0.30	303485989	0	-	-	1800.04	305221168	0	-	-	1800.04	10453672	0	-	-	1800.03
189	1	100	0.40	0.50	138425690	0	9900	9900	12.16	138425690	0	9900	9900	12.02	81679268	0	9900	9900	829.54
190	2	100	0.40	0.50	135062898	0	9900	9900	11.75	135062898	0	9900	9900	11.54	81821842	0	9900	9900	818.28
191	3	100	0.40	0.50	137086926	0	9900	9900	11.89	137086926	0	9900	9900	11.84	81812942	0	9900	9900	827.98
192	1	100	0.50	0.50	224914090	0	-	-	1800.06	224708978	0	-	-	1800.06	15050518	0	-	-	1800.04
193	2	100	0.50	0.50	227869270	0	-	-	1800.06	216719865	0	-	-	1800.06	14141005	0	-	-	1800.03
194	3	100	0.50	0.50	225621382	0	-	-	1800.06	21792061	0	-	-	1800.05	14850051	0	-	-	1800.05
195	1	100	0.50	0.70	792476	0	9900	9900	12.03	792476	0	9900	9900	12.33	693305	0	9900	9900	9.53
196	2	100	0.50	0.70	785343	0	9900	9900	12.77	783373	0	9900	9900	13.10	692423	0	9900	9900	10.39
197	3	100	0.50	0.70	794150	0	9900	9900	12.28	794150	0	9900	9900	12.74	693358	0	9900	9900	13.01
198	1	100	0.50	0.10	315507311	0	-	-	1800.10	296230799	0	-	-	1800.07	6059499	0	-	-	1800.06
199	2	100	0.50	0.10	341004603	0	-	-	1800.07	331783733	0	-	-	1800.07	6240921	0	-	-	1800.06
200	3	100	0.50	0.10	321022575	0	-	-	1800.07	306563891	0	-	-	1800.07	6464944	0	-	-	1800.06
201	1	100	0.50	0.30	289242689	0	-	-	1800.06	276013901	0	-	-	1800.06	2695328	0	-	-	1800.05
202	2	100	0.50	0.30	291588283	0	-	-	1800.06	281263885	0	-	-	1800.06	2972305	0	-	-	1800.05
203	3	100	0.50	0.30	286246856	0	-	-	1800.06	275558800	0	-	-	1800.06	3279430	0	-	-	1800.08
204	1	100	0.70	0.30	249869508	0	-	-	1800.14	241208238	0	-	-	1800.13	3014928	0	-	-	1800.12
205	2	100	0.70	0.30	246592604	0	-	-	1800.14	238468256	0	-	-	1800.14	2093006	0	-	-	1800.13
206	3	100	0.70	0.30	257827057	0	-	-	1800.14	238406998	0	-	-	1800.14	2378538	0	-	-	1800.13
207	1	100	0.70	0.50	185378296	0	-	-	1800.13	175190913	0	-	-	1800.13	1830709	0	-	-	1800.13
208	2	100	0.70	0.50	186106560	0	-	-	1800.12	183225558	0	-	-	1800.12	1996757	0	-	-	1800.12
209	3	100	0.70	0.50	184688695	0	-	-	1800.12	182450232	0	-	-	1800.13	1694680	0	-	-	1800.14
210	1	100	0.70	0.70	25567557	0	9900	9900	289.34	25567557	0	9900	9900	295.35	1480249	0	9900	9900	39.69
211	2	100	0.70	0.70	26248445	0	9900	9900	295.29	26248445	0	9900	9900	297.89	1480101	0	9900	9900	40.60
212	3	100	0.70	0.70	26271344	0	9900	9900	302.73	26271344	0	9900	9900	301.81	1480185	0	9900	9900	40.55
213	1	100	0.70	0.10	275962772	0	-	-	1800.14	276007429	0	-	-	1800.14	5525868	0	-	-	1800.14
214	2	100	0.70	0.10	285536899	0	-	-	1800.22	284573946	0	-	-	1800.14	5814806	0	-	-	1800.14
215	3	100	0.70	0.10	296347436	0	-	-	1800.14	295855021	0	-	-	1800.14	6142333	0	-	-	1800.14
Average				0.53	0.40	32869288	3854	637.81	638.38	175205921	32210034	3854	638.38	644.86	102489418	33488204	3900	619.76	

Table B.3. The effect of IT and Pr on BB-D using random dataset.

Instances				IT				IT+RootPr				IT+StaticPr				IT+DynamicPr				
ID	Rep	V(G)	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	1	20	0.70	0.30	22498	0	1900	0.35	22498	0	1900	0.34	19288	0	1900	0.37	1010	0	1900	0.10
1	2	20	0.70	0.30	23554	0	1900	0.37	23554	0	1900	0.36	20013	0	1900	0.36	965	0	1900	0.19
2	3	20	0.70	0.30	22157	0	1900	0.35	22157	0	1900	0.34	19305	0	1900	0.33	888	0	1900	0.17
3	1	20	0.70	0.50	303	0	1900	0.01	262	0	1900	0.01	251	0	1900	0.01	21	0	1900	0.01
4	2	20	0.70	0.50	308	0	1900	0.01	217	0	1900	0.01	205	0	1900	0.01	23	0	1900	0.00
5	3	20	0.70	0.50	282	0	1900	0.01	221	0	1900	0.01	213	0	1900	0.01	24	0	1900	0.00
6	1	20	0.70	0.70	45	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
7	2	20	0.70	0.70	43	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
8	3	20	0.70	0.70	37	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
9	1	20	0.70	0.10	13540	13527	750	0.02	13540	13527	750	0.07	13460	13447	750	0.07	13540	13527	750	0.01
10	2	20	0.70	0.10	13775	13775	777	0.02	13775	13775	777	0.03	13775	13775	777	0.03	13775	13775	777	0.02
11	3	20	0.70	0.10	4978	4976	769	0.01	4978	4976	769	0.01	4977	4975	769	0.01	4978	4976	769	0.01
12	1	20	0.40	0.50	15	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
13	2	20	0.40	0.50	21	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
14	3	20	0.40	0.50	23	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
15	1	20	0.40	0.70	11	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
16	2	20	0.40	0.70	11	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
17	3	20	0.40	0.70	11	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
18	1	20	0.40	0.10	1582	1568	1022	0.00	1582	1568	1022	0.01	1576	1562	1022	0.01	1582	1568	1022	0.00
19	2	20	0.40	0.10	653	649	964	0.00	653	649	964	0.00	653	649	964	0.00	653	649	964	0.00
20	3	20	0.40	0.10	7503	7496	1190	0.01	7503	7496	1190	0.02	7499	7492	1190	0.02	7503	7496	1190	0.01
21	1	20	0.40	0.30	84	0	1900	0.00	11	0	1900	0.00	2	0	1900	0.00	1	0	1900	0.00
22	2	20	0.40	0.30	205	0	1900	0.00	94	0	1900	0.00	83	0	1900	0.00	4	0	1900	0.00
23	3	20	0.40	0.30	208	0	1900	0.00	176	0	1900	0.00	171	0	1900	0.00	13	0	1900	0.00
24	1	20	0.50	0.30	1167	1	1883	0.02	1124	1	1883	0.02	1102	1	1883	0.02	41	1	1883	0.01
25	2	20	0.50	0.30	976	0	1900	0.02	821	0	1900	0.01	811	0	1900	0.01	42	0	1900	0.01
26	3	20	0.50	0.30	1151	0	1900	0.02	923	0	1900	0.02	906	0	1900	0.02	36	0	1900	0.01
27	1	20	0.50	0.50	51	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
28	2	20	0.50	0.50	39	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
29	3	20	0.50	0.50	52	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
30	1	20	0.50	0.70	10	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
31	2	20	0.50	0.70	12	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
32	3	20	0.50	0.70	11	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00	0	0	1900	0.00
33	1	20	0.50	0.10	11470	11074	893	0.02	11470	11074	893	0.05	11415	11018	893	0.05	11470	11074	893	0.02
34	2	20	0.50	0.10	3867	3865	879	0.01	3867	3865	879	0.01	3836	3834	879	0.01	3867	3865	879	0.01
35	3	20	0.50	0.10	2642	2634	924	0.01	2642	2634	924	0.01	2633	2625	924	0.01	2642	2634	924	0.01
36	1	30	0.40	0.70	20	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
37	2	30	0.40	0.70	17	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
38	3	30	0.40	0.70	21	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
39	1	30	0.40	0.10	641835	626843	1350	1.38	641835	626843	1350	1.09	615949	606591	1350	4.08	641835	626843	1350	1.23
40	2	30	0.40	0.10	7846752	7805018	1383	15.54	7846752	7805018	1383	13.28	7754152	7715911	1383	37.03	7846752	7805018	1383	14.02
41	3	30	0.40	0.10	7780621	7768413	1476	13.01	7780621	7768413	1476	13.41	7623997	7612261	1476	41.98	7780621	7768413	1476	17.02
42	1	30	0.40	0.30	6089	0	2900	0.20	5539	0	2900	0.18	5522	0	2900	0.19	113	0	2900	0.03
43	2	30	0.40	0.30	6672	0	2900	0.21	6672	0	2900	0.21	6662	0	2900	0.14	161	0	2900	0.05
44	3	30	0.40	0.30	6147	0	2900	0.20	6147	0	2900	0.21	6132	0	2900	0.21	164	0	2900	0.05
45	1	30	0.40	0.50	100	0	2900	0.01	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
46	2	30	0.40	0.50	117	0	2900	0.01	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
47	3	30	0.40	0.50	129	0	2900	0.01	2	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
48	1	30	0.50	0.50	488	0	2900	0.03	371	0	2900	0.03	356	0	2900	0.03	33	0	2900	0.02
49	2	30	0.50	0.50	476	0	2900	0.03	391	0	2900	0.03	379	0	2900	0.03	30	0	2900	0.02
50	3	30	0.50	0.50	562	0	2900	0.03	478	0	2900	0.03	461	0	2900	0.03	38	0	2900	0.02
51	1	30	0.50	0.70	55	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
52	2	30	0.50	0.70	52	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
53	3	30	0.50	0.70	55	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
54	1	30	0.50	0.10	5495520	5493493	1274	6.06	5495520	5493493	1274	5.96	5447302	5445316	1274	24.42	5495520	5493493	1274	7.48

Table B.3 . The effect of IT and Pr on BB-D using random dataset. (cont.)

Instances				IT				IT+RootPr				IT+StaticPr				IT+DynamicPr				
ID	Rep	V(G)	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
55	2	30	0.50	0.10	76964075	76911985	1265	79.16	76964075	76911985	1265	78.29	75737696	75701591	1265	423.33	76964075	76911985	1265	98.85
56	3	30	0.50	0.10	9030702	9030466	1222	9.32	9030702	9030466	1222	9.26	8954889	8954567	1222	44.05	9030702	9030466	1222	9.78
57	1	30	0.50	0.30	56017	0	2900	1.29	56017	0	2900	1.32	54484	0	2900	1.90	1486	0	2900	0.49
58	2	30	0.50	0.30	66955	0	2900	1.54	66955	0	2900	1.51	64556	0	2900	1.47	1097	0	2900	0.49
59	3	30	0.50	0.30	74679	0	2900	1.81	74679	0	2900	1.79	71764	0	2900	1.71	1492	0	2900	0.49
60	1	30	0.70	0.30	2033335	0	2900	42.26	2033335	0	2900	41.32	874171	0	2900	34.19	36123	0	2900	8.16
61	2	30	0.70	0.30	1836033	0	2900	37.63	1836033	0	2900	36.95	786409	0	2900	30.19	34142	0	2900	8.61
62	3	30	0.70	0.30	2102280	0	2900	40.52	2102280	0	2900	40.60	856640	0	2900	32.00	41408	0	2900	10.51
63	1	30	0.70	0.50	5318	0	2900	0.29	5318	0	2900	0.29	4725	0	2900	0.26	186	0	2900	0.07
64	2	30	0.70	0.50	4912	0	2900	0.26	4854	0	2900	0.26	4508	0	2900	0.24	163	0	2900	0.07
65	3	30	0.70	0.50	5226	0	2900	0.28	5226	0	2900	0.28	4704	0	2900	0.26	198	0	2900	0.08
66	1	30	0.70	0.70	248	0	2900	0.03	125	0	2900	0.01	105	0	2900	0.01	7	0	2900	0.01
67	2	30	0.70	0.70	218	0	2900	0.02	16	0	2900	0.00	0	0	2900	0.00	0	0	2900	0.00
68	3	30	0.70	0.70	232	0	2900	0.02	44	0	2900	0.00	15	0	2900	0.00	2	0	2900	0.00
69	1	30	0.70	0.10	29239545	29238932	1078	36.78	29239545	29238932	1078	36.85	27916889	27916594	1078	288.90	29239545	29238932	1078	36.34
70	2	30	0.70	0.10	81544208	81542959	1096	94.33	81544208	81542959	1096	94.33	79832391	79831180	1096	571.78	81544208	81542959	1096	118.72
71	3	30	0.70	0.10	4100622	4098487	1131	4.72	4100622	4098487	1131	4.72	4037545	4035429	1131	15.69	4100622	4098487	1131	4.69
72	1	40	0.40	0.30	152973	0	3900	5.59	152973	0	3900	5.54	150863	0	3900	5.41	2750	0	3900	1.07
73	2	40	0.40	0.30	137537	0	3900	4.96	137537	0	3900	4.91	135732	0	3900	4.80	1317	0	3900	0.89
74	3	40	0.40	0.30	165337	0	3900	5.86	165337	0	3900	6.00	163079	0	3900	5.43	2574	0	3900	1.16
75	1	40	0.40	0.50	729	0	3900	0.08	491	0	3900	0.06	471	0	3900	0.06	46	0	3900	0.03
76	2	40	0.40	0.50	643	0	3900	0.07	348	0	3900	0.04	326	0	3900	0.04	28	0	3900	0.03
77	3	40	0.40	0.50	786	0	3900	0.08	632	0	3900	0.07	609	0	3900	0.07	51	0	3900	0.04
78	1	40	0.40	0.70	64	0	3900	0.01	0	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
79	2	40	0.40	0.70	62	0	3900	0.01	0	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
80	3	40	0.40	0.70	74	0	3900	0.01	0	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
81	1	40	0.40	0.10	811666294	805077342	1857*	1800	833261152	826512310	1857*	1800	61511773	60094479	1948*	1800	841448137	834632240	1857*	1800
82	2	40	0.40	0.10	470331321	448238795	2074*	1800	489702015	466996963	1999*	1800	55424202	50532194	2172*	1800	487558792	464891612	1999*	1800
83	3	40	0.40	0.10	821163287	807426689	1813*	1800	838306627	824452278	1813*	1800	51836549	46241037	1813*	1800	836549835	822711536	1813*	1800
84	1	40	0.50	0.30	1537957	0	3900	58.25	1537957	0	3900	57.48	1153793	0	3900	53.40	22399	0	3900	8.34
85	2	40	0.50	0.30	1638689	0	3900	57.85	1638689	0	3900	56.97	1216110	0	3900	52.50	17879	0	3900	8.78
86	3	40	0.50	0.30	1772503	0	3900	62.25	1772503	0	3900	61.52	1318032	0	3900	56.71	22931	0	3900	9.91
87	1	40	0.50	0.50	4215	0	3900	0.34	4106	0	3900	0.34	4044	0	3900	0.30	153	0	3900	0.10
88	2	40	0.50	0.50	4832	0	3900	0.36	4598	0	3900	0.34	4508	0	3900	0.33	130	0	3900	0.10
89	3	40	0.50	0.50	4896	0	3900	0.37	4846	0	3900	0.36	4750	0	3900	0.35	146	0	3900	0.11
90	1	40	0.50	0.70	211	0	3900	0.03	0	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
91	2	40	0.50	0.70	231	0	3900	0.03	0	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
92	3	40	0.50	0.70	227	0	3900	0.03	9	0	3900	0.00	0	0	3900	0.00	0	0	3900	0.00
93	1	40	0.50	0.10	735668478	728594347	1809*	1800	767464411	760328477	1809*	1800	31920761	30022338	1855*	1800	764587778	757455192	1809*	1800
94	2	40	0.50	0.10	365528084	326420545	2025*	1800	378070783	338520773	2025*	1800	40270743	28497359	2054*	1800	376960456	337430992	2025*	1800
95	3	40	0.50	0.10	574663782	560309455	1737*	1800	598248238	583676008	1737*	1800	30337074	27730807	1737*	1800	593959840	579419193	1737*	1800
96	1	40	0.70	0.70	498	0	3900	0.11	485	0	3900	0.10	459	0	3900	0.10	113	0	3900	0.06
97	2	40	0.70	0.70	510	0	3900	0.11	494	0	3900	0.10	468	0	3900	0.10	112	0	3900	0.06
98	3	40	0.70	0.70	504	0	3900	0.11	480	0	3900	0.10	455	0	3900	0.09	109	0	3900	0.06
99	1	40	0.70	0.10	839501816	839160687	1443*	1800	880276711	879932780	1443*	1800	25227560	25055916	1599*	1800	872784127	872440475	1443*	1800
100	2	40	0.70	0.10	832779158	832627393	1507*	1800	869798354	869644670	1507*	1800	28869978	28835109	1596*	1800	855816089	855663059	1507*	1800
101	3	40	0.70	0.10	901006864	900908809	1441*	1800	945369442	945270614	1441*	1800	29284036	29269068	1519*	1800	933167546	933068932	1441*	1800
102	1	40	0.70	0.30	33877253	0	-	1800	34267160	0	-	1800	8883067	0	3900	1434.19	993139	0	3900	390.16
103	2	40	0.70	0.30	33505930	0	-	1800	34108748	0	-	1800	8599313	0	3900	1379.28	968592	0	3900	373.54
104	3	40	0.70	0.30	32926185	0	-	1800	33352296	0	-	1800	8622585	0	3900	1392.52	894181	0	3900	353.45
105	1	40	0.70	0.50	32549	0	3900	3.13	32549	0	3900	2.79	21280	0	3900	2.52	2165	0	3900	1.05
106	2	40	0.70	0.50	32114	0	3900	2.73	32114	0	3900	2.71	21175	0	3900	2.50	2005	0	3900	0.94
107	3	40	0.70	0.50	31268	0	3900	2.69	31268	0	3900	2.67	20841	0	3900	2.44	2082	0	3900	0.97
108	1	50	0.40	0.10	74513314	0	-	1800	78582668	0	-	1800	5298456	0	-	1800	77469780	0	-	1800
109	2	50	0.40	0.10	82504309	0	-	1800	83237441	0	-	1800	16776525	11604553	3426*	1800	84783491	0	-	1800

Table B.3 . The effect of IT and Pr on BB-D using random dataset. (cont.)

Instances				IT			IT+RootPr			IT+StaticPr			IT+DynamicPr			
ID	Rep	V(G)	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
110	3	50	0.40	0.10	99595603	0	-	1800	100664553	0	-	1800	7058151	0	-	1800
111	1	50	0.40	0.30	2144498	0	4900	118.94	2144498	0	4900	117.41	1875700	0	4900	116.65
112	2	50	0.40	0.30	1798218	0	4900	105.43	1798218	0	4900	105.54	1619216	0	4900	101.45
113	3	50	0.40	0.30	2122030	0	4900	122.89	2122030	0	4900	120.55	1861020	0	4900	115.77
114	1	50	0.40	0.50	5265	0	4900	0.53	5137	0	4900	0.54	5091	0	4900	0.51
115	2	50	0.40	0.50	4341	0	4900	0.44	3900	0	4900	0.40	3866	0	4900	0.39
116	3	50	0.40	0.50	4970	0	4900	0.51	4805	0	4900	0.49	4768	0	4900	0.47
117	1	50	0.40	0.70	257	0	4900	0.05	0	0	4900	0.00	0	0	4900	0.00
118	2	50	0.40	0.70	213	0	4900	0.03	0	0	4900	0.00	0	0	4900	0.00
119	3	50	0.40	0.70	224	0	4900	0.04	0	0	4900	0.00	0	0	4900	0.00
120	1	50	0.50	0.70	504	0	4900	0.13	121	0	4900	0.01	56	0	4900	0.01
121	2	50	0.50	0.70	469	0	4900	0.12	34	1	4900	0.01	1	0	4900	0.01
122	3	50	0.50	0.70	512	0	4900	0.13	193	0	4900	0.02	154	0	4900	0.02
123	1	50	0.50	0.10	94114007	0	-	1800	95198032	0	-	1800	6243151	0	-	1800
124	2	50	0.50	0.10	69142882	0	-	1800	69909909	0	-	1800	13945012	0	-	1800
125	3	50	0.50	0.10	80094689	0	-	1800	81108112	0	-	1800	8950603	0	-	1800
126	1	50	0.50	0.30	26471451	0	-	1800	28344023	0	-	1800	11341138	0	4900	1372.40
127	2	50	0.50	0.30	25421904	0	4900	1606.10	25421904	0	4900	1585.02	10708616	0	4900	1191.84
128	3	50	0.50	0.30	27767094	0	4900	1677.47	27767094	0	4900	1693.75	10918214	0	4900	1265.76
129	1	50	0.50	0.50	24347	0	4900	2.69	24347	0	4900	2.68	22095	0	4900	2.67
130	2	50	0.50	0.50	21621	0	4900	2.40	21621	0	4900	2.38	20285	0	4900	2.37
131	3	50	0.50	0.50	23415	0	4900	2.93	23415	0	4900	2.43	21606	0	4900	2.53
132	1	50	0.70	0.30	16582228	0	-	1800	16784653	0	-	1800	1622939	0	-	1800
133	2	50	0.70	0.30	16165473	0	-	1800	16327945	0	-	1800	1471670	0	-	1800
134	3	50	0.70	0.30	16124352	0	-	1800	16270281	0	-	1800	1491169	0	-	1800
135	1	50	0.70	0.50	177812	0	4900	23.08	177812	0	4900	22.89	58188	0	4900	18.48
136	2	50	0.70	0.50	174891	0	4900	24.02	174891	0	4900	23.07	58697	0	4900	18.31
137	3	50	0.70	0.50	185042	0	4900	29.31	185042	0	4900	24.88	59151	0	4900	18.81
138	1	50	0.70	0.70	852	0	4900	0.45	848	0	4900	0.45	800	0	4900	0.44
139	2	50	0.70	0.70	852	0	4900	0.46	849	0	4900	0.45	804	0	4900	0.45
140	3	50	0.70	0.70	861	0	4900	0.46	860	0	4900	0.45	808	0	4900	0.45
141	1	50	0.70	0.10	259582625	242039299	2115*	1800	264973124	247299180	2115*	1800	18902742	14905051	2350*	1800
142	2	50	0.70	0.10	138050647	79040468	2153*	1800	142675437	83516481	2153*	1800	15590769	10390348	2153*	1800
143	3	50	0.70	0.10	367072149	360677179	2035*	1800	377073179	370589150	2035*	1800	14159595	13123524	2074*	1800
144	1	75	0.40	0.30	7114741	0	-	1800	7070252	0	-	1800	2189768	0	-	1800
145	2	75	0.40	0.30	7122928	0	-	1800	7213465	0	-	1800	2199305	0	-	1800
146	3	75	0.40	0.30	7058941	0	-	1800	7134134	0	-	1800	2190677	0	-	1800
147	1	75	0.40	0.50	101195	0	7400	31.36	101195	0	7400	31.50	83602	0	7400	31.06
148	2	75	0.40	0.50	100094	0	7400	31.73	100094	0	7400	31.58	82812	0	7400	31.42
149	3	75	0.40	0.50	100269	0	7400	32.17	100269	0	7400	30.99	83650	0	7400	31.45
150	1	75	0.40	0.70	991	0	7400	0.76	909	0	7400	0.54	869	0	7400	0.56
151	2	75	0.40	0.70	1008	0	7400	0.77	907	0	7400	0.55	863	0	7400	0.56
152	3	75	0.40	0.70	994	0	7400	0.76	922	0	7400	0.55	879	0	7400	0.57
153	1	75	0.40	0.10	28826259	0	-	1800	29177078	0	-	1800	597022	0	-	1800
154	2	75	0.40	0.10	26117539	0	-	1800	26555127	0	-	1800	837124	0	-	1800
155	3	75	0.40	0.10	24760479	0	-	1800	24948179	0	-	1800	794593	0	-	1800
156	1	75	0.50	0.30	6495717	0	-	1800	6530889	0	-	1800	735201	0	-	1800
157	2	75	0.50	0.30	6372278	0	-	1800	6530782	0	-	1800	716117	0	-	1800
158	3	75	0.50	0.30	6603407	0	-	1800	6665072	0	-	1800	675954	0	-	1800
159	1	75	0.50	0.50	404436	0	7400	135.70	404436	0	7400	136.59	153977	0	7400	101.14
160	2	75	0.50	0.50	433603	0	7400	143.31	433603	0	7400	141.99	153466	0	7400	102.88
161	3	75	0.50	0.50	457560	0	7400	147.21	457560	0	7400	145.83	154772	0	7400	105.54
162	1	75	0.50	0.70	1354	0	7400	1.70	1351	0	7400	1.70	1301	0	7400	1.51
163	2	75	0.50	0.70	1361	0	7400	1.71	1360	0	7400	1.66	1306	0	7400	1.52
164	3	75	0.50	0.70	1366	0	7400	1.66	1362	0	7400	1.64	1306	0	7400	1.53

Table B.3 . The effect of IT and Pr on BB-D using random dataset. (cont.)

Instances				IT			IT+RootPr			IT+StaticPr			IT+DynamicPr			
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
165	1	75	0.50	0.10	25462953	0	-	1800	25630677	0	-	1800	853553	0	-	1800
166	2	75	0.50	0.10	240051147	0	-	1800	24164734	0	-	1800	723852	0	-	1800
167	3	75	0.50	0.10	21862507	0	-	1800	22081134	0	-	1800	737923	0	-	1800
168	1	75	0.70	0.70	16169	0	7400	7.22	16169	0	7400	7.08	2052	0	7400	5.51
169	2	75	0.70	0.70	16047	0	7400	7.17	16047	0	7400	7.14	1992	0	7400	5.81
170	3	75	0.70	0.70	15517	0	7400	7.06	15517	0	7400	7.68	2017	0	7400	5.48
171	1	75	0.70	0.10	20604044	0	-	1800	20754659	0	-	1800	730895	0	-	1800
172	2	75	0.70	0.10	23093805	0	-	1800	23257049	0	-	1800	648402	0	-	1800
173	3	75	0.70	0.10	18655986	0	-	1800	18712639	0	-	1800	657054	0	-	1800
174	1	75	0.70	0.30	6328044	0	-	1800	6349033	0	-	1800	95079	0	-	1800
175	2	75	0.70	0.30	6441935	0	-	1800	6520300	0	-	1800	116009	0	-	1800
176	3	75	0.70	0.30	6477977	0	-	1800	6448619	0	-	1800	114142	0	-	1800
177	1	75	0.70	0.50	4538223	0	-	1800	4573310	0	-	1800	541037	0	-	1800
178	2	75	0.70	0.50	4451610	0	-	1800	4155497	0	-	1800	522995	0	-	1800
179	3	75	0.70	0.50	4500586	0	-	1800	4120185	0	-	1800	518612	0	-	1800
180	1	100	0.40	0.70	1923	0	9900	8.90	1922	0	9900	7.05	1864	0	9900	9.28
181	2	100	0.40	0.70	1930	0	9900	6.05	1928	0	9900	5.21	1869	0	9900	5.12
182	3	100	0.40	0.70	1939	0	9900	5.27	1936	0	9900	5.20	1865	0	9900	6.39
183	1	100	0.40	0.10	11427650	0	-	1800	11658802	0	-	1800	271471	0	-	1800
184	2	100	0.40	0.10	10857921	0	-	1800	10850719	0	-	1800	314519	0	-	1800
185	3	100	0.40	0.10	10740979	0	-	1800	10798839	0	-	1800	345283	0	-	1800
186	1	100	0.40	0.30	3232802	0	-	1800	3222790	0	-	1800	294364	0	-	1800
187	2	100	0.40	0.30	3154084	0	-	1800	3192705	0	-	1800	326975	0	-	1800
188	3	100	0.40	0.30	3162045	0	-	1800	3188061	0	-	1800	315622	0	-	1800
189	1	100	0.40	0.50	961874	0	9900	662.08	961874	0	9900	652.74	316498	0	9900	486.53
190	2	100	0.40	0.50	942339	0	9900	647.34	942339	0	9900	643.26	316172	0	9900	527.23
191	3	100	0.40	0.50	946662	0	9900	647.04	946662	0	9900	652.73	316845	0	9900	485.05
192	1	100	0.50	0.50	2068899	0	-	1800	2075063	0	-	1800	225302	0	-	1800
193	2	100	0.50	0.50	2106800	0	-	1800	2085533	0	-	1800	228088	0	-	1800
194	3	100	0.50	0.50	2077095	0	-	1800	2099645	0	-	1800	227235	0	-	1800
195	1	100	0.50	0.70	6058	0	9900	12.55	6058	0	9900	12.14	2390	0	9900	11.89
196	2	100	0.50	0.70	5978	0	9900	11.93	5930	0	9900	12.03	2391	0	9900	11.26
197	3	100	0.50	0.70	6350	0	9900	16.42	6350	0	9900	13.47	2397	0	9900	11.25
198	1	100	0.50	0.10	10034330	0	-	1800	10013904	0	-	1800	257666	0	-	1800
199	2	100	0.50	0.10	10671693	0	-	1800	10679061	0	-	1800	244594	0	-	1800
200	3	100	0.50	0.10	9707460	0	-	1800	9722735	0	-	1800	287378	0	-	1800
201	1	100	0.50	0.30	3071532	0	-	1800	3050217	0	-	1800	60713	0	-	1800
202	2	100	0.50	0.30	3051549	0	-	1800	3055233	0	-	1800	68598	0	-	1800
203	3	100	0.50	0.30	3044770	0	-	1800	3055370	0	-	1800	71846	0	-	1800
204	1	100	0.70	0.30	2621668	0	-	1800	2626310	0	-	1800	48130	0	-	1800
205	2	100	0.70	0.30	2693456	0	-	1800	2693214	0	-	1800	52047	0	-	1800
206	3	100	0.70	0.30	2631326	0	-	1800	2641316	0	-	1800	54713	0	-	1800
207	1	100	0.70	0.50	1242720	0	-	1800	1237867	0	-	1800	25529	0	-	1800
208	2	100	0.70	0.50	1230881	0	-	1800	1222359	0	-	1800	24821	0	-	1800
209	3	100	0.70	0.50	1240422	0	-	1800	1244806	0	-	1800	25350	0	-	1800
210	1	100	0.70	0.70	303871	0	9900	219.58	303871	0	9900	219.68	16112	0	9900	110.04
211	2	100	0.70	0.70	307556	0	9900	226.46	307556	0	9900	224.66	16599	0	9900	108.86
212	3	100	0.70	0.70	307313	0	9900	224.31	307313	0	9900	223.32	16034	0	9900	109.22
213	1	100	0.70	0.10	8488612	0	-	1800	8337352	0	-	1800	272624	0	-	1800
214	2	100	0.70	0.10	10251870	0	-	1800	10168046	0	-	1800	280697	0	-	1800
215	3	100	0.70	0.10	8833330	0	-	1800	8812703	0	-	1800	272249	0	-	1800
Average				0.53 0.40	39384260	33116191	3843	618.30	40692244	34348684	3842	618.09	3608367	2815985	3901	601.07
													38669434	33984094	4021	535.60

Table B.4. The effect of IT and Pr on BB-E using random dataset.

ID	Rep	Instances			IT			IT+StaticPr			IT+DynamicPr		
		$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	CPU(s)
0	1	20	0.70	0.30	6534	0	1900	0.08	5970	0	1900	0.09	0.07
1	2	20	0.70	0.30	7328	0	1900	0.09	6800	0	1900	0.13	0.08
2	3	20	0.70	0.30	7842	0	1900	0.09	7380	0	1900	0.10	0.08
3	1	20	0.70	0.50	287	0	1900	0.04	300	0	1900	0.04	0.03
4	2	20	0.70	0.50	432	0	1900	0.03	174	0	1900	0.04	0.03
5	3	20	0.70	0.50	345	0	1900	0.03	250	0	1900	0.03	0.03
6	1	20	0.70	0.70	54	0	1900	0.03	0	0	1900	0.08	0.07
7	2	20	0.70	0.70	85	0	1900	0.03	0	0	1900	0.03	0.03
8	3	20	0.70	0.70	75	0	1900	0.03	0	0	1900	0.08	0.07
9	1	20	0.70	0.10	799	93	750	0.17	799	93	750	0.08	0.07
10	2	20	0.70	0.10	1652	320	777	0.20	1652	320	777	0.09	0.09
11	3	20	0.70	0.10	585	225	769	0.10	585	225	769	0.08	0.08
12	1	20	0.40	0.50	9	0	1900	0.03	0	0	1900	0.08	0.06
13	2	20	0.40	0.50	17	0	1900	0.03	0	0	1900	0.07	0.07
14	3	20	0.40	0.50	36	0	1900	0.03	0	0	1900	0.03	0.03
15	1	20	0.40	0.70	29	0	1900	0.03	0	0	1900	0.08	0.06
16	2	20	0.40	0.70	17	0	1900	0.03	0	0	1900	0.07	0.07
17	3	20	0.40	0.70	44	0	1900	0.03	0	0	1900	0.08	0.07
18	1	20	0.40	0.10	235	87	1022	0.09	235	87	1022	0.08	0.06
19	2	20	0.40	0.10	218	81	964	0.09	218	81	964	0.07	0.07
20	3	20	0.40	0.10	1590	472	1190	0.11	1590	472	1190	0.08	0.08
21	1	20	0.40	0.30	58	0	1900	0.03	0	0	1900	0.03	0.03
22	2	20	0.40	0.30	88	0	1900	0.03	7	0	1900	0.03	0.03
23	3	20	0.40	0.30	165	0	1900	0.03	139	0	1900	0.03	0.03
24	1	20	0.50	0.30	457	0	1883	0.03	387	0	1883	0.04	0.03
25	2	20	0.50	0.30	519	0	1900	0.03	364	0	1900	0.03	0.03
26	3	20	0.50	0.30	480	0	1900	0.03	311	0	1900	0.03	0.03
27	1	20	0.50	0.50	74	0	1900	0.03	0	0	1900	0.03	0.03
28	2	20	0.50	0.50	52	0	1900	0.03	0	0	1900	0.07	0.07
29	3	20	0.50	0.50	62	0	1900	0.03	0	0	1900	0.03	0.03
30	1	20	0.50	0.70	21	0	1900	0.03	0	0	1900	0.08	0.06
31	2	20	0.50	0.70	21	0	1900	0.03	0	0	1900	0.07	0.07
32	3	20	0.50	0.70	53	0	1900	0.03	0	0	1900	0.07	0.07
33	1	20	0.50	0.10	576	173	893	0.09	576	173	893	0.08	0.07
34	2	20	0.50	0.10	541	257	879	0.08	541	257	879	0.08	0.08
35	3	20	0.50	0.10	161	69	924	0.09	161	69	924	0.07	0.07
36	1	30	0.40	0.70	71	0	2900	0.03	0	0	2900	0.09	0.10
37	2	30	0.40	0.70	49	0	2900	0.03	0	0	2900	0.09	0.10
38	3	30	0.40	0.70	42	0	2900	0.03	0	0	2900	0.09	0.09
39	1	30	0.40	0.10	34377	15481	1350	0.43	34384	15475	1350	0.39	0.42
40	2	30	0.40	0.10	236391	94528	1383	1.69	236302	94439	1383	2.11	2.19
41	3	30	0.40	0.10	460392	197673	1476	3.06	460081	197365	1476	3.49	3.93
42	1	30	0.40	0.30	1880	0	2900	0.05	1721	0	2900	0.05	0.04
43	2	30	0.40	0.30	2506	0	2900	0.05	2450	0	2900	0.06	0.04
44	3	30	0.40	0.30	2406	0	2900	0.05	2214	0	2900	0.06	0.03
45	1	30	0.40	0.50	217	0	2900	0.03	0	0	2900	0.09	0.08
46	2	30	0.40	0.50	188	0	2900	0.03	0	0	2900	0.03	0.03
47	3	30	0.40	0.50	225	0	2900	0.03	0	0	2900	0.03	0.03
48	1	30	0.50	0.50	806	0	2900	0.03	676	0	2900	0.04	0.03
49	2	30	0.50	0.50	957	0	2900	0.04	756	0	2900	0.04	0.03
50	3	30	0.50	0.50	927	0	2900	0.04	764	0	2900	0.04	0.03
51	1	30	0.50	0.70	123	0	2900	0.03	0	0	2900	0.09	0.08
52	2	30	0.50	0.70	96	0	2900	0.03	0	0	2900	0.08	0.08
53	3	30	0.50	0.70	88	0	2900	0.03	0	0	2900	0.08	0.08
54	1	30	0.50	0.10	290671	139657	1274	2.16	290666	139652	1274	2.07	2.26

Table B.4 . The effect of IT and Pr on BB-E using random dataset. (cont.)

Instances				IT			IT+StaticPr			IT+DynamicPr		
ID	Rep	$V(G)$	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
55	2	30	0.50	0.10	1228002	526523	1265	6.41	1227871	526406	1265	8.16
56	3	30	0.50	0.10	151556	58733	1222	1.18	151556	58733	1222	1.40
57	1	30	0.50	0.30	18611	0	2900	0.22	18436	0	2900	0.25
58	2	30	0.50	0.30	16454	0	2900	0.21	16225	0	2900	0.23
59	3	30	0.50	0.30	22216	0	2900	0.28	21842	0	2900	0.29
60	1	30	0.70	0.30	362091	0	2900	3.17	272588	0	2900	3.46
61	2	30	0.70	0.30	373942	0	2900	3.35	279909	0	2900	3.59
62	3	30	0.70	0.30	406401	0	2900	3.55	301909	0	2900	3.79
63	1	30	0.70	0.50	4330	0	2900	0.08	3724	0	2900	0.09
64	2	30	0.70	0.50	3926	0	2900	0.07	3401	0	2900	0.09
65	3	30	0.70	0.50	4217	0	2900	0.08	3613	0	2900	0.09
66	1	30	0.70	0.70	597	0	2900	0.03	0	0	2900	0.03
67	2	30	0.70	0.70	546	0	2900	0.03	0	0	2900	0.03
68	3	30	0.70	0.70	550	0	2900	0.03	0	0	2900	0.03
69	1	30	0.70	0.10	675820	188789	1078	4.68	675820	188789	1078	4.79
70	2	30	0.70	0.10	425813	185194	1096	3.13	425813	185194	1096	2.69
71	3	30	0.70	0.10	307317	79970	1131	2.72	307317	79970	1131	2.82
72	1	40	0.40	0.30	33440	0	3900	0.42	32979	0	3900	0.44
73	2	40	0.40	0.30	24991	0	3900	0.35	24636	0	3900	0.42
74	3	40	0.40	0.30	43614	0	3900	0.55	43259	0	3900	0.63
75	1	40	0.40	0.50	1246	0	3900	0.04	938	0	3900	0.04
76	2	40	0.40	0.50	1155	0	3900	0.04	593	0	3900	0.04
77	3	40	0.40	0.50	1126	0	3900	0.04	1235	0	3900	0.05
78	1	40	0.40	0.70	108	0	3900	0.02	0	0	3900	0.08
79	2	40	0.40	0.70	138	0	3900	0.02	0	0	3900	0.10
80	3	40	0.40	0.70	138	0	3900	0.03	0	0	3900	0.08
81	1	40	0.40	0.10	261956711	127352968	-	1800	186195558	90563908	-	1806.98
82	2	40	0.40	0.10	218270508	59551698	1883	1707.37	193335556	80612115	1886*	1800.05
83	3	40	0.40	0.10	177682796	76389788	1753	1235.21	176792953	75511788	1753	1463.45
84	1	40	0.50	0.30	328064	0	3900	2.72	303671	0	3900	3.18
85	2	40	0.50	0.30	335235	0	3900	3.67	308257	0	3900	3.25
86	3	40	0.50	0.30	353398	0	3900	2.98	327653	0	3900	3.10
87	1	40	0.50	0.50	3270	0	3900	0.06	3152	0	3900	0.08
88	2	40	0.50	0.50	4477	0	3900	0.08	5095	0	3900	0.10
89	3	40	0.50	0.50	3855	0	3900	0.07	3703	0	3900	0.08
90	1	40	0.50	0.70	419	0	3900	0.03	0	0	3900	0.02
91	2	40	0.50	0.70	478	0	3900	0.03	0	0	3900	0.08
92	3	40	0.50	0.70	543	0	3900	0.03	0	0	3900	0.02
93	1	40	0.50	0.10	266001736	132673827	-	1800	217733719	108615218	-	1800
94	2	40	0.50	0.10	247228712	123082864	-	1800	231812345	115389511	-	1800.06
95	3	40	0.50	0.10	250896704	125038189	-	1800	220758865	109993342	-	1800
96	1	40	0.70	0.70	3048	0	3900	0.05	1215	0	3900	0.04
97	2	40	0.70	0.70	2597	0	3900	0.05	1540	0	3900	0.05
98	3	40	0.70	0.70	2749	0	3900	0.05	1431	0	3900	0.06
99	1	40	0.70	0.10	237622864	118801333	-	1800	201536611	100760647	-	1800
100	2	40	0.70	0.10	221002308	110491972	-	1800	208731886	104357591	-	1800.01
101	3	40	0.70	0.10	215389784	97583355	1397	1795.45	210130457	105056414	-	1800.01
102	1	40	0.70	0.30	9996569	0	3900	84.07	3933823	0	3900	87.87
103	2	40	0.70	0.30	12624849	0	3900	106.21	5160238	0	3900	106.40
104	3	40	0.70	0.30	11364995	0	3900	93.75	4851113	0	3900	95.55
105	1	40	0.70	0.50	34648	0	3900	0.43	19114	0	3900	0.50
106	2	40	0.70	0.50	39450	0	3900	0.53	22894	0	3900	0.54
107	3	40	0.70	0.50	32881	0	3900	0.41	19177	0	3900	0.49
108	1	50	0.40	0.10	194156637	96933851	-	1800	18418764	91921661	-	1800.01
109	2	50	0.40	0.10	199035510	99170181	-	1800	186384466	92861484	-	1800

Table B.4 . The effect of IT and Pr on BB-E using random dataset. (cont.)

Instances			IT			IT+StaticPr			IT+DynamicPr			
ID	Rep	$V(G)$	Ω	Γ	Active	Explored	Best UB	CPU(s)	Active	Explored	Best UB	CPU(s)
165	1	75	0.50	0.10	50383213	100766436	-	1800	96977371	48488681	-	1800
166	2	75	0.50	0.10	98913544	49456773	-	1800	95933082	47966542	-	1800
167	3	75	0.50	0.10	100053734	50026869	-	1800	98069922	49034963	-	1800
168	1	75	0.70	0.70	60797	60797	7400	1.61	49413	0	7400	1.44
169	2	75	0.70	0.70	55588	55588	7400	1.08	46168	0	7400	1.32
170	3	75	0.70	0.70	49995	49995	7400	0.94	44032	0	7400	1.18
171	1	75	0.70	0.10	91828106	91828106	-	1800	88013134	44006569	-	1800.01
172	2	75	0.70	0.10	93503432	46751718	-	1800	88727582	44363793	-	1800
173	3	75	0.70	0.10	92915642	46457823	-	1800	89240012	44620008	-	1800.01
174	1	75	0.70	0.30	40595923	19848674	-	1800	16255626	7262349	-	1800
175	2	75	0.70	0.30	40884342	20011408	-	1800	16229583	7204741	-	1800
176	3	75	0.70	0.30	38603576	18900888	-	1800	15886695	7126799	-	1800.01
177	1	75	0.70	0.50	8642323	0	7400	195.25	1243864	0	7400	182.91
178	2	75	0.70	0.50	7340278	0	7400	170.30	1235458	0	7400	162.06
179	3	75	0.70	0.50	7772335	0	7400	178.12	1209214	0	7400	174.00
180	1	100	0.40	0.70	35712	0	9900	0.82	34682	0	9900	1.20
181	2	100	0.40	0.70	34534	0	9900	1.27	34005	0	9900	1.32
182	3	100	0.40	0.70	31972	0	9900	0.72	29875	0	9900	0.84
183	1	100	0.40	0.10	66810594	33405299	-	1800	64563348	32281676	-	1800
184	2	100	0.40	0.10	68636706	34318355	-	1800	64020006	32010005	-	1800
185	3	100	0.40	0.10	69507104	34753554	-	1800	64543120	32271562	-	1800
186	1	100	0.40	0.30	28522205	12976448	-	1800	18201765	8001585	-	1800
187	2	100	0.40	0.30	28502530	12825372	-	1800	18036847	7838031	-	1800
188	3	100	0.40	0.30	27113239	12374063	-	1800	17239066	7546742	-	1800
189	1	100	0.40	0.50	1627368	0	9900	37.97	816955	0	9900	51.93
190	2	100	0.40	0.50	1712234	0	9900	39.48	832587	0	9900	53.46
191	3	100	0.40	0.50	2050088	0	9900	47.97	892798	0	9900	66.43
192	1	100	0.50	0.50	7466491	0	9900	212.58	2243167	0	9900	265.75
193	2	100	0.50	0.50	8484288	0	9900	239.64	2312987	0	9900	306.07
194	3	100	0.50	0.50	7244366	0	9900	206.83	2181187	0	9900	264.53
195	1	100	0.50	0.70	88868	0	9900	2.17	74443	0	9900	2.79
196	2	100	0.50	0.70	47446	0	9900	1.55	45506	0	9900	1.38
197	3	100	0.50	0.70	74867	0	9900	1.86	66681	0	9900	2.68
198	1	100	0.50	0.10	62674702	31337353	-	1800	60322834	30161419	-	1800
199	2	100	0.50	0.10	63370242	31685123	-	1800	61742904	30871454	-	1800
200	3	100	0.50	0.10	62697356	31348680	-	1800	61869730	30934867	-	1800
201	1	100	0.50	0.30	24269746	12031247	-	1800	18224115	8833134	-	1800
202	2	100	0.50	0.30	24025819	11972302	-	1800	18016060	8753223	-	1800
203	3	100	0.50	0.30	24134513	11971107	-	1800	17895442	8686338	-	1800
204	1	100	0.70	0.30	21937846	10968016	-	1800	15095634	7518198	-	1800.01
205	2	100	0.70	0.30	22037471	11017937	-	1800	15433915	7684709	-	1800.01
206	3	100	0.70	0.30	21901740	10949938	-	1800	14966174	7446120	-	1800
207	1	100	0.70	0.50	25062107	4502360	-	1800	2215017	88585	-	1800.01
208	2	100	0.70	0.50	24479739	4461524	-	1800	2142104	90705	-	1800
209	3	100	0.70	0.50	24505667	4508636	-	1800	2146191	91426	-	1800
210	1	100	0.70	0.70	312458	0	9900	11.86	164534	0	9900	15.93
211	2	100	0.70	0.70	277616	0	9900	12.00	150187	0	9900	14.91
212	3	100	0.70	0.70	287784	0	9900	13.20	154625	0	9900	15.05
213	1	100	0.70	0.10	55023376	27511690	-	1800	56127136	28063570	-	1800
214	2	100	0.70	0.10	56215902	28017953	-	1800	56158604	28079304	-	1800
215	3	100	0.70	0.10	58218646	29109325	-	1800	57816370	28908187	-	1800
Average			0.53	0.40	30084450	12847822	4203	497.01	24504034	11405681	4208	502.98
												496.12

Table B.5. The effect of IT and Pr on BB-S using random dataset.

ID	Rep	Instances			IT			IT+StaticPr			IT+DynamicPr		
		$V(G)$	f	Ω	Explored	Active	Best UB	Explored	Active	Best UB	Explored	Active	Best UB
0	1	20	0.70	0.30	3402	0	1900	2863	0	1900	57	0	1900
1	2	20	0.70	0.30	3131	0	1900	2696	0	1900	51	0	1900
2	3	20	0.70	0.30	2893	0	1900	2530	0	1900	43	0	1900
3	1	20	0.70	0.50	44	0	1900	35	0	1900	0	0	1900
4	2	20	0.70	0.50	47	0	1900	39	0	1900	0	0	1900
5	3	20	0.70	0.50	50	0	1900	36	0	1900	0	0	1900
6	1	20	0.70	0.70	7	0	1900	0	0	1900	0	0	1900
7	2	20	0.70	0.70	5	0	1900	0	0	1900	0	0	1900
8	3	20	0.70	0.70	9	0	1900	0	0	1900	0	0	1900
9	1	20	0.70	0.10	488812	11007	750	472755	10820	750	488812	11007	750
10	2	20	0.70	0.10	2439814	204780	777	2316651	182334	777	2439814	204780	777
11	3	20	0.70	0.10	1591063	452909	769	1527607	442690	769	1591063	452909	769
12	1	20	0.40	0.50	1	0	1900	0	0	1900	0	0	1900
13	2	20	0.40	0.50	0	0	1900	0	0	1900	0	0	1900
14	3	20	0.40	0.50	0	0	1900	0	0	1900	0	0	1900
15	1	20	0.40	0.70	0	0	1900	0	0	1900	0	0	1900
16	2	20	0.40	0.70	0	0	1900	0	0	1900	0	0	1900
17	3	20	0.40	0.70	0	0	1900	0	0	1900	0	0	1900
18	1	20	0.40	0.10	4487	1499	1022	4476	1492	1022	4487	1499	1022
19	2	20	0.40	0.10	3200	1746	964	3197	1740	964	3200	1746	964
20	3	20	0.40	0.10	25247	4366	1190	24940	4318	1190	25247	4366	1190
21	1	20	0.40	0.30	18	0	1900	0	0	1900	0	0	1900
22	2	20	0.40	0.30	28	0	1900	7	0	1900	0	0	1900
23	3	20	0.40	0.30	30	0	1900	20	0	1900	1	0	1900
24	1	20	0.50	0.30	218	0	1883	179	0	1883	4	0	1883
25	2	20	0.50	0.30	148	0	1900	80	0	1900	2	0	1900
26	3	20	0.50	0.30	151	0	1900	115	0	1900	1	0	1900
27	1	20	0.50	0.50	5	0	1900	0	0	1900	0	0	1900
28	2	20	0.50	0.50	5	0	1900	0	0	1900	0	0	1900
29	3	20	0.50	0.50	14	0	1900	0	0	1900	0	0	1900
30	1	20	0.50	0.70	0	0	1900	0	0	1900	0	0	1900
31	2	20	0.50	0.70	0	0	1900	0	0	1900	0	0	1900
32	3	20	0.50	0.70	0	0	1900	0	0	1900	0	0	1900
33	1	20	0.50	0.10	50134	7143	893	47903	5496	893	50134	7143	893
34	2	20	0.50	0.10	19671	2013	879	19091	1998	879	19671	2013	879
35	3	20	0.50	0.10	3496	1388	924	3506	1400	924	3496	1388	924
36	1	30	0.40	0.70	0	0	2900	0	0	2900	0	0	2900
37	2	30	0.40	0.70	0	0	2900	0	0	2900	0	0	2900
38	3	30	0.40	0.70	0	0	2900	0	0	2900	0	0	2900
39	1	30	0.40	0.10	35114273	11561335	1350	36461721	13563921	1350	35114273	11561335	1350
40	2	30	0.40	0.10	62152417	28854418	1383*	48526554	35543922	1383*	62397951	28438344	1383*
41	3	30	0.40	0.10	98569015	75122043	1492*	63643217	51803110	1492*	99698844	75839847	1492*
42	1	30	0.40	0.30	490	0	2900	396	0	2900	6	0	2900
43	2	30	0.40	0.30	824	0	2900	745	0	2900	3	0	2900
44	3	30	0.40	0.30	587	0	2900	520	0	2900	3	0	2900
45	1	30	0.40	0.50	20	0	2900	0	0	2900	0	0	2900
46	2	30	0.40	0.50	24	0	2900	1	0	2900	0	0	2900
47	3	30	0.40	0.50	24	0	2900	0	0	2900	0	0	2900
48	1	30	0.50	0.50	89	0	2900	70	0	2900	0	0	2900
49	2	30	0.50	0.50	72	0	2900	69	0	2900	0	0	2900
50	3	30	0.50	0.50	91	0	2900	77	0	2900	0	0	2900
51	1	30	0.50	0.70	5	0	2900	0	0	2900	0	0	2900
52	2	30	0.50	0.70	4	0	2900	0	0	2900	0	0	2900
53	3	30	0.50	0.70	3	0	2900	0	0	2900	0	0	2900
54	1	30	0.50	0.10	121182556	115086795	1442*	73739665	69955091	1385*	135018117	128079344	1426*
							1800.05			1800.03			1800.04

Table B.5 . The effect of IT and Pr on BB-S using random dataset. (cont.)

Instances			IT			IT+StaticPr			IT+DynamicPr			
ID	Rep	V(G)	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
55	2	30	0.50	0.10	127628637	119442394	1298*	1800.04	72947017	68831625	1305*	1800.04
56	3	30	0.50	0.10	118373555	107998687	1259*	1800.05	69533342	64415313	1240*	1800.05
57	1	30	0.50	0.30	5503	0	2900	0.59	5333	0	2900	0.64
58	2	30	0.50	0.30	6598	0	2900	0.78	6348	0	2900	0.71
59	3	30	0.50	0.30	7400	0	2900	0.80	7091	0	2900	0.75
60	1	30	0.70	0.30	171189	0	2900	12.36	89810	0	2900	8.95
61	2	30	0.70	0.30	154452	0	2900	11.43	82390	0	2900	8.61
62	3	30	0.70	0.30	171825	0	2900	12.31	87531	0	2900	8.42
63	1	30	0.70	0.50	625	0	2900	0.15	467	0	2900	0.14
64	2	30	0.70	0.50	596	0	2900	0.15	469	0	2900	0.14
65	3	30	0.70	0.50	661	0	2900	0.17	499	0	2900	0.10
66	1	30	0.70	0.70	52	0	2900	0.03	10	0	2900	0.03
67	2	30	0.70	0.70	46	0	2900	0.03	0	0	2900	0.03
68	3	30	0.70	0.70	50	0	2900	0.03	0	0	2900	0.03
69	1	30	0.70	0.10	114871178	112626665	1218*	1800.03	62962696	61786837	1214*	1800.05
70	2	30	0.70	0.10	123626024	120991589	1227*	1800.05	56678065	55575633	1246*	1800.05
71	3	30	0.70	0.10	126829144	124229678	1297*	1800.03	61230881	60079736	1297*	1800.03
72	1	40	0.40	0.30	10830	0	3900	1.70	10609	0	3900	2.11
73	2	40	0.40	0.30	8585	0	3900	1.57	8490	0	3900	1.48
74	3	40	0.40	0.30	12089	0	3900	1.84	11877	0	3900	1.85
75	1	40	0.40	0.50	95	0	3900	0.05	70	0	3900	0.05
76	2	40	0.40	0.50	83	0	3900	0.06	64	0	3900	0.05
77	3	40	0.40	0.50	90	0	3900	0.07	87	0	3900	0.05
78	1	40	0.40	0.70	4	0	3900	0.03	0	0	3900	0.09
79	2	40	0.40	0.70	6	0	3900	0.04	0	0	3900	0.09
80	3	40	0.40	0.70	3	0	3900	0.03	0	0	3900	0.10
81	1	40	0.40	0.10	99361670	97158036	2331*	1800.05	86612308	84630301	2331*	1800.06
82	2	40	0.40	0.10	106284387	103479447	2136*	1800.06	84948065	82646491	2153*	1800.06
83	3	40	0.40	0.10	110818277	107553767	2020*	1800.06	82505277	79993765	2020*	1800.06
84	1	40	0.50	0.30	104460	0	3900	14.06	84193	0	3900	13.79
85	2	40	0.50	0.30	85813	0	3900	12.66	70309	0	3900	10.86
86	3	40	0.50	0.30	120121	0	3900	14.87	95654	0	3900	13.63
87	1	40	0.50	0.50	413	0	3900	0.15	341	0	3900	0.14
88	2	40	0.50	0.50	408	0	3900	0.15	323	0	3900	0.13
89	3	40	0.50	0.50	417	0	3900	0.14	365	0	3900	0.13
90	1	40	0.50	0.70	32	0	3900	0.03	0	0	3900	0.02
91	2	40	0.50	0.70	34	0	3900	0.03	0	0	3900	0.07
92	3	40	0.50	0.70	34	0	3900	0.03	0	0	3900	0.02
93	1	40	0.50	0.10	88079349	87091335	2189*	1800.06	68742563	67938025	2189*	1800.05
94	2	40	0.50	0.10	97094617	95932956	2091*	1800.06	74282784	73353599	2091*	1800.06
95	3	40	0.50	0.10	97992500	96788380	1974*	1800.06	70986374	70076758	1974*	1800.05
96	1	40	0.70	0.70	152	0	3900	0.06	102	0	3900	0.07
97	2	40	0.70	0.70	160	0	3900	0.07	110	0	3900	0.06
98	3	40	0.70	0.70	170	0	3900	0.07	109	0	3900	0.05
99	1	40	0.70	0.10	75978059	75518605	1856*	1800.06	41667667	41410714	1856*	1800.07
100	2	40	0.70	0.10	82549067	82058121	1998*	1800.06	45642647	45366593	2004*	1800.06
101	3	40	0.70	0.10	81840521	81282182	1754*	1800.06	41059496	40774949	1825*	1800.06
102	1	40	0.70	0.30	4947630	0	3900	762.21	940055	0	3900	302.77
103	2	40	0.70	0.30	4436388	0	3900	704.43	877947	0	3900	269.84
104	3	40	0.70	0.30	4966169	0	3900	734.22	940819	0	3900	284.52
105	1	40	0.70	0.50	4640	0	3900	1.34	2291	0	3900	0.90
106	2	40	0.70	0.50	4452	0	3900	1.26	2214	0	3900	0.84
107	3	40	0.70	0.50	4333	0	3900	1.15	2184	0	3900	0.81
108	1	50	0.40	0.10	59424733	58939824	-	1800	57246015	56762192	-	1800.01
109	2	50	0.40	0.10	65841550	65254221	-	1800.01	66227602	65614520	-	1800.01

Table B.5 . The effect of IT and Pr on BB-S using random dataset. (cont.)

Instances			IT			IT+StaticPr			IT+DynamicPr			
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
110	3	50	0.40	0.10	61764516	61230852	-	1800.01	61554402	61001894	-	1800.01
111	1	50	0.40	0.30	127891	0	4900	26.02	115395	0	4900	2.09
112	2	50	0.40	0.30	108192	0	4900	25.21	100379	0	4900	1.92
113	3	50	0.40	0.30	130368	0	4900	28.15	117741	0	4900	2.20
114	1	50	0.40	0.50	392	0	4900	0.20	338	0	4900	0.07
115	2	50	0.40	0.50	304	0	4900	0.14	235	0	4900	0.06
116	3	50	0.40	0.50	381	0	4900	0.19	343	0	4900	0.07
117	1	50	0.40	0.70	40	0	4900	0.03	0	0	4900	0.04
118	2	50	0.40	0.70	25	0	4900	0.03	0	0	4900	0.08
119	3	50	0.40	0.70	28	0	4900	0.03	0	0	4900	0.08
120	1	50	0.50	0.70	115	0	4900	0.05	4	0	4900	0.02
121	2	50	0.50	0.70	104	0	4900	0.06	0	0	4900	0.02
122	3	50	0.50	0.70	114	0	4900	0.07	10	0	4900	0.02
123	1	50	0.50	0.10	49125189	48887073	-	1800.02	44451861	44222165	-	1800.01
124	2	50	0.50	0.10	57653122	57323242	-	1800.01	52153868	51834054	-	1800.01
125	3	50	0.50	0.10	56705764	56416750	-	1800.01	50198580	49923821	-	1800.01
126	1	50	0.50	0.30	1778631	0	4900	436.34	870642	0	4900	284.37
127	2	50	0.50	0.30	1373205	0	4900	348.84	712308	0	4900	218.53
128	3	50	0.50	0.30	1661113	0	4900	406.46	815739	0	4900	251.43
129	1	50	0.50	0.50	2307	0	4900	0.86	1710	0	4900	0.75
130	2	50	0.50	0.50	2090	0	4900	0.73	1463	0	4900	0.59
131	3	50	0.50	0.50	2109	0	4900	0.79	1555	0	4900	0.67
132	1	50	0.70	0.30	20293442	19746809	-	1800.05	4712989	3980308	-	1800.01
133	2	50	0.70	0.30	28401793	27371141	-	1800.01	4891389	4078077	4900*	1800.07
134	3	50	0.70	0.30	28563459	27541148	-	1800	4828216	3995285	-	1800
135	1	50	0.70	0.50	22301	0	4900	10.66	7006	0	4900	5.12
136	2	50	0.70	0.50	22176	0	4900	10.10	6885	0	4900	5.04
137	3	50	0.70	0.50	23875	0	4900	10.77	7400	0	4900	5.53
138	1	50	0.70	0.70	348	0	4900	0.20	252	0	4900	0.18
139	2	50	0.70	0.70	356	0	4900	0.22	257	0	4900	0.19
140	3	50	0.70	0.70	358	0	4900	0.22	253	0	4900	0.20
141	1	50	0.70	0.10	36598409	36490428	-	1800.07	27677449	27591856	-	1800.01
142	2	50	0.70	0.10	27425353	27425353	2840*	1800.10	29049443	28955453	2840*	1800.09
143	3	50	0.70	0.10	41812587	41687542	-	1800.03	29767698	29674570	-	1800.01
144	1	75	0.40	0.30	11303578	10790192	-	1800	6491394	5757083	-	1800
145	2	75	0.40	0.30	10372074	9960466	-	1800.05	6628090	5820333	-	1800
146	3	75	0.40	0.30	11086775	10568420	7400*	1800.11	6476461	5664959	7400*	1800.10
147	1	75	0.40	0.50	11279	0	7400	8.44	6622	0	7400	5.99
148	2	75	0.40	0.50	10632	0	7400	8.88	6493	0	7400	6.33
149	3	75	0.40	0.50	11381	0	7400	8.60	6970	0	7400	6.30
150	1	75	0.40	0.70	263	0	7400	0.17	168	0	7400	0.11
151	2	75	0.40	0.70	274	0	7400	0.18	166	0	7400	0.12
152	3	75	0.40	0.70	274	0	7400	0.20	168	0	7400	0.12
153	1	75	0.40	0.10	18085656	18047942	-	1800.11	24177030	24122141	-	1800.02
154	2	75	0.40	0.10	18595758	18557208	-	1800.07	23285784	23233609	-	1800.01
155	3	75	0.40	0.10	19079968	19039316	-	1800.13	25640569	25581258	-	1800.02
156	1	75	0.50	0.30	13088306	13002501	-	1800.07	8921432	8804099	-	1800.01
157	2	75	0.50	0.30	14669533	14573096	-	1800.02	9217530	9105521	-	1800.02
158	3	75	0.50	0.30	10266427	10210700	-	1800.11	8835581	8726411	-	1800.12
159	1	75	0.50	0.50	50174	0	7400	56.62	17406	0	7400	28.61
160	2	75	0.50	0.50	52707	0	7400	61.67	17902	0	7400	29.03
161	3	75	0.50	0.50	53833	0	7400	62.50	17631	0	7400	30.33
162	1	75	0.50	0.70	557	0	7400	0.61	410	0	7400	0.52
163	2	75	0.50	0.70	575	0	7400	0.54	413	0	7400	0.49
164	3	75	0.50	0.70	597	0	7400	0.61	424	0	7400	0.49

Table B.5 . The effect of IT and Pr on BB-S using random dataset. (cont.)

Instances			IT			IT+StaticPr			IT+DynamicPr			
ID	Rep	$ V(G) $	Ω	Γ	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
165	1	75	0.50	0.10	19531733	19500984	-	1800.02	16970222	16941522	-	1800.03
166	2	75	0.50	0.10	21194334	21162693	-	1800.01	16492199	16465974	-	1800.02
167	3	75	0.50	0.10	19974174	19944581	-	1800.05	16932475	16905599	-	1800.05
168	1	75	0.70	0.70	1290	0	7400	3.80	642	0	7400	2.51
169	2	75	0.70	0.70	1313	0	7400	2.93	635	0	7400	2.15
170	3	75	0.70	0.70	1277	0	7400	3.09	629	0	7400	2.40
171	1	75	0.70	0.10	17749934	17733209	-	1800.11	11329350	11318394	-	1800.04
172	2	75	0.70	0.10	17986190	17969744	-	1800.08	11649293	11638361	-	1800.04
173	3	75	0.70	0.10	17125410	17109322	-	1800.05	11557611	11546488	-	1800.12
174	1	75	0.70	0.30	12230474	12206055	-	1800.04	4217592	4204489	-	1800.12
175	2	75	0.70	0.30	11332203	11309703	-	1800.01	4283792	4270557	-	1800.06
176	3	75	0.70	0.30	12349668	12324617	-	1800.05	4232782	4219198	-	1800.05
177	1	75	0.70	0.50	789174	0	7400	1171.54	39943	0	7400	221.06
178	2	75	0.70	0.50	787911	0	7400	1174.58	38884	0	7400	220.59
179	3	75	0.70	0.50	760234	0	7400	1147.69	39351	0	7400	213.91
180	1	100	0.40	0.70	805	0	9900	1.94	599	0	9900	1.30
181	2	100	0.40	0.70	795	0	9900	1.59	596	0	9900	1.26
182	3	100	0.40	0.70	790	0	9900	1.51	581	0	9900	1.24
183	1	100	0.40	0.10	11723941	11712932	-	1800.08	10294496	10284178	-	1800.14
184	2	100	0.40	0.10	12063238	12051290	-	1800.04	10434473	10423453	-	1800.15
185	3	100	0.40	0.10	11770650	11759671	-	1800.09	9946033	9936200	-	1800.07
186	1	100	0.40	0.30	8913932	8890515	-	1800.02	7661105	7633661	-	1800.03
187	2	100	0.40	0.30	8270827	8249490	-	1800.01	7744810	7717435	-	1800.02
188	3	100	0.40	0.30	8493859	8471675	-	1800.05	7577653	7550690	-	1800.06
189	1	100	0.40	0.50	107714	0	9900	308.96	37447	0	9900	118.71
190	2	100	0.40	0.50	105994	0	9900	257.54	35570	0	9900	116.36
191	3	100	0.40	0.50	109921	0	9900	269.67	36256	0	9900	115.84
192	1	100	0.50	0.50	577638	0	9900	1731.12	61605	0	9900	417.14
193	2	100	0.50	0.50	625626	95404	9900*	1800.14	62906	0	9900	417.07
194	3	100	0.50	0.50	608013	0	9900	1680.12	62790	0	9900	426.19
195	1	100	0.50	0.70	1290	0	9900	3.90	796	0	9900	3.27
196	2	100	0.50	0.70	1365	0	9900	3.97	799	0	9900	3.29
197	3	100	0.50	0.70	1338	0	9900	3.99	801	0	9900	3.34
198	1	100	0.50	0.10	10595495	10588315	-	1800.09	7344141	7338984	-	1800.03
199	2	100	0.50	0.10	11504616	11496578	-	1800.22	7773198	7767558	-	1800.26
200	3	100	0.50	0.10	11800813	11792499	-	1800.06	7188099	7182869	-	1800.29
201	1	100	0.50	0.30	8084581	8072008	-	1800.11	3974460	3965517	-	1800.02
202	2	100	0.50	0.30	8326725	8313642	-	1800.07	3979933	3970761	-	1800.09
203	3	100	0.50	0.30	8228250	8215478	-	1800.04	3908905	3900086	-	1800.16
204	1	100	0.70	0.30	7008246	7002275	-	1800.08	1614818	1613250	-	1800.91
205	2	100	0.70	0.30	7124770	7118707	-	1800.11	1735840	1734203	-	1800.85
206	3	100	0.70	0.30	6965523	6959743	-	1800.30	1736543	1734902	-	1800.13
207	1	100	0.70	0.50	4976159	4944361	9900*	1800.17	126146	111310	9900*	1800.14
208	2	100	0.70	0.50	5034575	5002247	9900*	1800.19	129915	114428	9900*	1800.18
209	3	100	0.70	0.50	4966790	4935307	-	1800.02	126465	111385	-	1800.03
210	1	100	0.70	0.70	15385	0	9900	52.10	1127	0	9900	15.23
211	2	100	0.70	0.70	15461	0	9900	53.12	1124	0	9900	14.90
212	3	100	0.70	0.70	14828	0	9900	49.81	1134	0	9900	14.65
213	1	100	0.70	0.10	9494762	9490382	-	1800.40	5365531	5363040	-	1800.22
214	2	100	0.70	0.10	8454097	8450382	-	1800.13	5345546	5343162	-	1800.39
215	3	100	0.70	0.10	8835010	8831105	-	1800.02	5468783	5466331	-	1800.21
Average			0.53	0.40	12919165	12167743	4158	635.25	8855387	8424177	4182	595.27
									12258155	11621741	4226	557.37

Table B.6. Branch-and-bound performance results: Zhang’s dataset.

Instances				BB-E				BB-S				Rec-D				BB-D			
ID	Type	$ V(G) $	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	2	100	0.06	0	0	7968	0.03	0	0	7968	0.18	60	56	7968	0.01	59	56	7968	0.01
1	2	100	0.06	0	0	8166	0.03	0	0	8166	0.22	55	54	8166	0.01	54	54	8166	0.01
2	1a	100	0.06	106325541	0	-	1594.33	43345816	37243022	-	1800	599827646	0	-	1800	114685817	0	-	1800
3	1b	100	0.06	6174477	2566028	4041	32.73	62983886	62401398	6449*	1800.14	2087281439	0	-	1800	1153409296	0	-	1800
4	2	100	0.06	0	0	7434	0.03	0	0	7434	0.04	55	54	7434	0.01	54	57	7434	0.01
5	1b	100	0.06	185183255	87810081	-	1800	47674531	46180247	-	1800	1292425085	0	-	1800	837024605	0	-	1800
6	1b	100	0.10	4796849	1253153	4275	29.47	38986207	38858762	8991*	1800.13	2459018191	2459015909	4657*	1800	1685686563	1685686477	4657*	1800
7	1a	100	0.10	646864	0	-	7.74	13420807	7757807	-	1800	860220741	0	-	1800	45517518	0	-	1800
8	2	100	0.10	10	0	12652	0.05	3	0	12652	0.09	1752	174	12652	0.57	256	189	12652	0.30
9	1b	100	0.10	227003024	113366969	-	1800	40982518	40674450	9768*	1800.14	1770051324	1746106407	7002*	1800	1210287562	1204239530	7002*	1800
10	2	100	0.10	0	0	11232	0.03	0	0	11232	0.03	70	70	11232	0.01	69	71	11232	0.01
11	2	100	0.10	0	0	11481	0.06	0	0	11481	0.03	62	62	11481	0.01	61	64	11481	0.01
12	1b	100	0.10	186780163	93212430	-	1800	42561695	42206966	-	1800.01	716772229	0	-	1800	307962967	0	-	1800
13	1a	100	0.10	135329213	65913405	-	1800	40506912	39975268	-	1800.01	952722070	0	-	1800	51880237	0	-	1800
14	2	200	0.02	0	0	17728	0.05	0	0	17728	0.04	159	150	17728	0.03	158	155	17728	0.03
15	2	200	0.02	0	0	18617	0.04	0	0	18617	0.04	124	115	18617	0.03	123	116	18617	0.03
16	2	200	0.02	0	0	19140	0.04	0	0	19140	0.05	116	103	19140	0.03	115	103	19140	0.03
17	1a	200	0.03	122214867	60910505	-	1800	22234016	22042639	-	1800	931656403	0	-	1800.01	502167028	0	-	1800
18	2	200	0.03	0	0	20716	0.04	0	0	20716	0.43	169	162	20716	0.04	168	164	20716	0.04
19	1a	200	0.03	75724736	25955792	-	1800	19275932	18386209	-	1800.01	640530878	0	-	1800	44516492	0	-	1800
20	2	200	0.03	0	0	18025	0.03	0	0	18025	0.36	131	127	18025	0.04	130	127	18025	0.04
21	2	200	0.03	0	0	20864	0.03	0	0	20864	0.31	118	110	20864	0.03	117	110	20864	0.03
22	1a	200	0.03	174492	0	-	3.93	8720143	6633320	-	1800	703334094	0	-	1800	31670501	0	-	1800
23	1a	200	0.04	14888177	0	-	306.72	16607089	16181585	-	1800.01	589373440	0	-	1800.01	16694857	0	-	1800
24	1a	200	0.04	112935899	56466279	-	1800.01	14009505	13961932	-	1800.01	1335924534	0	-	1800.02	666466248	0	-	1800
25	2	200	0.04	0	0	39895	0.03	0	0	39895	0.05	289	273	39895	0.05	288	275	39895	0.06
26	1a	200	0.04	91707408	45812436	-	1800	15562284	15477002	-	1800	250659712	0	-	1800.01	22824967	0	-	1800
27	2	200	0.04	0	0	37671	0.03	0	0	37671	0.05	116	116	37671	0.03	115	117	37671	0.04
28	2	200	0.04	0	0	38798	0.03	0	0	38798	0.03	114	111	38798	0.03	113	111	38798	0.04
29	1a	200	0.04	63063473	30268770	-	1800	17752412	17575248	-	1800.01	643463369	0	-	1800.01	21808143	0	-	1800
30	2	300	0.02	0	0	76345	0.02	0	0	76345	0.69	0	0	76345	0.01	0	0	76345	0.01
31	2	300	0.02	0	0	78880	0.03	0	0	78880	0.45	0	0	78880	0.01	0	0	78880	0.01
32	1a	300	0.02	639374	0	-	17.42	8718786	8304395	-	1800.01	492160166	0	-	1800.01	14605021	0	-	1800
33	1a	300	0.02	72843280	36367957	-	1800	14440773	14326203	-	1800.02	452618974	0	-	1800.02	121814790	0	-	1800
34	2	300	0.02	0	0	71562	0.02	0	0	71562	0.05	178	177	71562	0.08	177	177	71562	0.09
35	1a	300	0.02	44963586	19792754	-	1800	13858411	13646818	-	1800.01	457727873	0	-	1800.01	16574870	0	-	1800
36	2	300	0.01	0	0	43721	0.03	0	0	43721	0.45	0	0	43721	0.00	0	0	43721	0.00
37	2	300	0.01	0	0	44267	0.03	0	0	44267	0.04	178	170	44267	0.08	177	172	44267	0.08
38	2	300	0.01	0	0	43071	0.04	0	0	43071	0.45	1517	1499	43071	0.32	1516	1499	43071	0.38
39	1a	300	0.02	40076081	0	-	1347.47	11919655	11256212	-	1800.01	350196319	0	-	1800.01	44590590	0	-	1800
40	2	300	0.02	0	0	43125	0.02	0	0	43125	0.47	180	159	43125	0.08	179	160	43125	0.09
41	2	300	0.02	0	0	42292	0.02	0	0	42292	0.41	178	175	42292	0.08	177	175	42292	0.08
42	2	300	0.02	0	0	44114	0.02	0	0	44114	0.42	173	170	44114	0.07	172	170	44114	0.07
43	1b	50	0.16	360	43	708	0.10	159621923	151866627	797*	1800.07	558283820	558283813	708	174.84	558283812	558283805	708	222.25
44	2	50	0.16	7	2	1636	0.10	3	2	1636	0.15	469	388	1636	0.04	406	394	1636	0.04
45	1b	50	0.16	1367	304	770	0.10	148405965	143350778	803*	1800.07	2066034854	2066026628	770	738.35	2066026627	2066026614	770	959.44
46	2	50	0.16	5	2	2043	0.09	2	3	2043	0.11	49	47	2043	0.01	54	55	2043	0.01
47	2	50	0.16	2	0	2338	0.03	1	0	2338	0.04	70	69	2338	0.00	68	68	2338	0.00
48	1b	50	0.16	27361	8676	917	0.33	119406960	114373134	1072*	1800.07	1138811215	1123952107	917	598.71	1123952106	1123952095	917	705.16
49	1b	50	0.16	7582196	2756649	1324	46.71	100620893	92510270	1457*	1800.07	2114327056	2055312629	1324*	1800	1854189598	1853987085	1324*	1800
Average			0.06	29981641	12849245	24116	463.76	20432343	19503806	23918	828.13	469268556	200174042	24217	750.67	250252900	169843605	24217	757.77

Table B.7. Branch-and-bound performance results: Carrabs' dataset.

Instances				BB-E			BB-S			Rec-D			BB-D				
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	
1	1	100	0.30	0.01	8239537	851103	1079	103.31	14676390	14663146	1740*	1800.19	1800.19	2489745870	2489743390	1099*	1800
2	2	100	0.30	0.01	1122463	369730	1056	9.41	16942013	16926863	1722*	1800.14	1800.14	2388395363	2388393148	1072*	1800
3	3	100	0.30	0.01	967389	249026	1059	9.42	16622077	16607557	1673*	1800.22	1800.22	2583012628	2583010471	1073*	1800
4	4	100	0.30	0.01	1831517	597880	1046	14.65	15861398	15848335	1632*	1800.20	1800.20	2872903473	2872900625	1078*	1800
5	5	100	0.30	0.01	24994346	7266003	1072	196.03	14138231	14126582	1910*	1800.26	1800.26	2030503793	2030501036	1088*	1800
6	1	100	0.30	0.04	75967512	37983758	-	1800.07	16419592	16399429	-	1800.03	1800.03	384523415	0	-	1800.03
7	2	100	0.30	0.04	78450102	39225053	-	1800.24	15611143	15592788	1994*	1800.14	1800.14	364225221	0	-	1800.03
8	3	100	0.30	0.04	107290132	53645068	-	1800.01	16074579	16055175	-	1800.10	1800.10	382420962	0	-	1800.03
9	4	100	0.30	0.04	103513410	51756707	-	1800.01	14383290	14367330	-	1800.01	1800.01	386375952	0	-	1800.03
10	5	100	0.30	0.04	105639636	52819820	-	1800	15929855	15911960	1916*	1800.16	1800.16	354863370	0	-	1800.03
11	1	100	0.30	0.07	86783840	43391918	-	1800	13927152	13909675	-	1800.08	1800.08	457411016	0	-	1800.02
12	2	100	0.30	0.07	85358652	42679314	-	1800	14108682	14089681	-	1800.03	1800.03	462300661	0	-	1800.03
13	3	100	0.30	0.07	80915087	40457544	-	1800	14163065	14144998	-	1800.08	1800.08	476148577	0	-	1800.03
14	4	100	0.30	0.07	84144295	42072149	-	1800	13665454	13647740	-	1800.10	1800.10	396857700	0	-	1800.03
15	5	100	0.30	0.07	85813376	42906688	-	1800	14661120	14642001	-	1800.01	1800.01	426188483	0	-	1800.03
16	1	100	0.40	0.07	76796230	38398117	-	1800	12011545	12004143	-	1800.13	1800.13	430088096	0	-	1800.04
17	2	100	0.40	0.07	75067228	37533616	-	1800	11808762	11798530	-	1800.08	1800.08	380693351	0	-	1800.04
18	3	100	0.40	0.07	76447858	38223931	-	1800	12670647	12659714	-	1800.07	1800.07	401683081	0	-	1800.05
19	4	100	0.40	0.07	78157368	39078686	-	1800	12366118	12355046	-	1800.03	1800.03	363883754	0	-	1800.04
20	5	100	0.40	0.07	76831004	38415504	-	1800	12068849	12058606	-	1800.16	1800.16	384212609	0	-	1800.04
21	1	100	0.40	0.01	53979671	26989837	1024	360.39	13360082	13351222	1643*	1800.18	1800.18	2625802572	2625799440	1047*	1800
22	2	100	0.40	0.01	178707	33428	1025	2.49	14065475	14056730	1711*	1800.24	1800.24	2908313976	2908311917	1038*	1800
23	3	100	0.40	0.01	3870910	421734	1028	37.58	14444792	14435002	1681*	1800.19	1800.19	2243035106	2243032483	1048*	1800
24	4	100	0.40	0.01	45174278	4988926	1031	431.02	13402234	13393355	1577*	1800.14	1800.14	2513234041	2513231763	1047*	1800
25	5	100	0.40	0.01	11092279	3817452	1036	88.95	14656148	14647646	1792*	1800.33	1800.33	2880102543	2880100485	1050*	1800
26	1	100	0.40	0.04	101692580	50846292	-	1800	13780065	13768654	1883*	1800.28	1800.28	323198443	0	-	1800.05
27	2	100	0.40	0.04	102100026	51050015	-	1800.01	14581369	14568497	1783*	1800.22	1800.22	350683779	286187572	1356*	1800
28	3	100	0.40	0.04	105265636	52632820	-	1800.01	13593541	13581507	1809*	1800.33	1800.33	321141704	0	-	1800.05
29	4	100	0.40	0.04	101548842	50774423	-	1800	13751619	13739054	1768*	1800.26	1800.26	304808580	0	-	1800.05
30	5	100	0.40	0.04	104021222	52010613	-	1800	14490203	14477990	1858*	1800.25	1800.25	333946582	0	-	1800.05
31	1	100	0.20	0.04	143769686	71884234	-	1800.01	26221528	26162143	-	1800.01	1800.01	296420678	0	-	1800.01
32	2	100	0.20	0.04	140558964	70278934	-	1800	24691915	24637536	-	1800.02	1800.02	338878551	0	-	1800.01
33	3	100	0.20	0.04	143373646	71686741	-	1800.01	22979439	22929416	-	1800.04	1800.04	267854068	0	-	1800.01
34	4	100	0.20	0.04	143898953	71948871	-	1800	24696109	24643620	-	1800.01	1800.01	254125600	0	-	1800.01
35	5	100	0.20	0.04	142419829	71209876	-	1800	23197389	23145283	-	1800.04	1800.04	275114643	0	-	1800.01
36	1	100	0.20	0.07	97364615	48651127	-	1800	21374387	21320463	-	1800.04	1800.04	510948537	0	-	1800.01
37	2	100	0.20	0.07	96364458	48178144	-	1800	21201929	21144792	-	1800.02	1800.02	541282649	0	-	1800.01
38	3	100	0.20	0.07	97670779	48832681	-	1800.01	21733667	21675656	-	1800.03	1800.03	525836151	0	-	1800.01
39	4	100	0.20	0.07	98839155	49415409	-	1800	22464147	22405591	-	1800.03	1800.03	564521462	0	-	1800.01
40	5	100	0.20	0.07	97539761	48764610	-	1800	20195516	20144962	-	1800.03	1800.03	510954814	0	-	1800.01
41	1	100	0.20	0.01	99391419	41411622	1119	620.00	20563098	20534033	1808*	1800.16	1800.16	2969379532	2969379532	1144*	1800
42	2	100	0.20	0.01	1427890	572342	1137	10.08	21891011	21860868	1672*	1800.19	1800.19	2533096686	2533094543	1169*	1800
43	3	100	0.20	0.01	28025280	11877074	1113	162.02	19812930	19787472	1741*	1800.18	1800.18	2639045243	2639041489	1159*	1800
44	4	100	0.20	0.01	2483088	431247	1110	18.33	21908666	21876410	1755*	1800.17	1800.17	2404478196	2404475948	1131*	1800
45	5	100	0.20	0.01	45501124	15599270	1090	291.65	22716489	22684278	1752*	1800.16	1800.16	2584412432	2584411719	1125*	1800
46	1	25	0.40	0.04	134	62	321	0.07	125815503	24740957	321*	1800.04	1800.04	12687	12681	321	0.01

Table B.7 . Branch-and-bound performance results: Carrabs’ dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
47	2	25	0.40	0.04	81	26	317	0.07	62443438	22549	317	731.36	7233	7222	317	0.01	7221	7221	317	0.01
48	3	25	0.40	0.04	30	14	284	0.07	567889	3736	284	11.24	379	379	284	0.00	378	378	284	0.00
49	4	25	0.40	0.04	129	49	311	0.07	6791181	35874	311	149.55	59189	59182	311	0.04	59181	59174	311	0.06
50	5	25	0.40	0.04	6	4	290	0.07	215468	21328	290	6.81	681	681	290	0.00	680	680	290	0.00
51	1	25	0.40	0.07	894	265	329	0.07	2357096	563459	329	70.62	223359	222925	329	0.21	222924	222915	329	0.26
52	2	25	0.40	0.07	1584	542	339	0.08	7978369	72989	339	249.56	20303	19918	339	0.02	19917	19912	339	0.03
53	3	25	0.40	0.07	1814	686	368	0.08	22181383	3612514	368	531.02	25775	25364	368	0.02	25392	25351	368	0.03
54	4	25	0.40	0.07	1211	514	311	0.07	667937	79153	311	24.79	15510	15342	311	0.02	15341	15337	311	0.02
55	5	25	0.40	0.07	771	207	321	0.07	290448	36808	321	11.02	8007	7963	321	0.01	7962	7959	321	0.01
56	1	25	0.40	0.01	0	0	282	0.01	160241459	56624796	285*	1800.03	669	668	282	0.00	667	666	282	0.00
57	2	25	0.40	0.01	10	4	294	0.06	284322983	249295465	296*	1800.04	83605	83603	294	0.04	83602	83600	294	0.06
58	3	25	0.40	0.01	8	0	284	0.02	248800217	166521407	284*	1800.04	67	67	284	0.00	66	66	284	0.00
59	4	25	0.40	0.01	2	0	281	0.02	176356482	106063791	281*	1800.04	89538	89537	281	0.04	89536	89535	281	0.06
60	5	25	0.40	0.01	4	3	292	0.06	135815137	10367214	292*	1800.04	101	101	292	0.00	100	100	292	0.00
61	1	25	0.20	0.07	59	10	385	0.06	2595	358	385	0.12	1500	1417	385	0.00	1446	1416	385	0.00
62	2	25	0.20	0.07	51	17	432	0.06	764	82	432	0.09	1119	1069	432	0.00	1072	1070	432	0.00
63	3	25	0.20	0.07	380	133	458	0.07	4855	266	458	0.18	8681	5349	458	0.01	7715	5284	458	0.01
64	4	25	0.20	0.07	258	107	400	0.06	11062	1771	400	0.37	2413	2399	400	0.00	2403	2399	400	0.00
65	5	25	0.20	0.07	223	42	420	0.06	1708	528	420	0.19	3820	3279	420	0.01	3453	3273	420	0.01
66	1	25	0.20	0.01	2	2	347	0.06	10031200	265487	347	105.88	4759	4759	347	0.00	4758	4759	347	0.00
67	2	25	0.20	0.01	0	0	353	0.02	4192064	354	353	27.24	2755	2755	353	0.00	2754	2755	353	0.00
68	3	25	0.20	0.01	0	0	346	0.02	283132	3990	346	3.59	62	62	346	0.00	61	61	346	0.00
69	4	25	0.20	0.01	6	3	336	0.06	1730982	235759	336	16.79	1453	1452	336	0.00	1451	1450	336	0.00
70	5	25	0.20	0.01	0	0	389	0.02	2088532	936	389	12.37	13540	13540	389	0.01	13539	13540	389	0.01
71	1	25	0.20	0.04	194	62	381	0.06	35255	3911	381	0.96	3439	3411	381	0.00	3410	3402	381	0.00
72	2	25	0.20	0.04	126	23	390	0.06	23621	5917	390	0.71	16289	4744	390	0.01	13713	4737	390	0.02
73	3	25	0.20	0.04	11	5	372	0.06	7369	286	372	0.18	1498	1498	372	0.00	1497	1498	372	0.00
74	4	25	0.20	0.04	22	4	357	0.06	95222	7274	357	1.46	675	660	357	0.00	659	655	357	0.00
75	5	25	0.20	0.04	44	19	406	0.06	9969	140	406	0.33	2872	2870	406	0.00	2869	2868	406	0.00
76	1	25	0.30	0.04	122	23	305	0.06	786492	142516	305	15.28	19539	19355	305	0.01	19354	19349	305	0.02
77	2	25	0.30	0.04	28	9	339	0.06	479621	1543	339	11.09	3479	3475	339	0.00	3476	3472	339	0.00
78	3	25	0.30	0.04	143	62	344	0.06	3618374	141515	344	81.65	2104	2103	344	0.00	2102	2101	344	0.00
79	4	25	0.30	0.04	123	22	329	0.06	56592485	2860727	329	1127.70	241078	240476	329	0.19	240475	240472	329	0.24
80	5	25	0.30	0.04	113	50	326	0.07	4540615	10983	326	60.07	107913	107830	326	0.09	107829	107819	326	0.12
81	1	25	0.30	0.07	106	34	349	0.06	16533	6031	349	0.68	5457	5397	349	0.01	5397	5395	349	0.01
82	2	25	0.30	0.07	1296	235	385	0.07	206391	6329	385	4.25	65312	62824	385	0.09	62991	62818	385	0.10
83	3	25	0.30	0.07	719	287	335	0.07	258752	46855	335	5.67	13864	13842	335	0.01	13842	13836	335	0.02
84	4	25	0.30	0.07	1982	601	348	0.08	286614	22719	348	6.31	135382	134312	348	0.14	134327	134307	348	0.16
85	5	25	0.30	0.07	487	188	357	0.07	650090	74887	357	12.49	11910	11490	357	0.01	11489	11489	357	0.01
86	1	25	0.30	0.01	13	8	299	0.06	300000000	242525458	315	1797.35	35458	35457	299	0.02	35456	35455	299	0.03
87	2	25	0.30	0.01	3	2	297	0.06	179638858	50852358	297*	1800.04	1866	1865	297	0.00	1864	1863	297	0.00
88	3	25	0.30	0.01	3	2	318	0.06	213116798	60595616	318*	1800.03	1278	1277	318	0.00	1276	1275	318	0.00
89	4	25	0.30	0.01	16	6	311	0.06	280170302	257893017	331*	1800.03	2234	2234	311	0.00	2233	2233	311	0.00
90	5	25	0.30	0.01	0	0	306	0.02	36498635	424	306	437.62	182	182	306	0.00	181	182	306	0.00
91	1	50	0.20	0.04	385704	174709	678	2.37	104746353	102591521	738*	1800.07	52350361	49148003	678	30.04	50855526	49147989	678	34.46
92	2	50	0.20	0.04	708300	223110	681	4.20	107746325	105267894	733*	1800.06	2868027580	2852973553	716*	1800	2253641203	2253641170	716*	1800

Table B.7 . Branch-and-bound performance results: Carrabs’ dataset. (cont.)

Instances				BB-E			BB-S			Rec-D			BB-D			
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
93	3	50	0.20	0.04	1950911	597423	709	11.09	98690682	96123083	768*	1800.07	2673010605	2649331852	713*	1800
94	4	50	0.20	0.04	53849	25273	639	0.54	110097014	106362784	681*	1800.07	92496504	91903968	639	48.06
95	5	50	0.20	0.04	2164033	844872	681	10.81	101350682	99280278	752*	1800.06	1032258004	1026539828	681	723.99
96	1	50	0.20	0.07	228595800	95943839	-	1800	96117210	90873758	891*	1800.07	1257872057	899785259	816*	1800
97	2	50	0.20	0.07	187983611	75808494	835	1406.03	93113386	87292691	-	1800.01	1225176282	0	-	1800
98	3	50	0.20	0.07	217550842	84526055	-	1800	98112450	91145841	910*	1800.08	1079847895	0	-	1800
99	4	50	0.20	0.07	833922233	27413840	832	652.29	81698365	72587635	908*	1800.07	1259495133	0	-	1800
100	5	50	0.20	0.07	9186112	2934366	769	66.30	101110158	95744022	839*	1800.07	1355263316	0	-	1800
101	1	50	0.20	0.01	100	47	619	0.10	179955154	166294812	636*	1800.37	967049217	967048784	622*	1800
102	2	50	0.20	0.01	56	26	604	0.10	115633345	67690310	608*	1800.07	888449	888446	604	0.69
103	3	50	0.20	0.01	24	10	634	0.10	127208910	125801717	866*	1800.08	50914	50908	634	0.04
104	4	50	0.20	0.01	228	115	616	0.10	184687110	178500540	707*	1800.36	106344015	106344004	616	36.35
105	5	50	0.20	0.01	263	121	595	0.10	177432516	168657162	668*	1800.83	18061178	18061175	595	6.07
106	1	50	0.30	0.04	2329284	1164502	612	10.94	82227208	81520703	742*	1800.07	3382157468	3366848512	632*	1800
107	2	50	0.30	0.04	2201101	1058704	615	10.17	89356019	88523707	713*	1800.08	3341288988	3340490136	615*	1800
108	3	50	0.30	0.04	189622	80207	587	1.35	85801093	84928673	681*	1800.07	2817889005	2782021713	603*	1800
109	4	50	0.30	0.04	9951061	4046634	634	50.11	87094551	86363425	726*	1800.08	2995707589	2973475478	635*	1800
110	5	50	0.30	0.04	357673	170968	643	2.12	88102486	87234581	707*	1800.08	3402912148	3398626848	643*	1800
111	1	50	0.30	0.07	239699966	119309354	-	1800.01	88111229	86943249	861*	1800.07	1082046676	866285508	774*	1800
112	2	50	0.30	0.07	240500765	119792806	-	1800.01	89223132	87839915	874*	1800.07	1250568769	1147107311	782*	1800
113	3	50	0.30	0.07	250000000	124793932	-	1763.24	89098973	87991211	857*	1800.08	820354216	525241416	867*	1800
114	4	50	0.30	0.07	250000001	124624978	-	1752.38	94285045	83182376	810*	1800.06	1190245212	964264905	751*	1800
115	5	50	0.30	0.07	250000001	124591356	-	1766.00	80285647	89051469	858*	1800.06	894739013	0	-	1800
116	1	50	0.30	0.01	48	10	570	0.13	97527269	96388875	609*	1800.06	1223797597	1223797471	570*	1800
117	2	50	0.30	0.01	100	21	561	0.11	105378420	101135026	571*	1800.07	853473861	853473514	565*	1800
118	3	50	0.30	0.01	447	225	573	0.12	105244656	104629466	691*	1800.07	787288	787286	573	0.57
119	4	50	0.30	0.01	63	28	560	0.12	157793075	155774149	636*	1800.49	7409056	7409050	560	3.16
120	5	50	0.30	0.01	41	19	549	0.13	161292604	157549198	577*	1800.21	23722200	23722198	549	7.69
121	1	50	0.40	0.01	50	0	548	0.03	133153999	131149100	611*	1800.07	1283970669	1283970668	548	432.99
122	2	50	0.40	0.01	26	6	530	0.14	127250028	110964920	541*	1800.07	3721612446	3721612097	533*	1800
123	3	50	0.40	0.01	35	19	549	0.14	84771308	84401607	660*	1800.09	493782967	493782648	558*	1800
124	4	50	0.40	0.01	169	86	540	0.11	94295978	93515614	634*	1800.07	984403371	984403368	540	308.32
125	5	50	0.40	0.01	40	21	540	0.11	85460230	85090812	647*	1800.07	407285199	407285196	540	128.95
126	1	50	0.40	0.04	398239	68611	594	3.64	70319253	69971760	698*	1800.07	2954430107	2954053631	598*	1800
127	2	50	0.40	0.04	5296289	1079106	579	33.37	68590771	68218647	690*	1800.07	2989169089	2987799289	582*	1800
128	3	50	0.40	0.04	686429	343216	589	3.70	69209735	68747218	709*	1800.08	1318744290	1318603689	589	783.05
129	4	50	0.40	0.04	365683	148220	577	2.45	70277997	69964716	682*	1800.07	3400392650	3399083052	577*	1800
130	5	50	0.40	0.04	1538134	763628	592	7.51	70551689	70117619	716*	1800.07	2487179679	2487088501	592	1549.37
131	1	50	0.40	0.07	233927316	116938607	-	1800	72264511	71812138	819*	1800.06	935565223	766157045	710*	1800
132	2	50	0.40	0.07	242877194	121405515	-	1800	68016362	67553841	816*	1800.08	1058795577	977020425	699*	1800
133	3	50	0.40	0.07	238946947	119456695	-	1800	69755748	69317780	854*	1800.08	1058031402	967531919	727*	1800
134	4	50	0.40	0.07	235216850	117587408	-	1800	71889683	71408468	851*	1800.08	1143825305	1077125719	724*	1800
135	5	50	0.40	0.07	240089287	119969567	-	1800	72223504	71762566	775*	1800.08	1053783751	951137242	698*	1800
136	1	75	0.40	0.04	170623424	85311714	-	1800	26868427	26817695	1195*	1800.09	722388838	707230528	940*	1800
137	2	75	0.40	0.04	173152028	86576016	-	1800.01	28135333	28087449	1261*	1800.06	1041208732	1027001072	933*	1800
138	3	75	0.40	0.04	167777768	83888886	-	1800	28384511	28334093	1193*	1800.04	1028565958	1020224793	978*	1800

Table B.7 . Branch-and-bound performance results: Carrabs’ dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best_UB	CPU(s)	Explored	Active	Best_UB	CPU(s)	Explored	Active	Best_UB	CPU(s)	Explored	Active	Best_UB	CPU(s)
139	4	75	0.40	0.04	108574230	84287117	-	1800	29403387	29351744	1249*	1800.03	916734480	904959004	950*	1800	637674450	637674121	950*	1800
140	5	75	0.40	0.04	171009120	85504562	-	1800	26874615	26827760	1246*	1800.06	761088839	759692387	955*	1800	506491554	506491445	960*	1800
141	1	75	0.40	0.07	112856994	56428497	-	1800	27607013	27550882	-	1800.01	515679280	0	-	1800.01	37169308	0	-	1800
142	2	75	0.40	0.07	115512729	57756363	-	1800	24861096	24813747	-	1800.02	565143613	0	-	1800.01	43351037	0	-	1800
143	3	75	0.40	0.07	114150435	5775208	-	1800	25136631	25086139	-	1800.02	400978559	0	-	1800.01	49487580	0	-	1800
144	4	75	0.40	0.07	112217472	56108731	-	1800	25419029	25372485	-	1800.04	524415037	0	-	1800.01	43368221	0	-	1800
145	5	75	0.40	0.07	112680454	56340225	-	1800	27428612	27373766	-	1800.03	501651941	0	-	1800.01	49560140	0	-	1800
146	1	75	0.40	0.01	2102	1052	787	0.18	33759421	33692515	1213*	1800.05	763106405	763105570	794*	1800	3461137063	3461136995	794*	1800
147	2	75	0.40	0.01	6451	366	785	0.25	28817247	28781656	1188*	1800.04	3358403680	3358402757	794*	1800	2071953063	2071953000	794*	1800
148	3	75	0.40	0.01	53574	14270	783	0.79	30268489	30232280	1194*	1800.05	2934186616	2934185220	794*	1800	1886033979	1886033921	794*	1800
149	4	75	0.40	0.01	3500	1751	784	0.18	28381078	282346937	1215*	1800.07	3383137280	3383136627	788*	1800	1422571114	1422571052	788*	1800
150	5	75	0.40	0.01	17203	6083	797	0.35	30449628	30413976	1271*	1800.08	2728121895	2728120589	810*	1800	1672330498	1672330427	810*	1800
151	1	75	0.20	0.07	156874059	78340998	-	1800	47444288	47064795	-	1800	691544118	0	-	1800	50374032	0	-	1800
152	2	75	0.20	0.07	159059437	79199647	-	1800	40142062	39877212	-	1800	840639490	0	-	1800	54091248	0	-	1800
153	3	75	0.20	0.07	154301771	77909654	-	1800	46071187	45727335	-	1800	725803264	0	-	1800	65672684	0	-	1800
154	4	75	0.20	0.07	152799560	76145120	-	1800	48593328	48188749	-	1800.01	692253571	0	-	1800	55849821	0	-	1800
155	5	75	0.20	0.07	153293400	76391421	-	1800	42481821	42168798	-	1800.01	7224445653	0	-	1800	57121635	0	-	1800
156	1	75	0.20	0.01	11161	4693	868	0.27	39397025	39290331	1240*	1800.02	3101064334	3101063009	896*	1800	1999030310	1999030244	896*	1800
157	2	75	0.20	0.01	6060	1003	871	0.21	42620722	42498194	1155*	1800.04	3631195441	3631194486	885*	1800	2522883761	375400046	885*	1800
158	3	75	0.20	0.01	26028	13015	838	0.37	44034016	43914544	1215*	1800.04	121779039	121778471	844*	1800	3259921403	1112437690	844*	1800
159	4	75	0.20	0.01	1329	629	855	0.21	46819717	46681917	1156*	1800.03	3571046062	3571045218	863*	1800	2537280650	384680065	863*	1800
160	5	75	0.20	0.01	3963	1021	857	0.19	42345967	42323996	1253*	1800.03	3574167549	3574166847	860*	1800	2537280650	389796939	860*	1800
161	1	75	0.20	0.04	196943929	98457200	-	1800	46525735	46286338	1353*	1800.03	1271121199	33015343	1164*	1800	582808366	432244091	1160*	1800
162	2	75	0.20	0.04	201886303	100939781	-	1800.01	49114081	48886425	1362*	1800.03	700463855	583164340	1200*	1800	459449007	358059439	1139*	1800
163	3	75	0.20	0.04	197208650	98597382	-	1800	43373028	43169611	1326*	1800.04	926928635	874266104	1134*	1800	463019646	391128648	1200*	1800
164	4	75	0.20	0.04	2045116638	102238656	-	1800	49915813	49654177	1220*	1800.04	1111341263	0	-	1800	498303422	0	-	1800
165	5	75	0.20	0.04	202349864	101168063	-	1800	48555970	48279504	1321*	1800.04	926928635	874266104	1134*	1800	631758401	631397580	1134*	1800
166	1	75	0.30	0.04	192614316	96340703	-	1800	36159733	36037969	1159*	1800.02	724134279	250556517	1044*	1800	360238861	273391200	1044*	1800
167	2	75	0.30	0.04	198696615	99484803	-	1800	32805050	32724547	1292*	1800.03	730587280	712338242	1045*	1800	560541874	560541528	1057*	1800
168	3	75	0.30	0.04	187258121	936290015	-	1800	35651467	35552800	1236*	1800.04	731449266	0	-	1800.01	328628137	0	-	1800
169	4	75	0.30	0.04	194917291	97458590	-	1800	34602217	34374699	1273*	1800.06	767956778	755939379	999*	1800	575422918	575408861	999*	1800
170	5	75	0.30	0.04	191070253	95535127	-	1800.01	35665384	35518702	1185*	1800.05	620738468	599410088	1042*	1800	484917403	482241501	1042*	1800
171	1	75	0.30	0.07	141152283	70572907	-	1800	34782669	34681293	1187*	1800.01	526351821	0	-	1800.01	51524466	0	-	1800
172	2	75	0.30	0.07	139685124	69840877	-	1800	31305320	31204008	-	1800.01	553121062	0	-	1800.01	56311375	0	-	1800
173	3	75	0.30	0.07	139401839	69699198	-	1800	33888436	33783497	-	1800.01	564287256	0	-	1800.01	40432946	0	-	1800
174	4	75	0.30	0.07	139713544	69855090	-	1800	32942443	32844493	-	1800.01	498174259	0	-	1800.01	65496371	0	-	1800
175	5	75	0.30	0.07	139674864	69834910	-	1800	32514471	32422994	-	1800.01	468797461	0	-	1800.01	48427124	0	-	1800
176	1	75	0.30	0.01	11571	1684	798	0.22	33404903	33347929	1154*	1800.04	3348945574	3348944452	811*	1800	2129944141	2129944079	812*	1800
177	2	75	0.30	0.01	16476	7081	821	0.26	32001648	3194086	1182*	1800.03	3136953352	3136952505	825*	1800	2142461332	2142461269	825*	1800
178	3	75	0.30	0.01	18941	27430	816	0.29	42332985	42243375	1124*	1800.07	3940732517	3940731875	821*	1800	3243388994	3243388934	821*	1800
179	4	75	0.30	0.01	9205	3615	820	0.19	32230446	32174930	1254*	1800.04	3801904346	3801903508	826*	1800	2780307201	2780307137	826*	1800
180	5	75	0.30	0.01	359343	115639	815	2.62	31629683	31580963	1340*	1800.05	3597784708	3597783898	817*	1800	2567161208	2567161149	817*	1800
Average			0.30	0.04	631119369	308237664	594	745.15	51175386	44107026	876	1470.72	998172123	877126206	653	1222.56	745542920	665727067	667	1226.22

Table B.8. Branch-and-bound performance results: random dataset.

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	Γ	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
0	1	20	0.60	0.20	3879	478	1133	0.50	2768	761	1133	0.68	81258	18789	1133	2.45	22365	18329	1133	0.99
1	1	20	0.60	0.30	57	0	1883	0.04	18	0	1883	0.04	2159	1	1883	0.08	222	1	1883	0.03
2	1	20	0.60	0.40	29	0	1900	0.03	0	0	1900	0.03	91	0	1900	0.01	31	0	1900	0.00
3	1	20	0.60	0.50	0	0	1900	0.03	0	0	1900	0.04	71	0	1900	0.00	3	0	1900	0.00
4	1	20	0.60	0.60	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
5	1	20	0.60	0.70	0	0	1900	0.08	0	0	1900	0.08	0	0	1900	0.00	0	0	1900	0.00
6	1	20	0.60	0.80	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
7	1	20	0.60	0.10	1081	68	833	0.08	660738	84701	833	19.22	23930	23153	833	0.03	23152	23140	833	0.04
8	1	20	0.70	0.20	15297	2739	1018	1.92	16683	5499	1018	3.35	64867	43624	1018	2.55	47741	45196	1018	1.89
9	1	20	0.70	0.30	241	0	1900	0.07	57	0	1900	0.07	12050	0	1900	0.44	1010	0	1900	0.10
10	1	20	0.70	0.40	42	0	1900	0.03	3	0	1900	0.03	460	0	1900	0.02	99	0	1900	0.01
11	1	20	0.70	0.50	31	0	1900	0.03	0	0	1900	0.03	110	0	1900	0.01	21	0	1900	0.01
12	1	20	0.70	0.60	0	0	1900	0.03	0	0	1900	0.03	58	0	1900	0.00	0	0	1900	0.00
13	1	20	0.70	0.70	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
14	1	20	0.70	0.80	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
15	1	20	0.70	0.10	799	93	750	0.07	488812	11007	750	18.98	13821	13537	750	0.02	13540	13527	750	0.01
16	1	20	0.80	0.10	781	373	717	0.06	7088165	617356	717	291.33	8050	8047	717	0.01	8046	8043	717	0.01
17	1	20	0.80	0.20	42996	10646	924	3.70	52690	12648	924	12.12	180111	143101	924	7.00	151648	145350	924	6.01
18	1	20	0.80	0.30	819	0	1900	0.20	161	0	1900	0.16	61453	0	1900	1.90	2767	0	1900	0.36
19	1	20	0.80	0.40	59	0	1900	0.04	14	0	1900	0.05	2301	0	1900	0.07	218	0	1900	0.04
20	1	20	0.80	0.50	56	0	1900	0.03	1	0	1900	0.03	148	0	1900	0.01	46	0	1900	0.01
21	1	20	0.80	0.60	1	0	1900	0.03	0	0	1900	0.03	118	0	1900	0.00	12	0	1900	0.00
22	1	20	0.80	0.70	0	0	1900	0.03	0	0	1900	0.03	0	0	1900	0.00	0	0	1900	0.00
23	1	20	0.80	0.80	0	0	1900	0.06	0	0	1900	0.08	0	0	1900	0.00	0	0	1900	0.00
24	1	20	0.20	0.20	0	0	1900	0.03	0	0	1900	0.03	0	0	1900	0.00	0	0	1900	0.00
25	1	20	0.20	0.30	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
26	1	20	0.20	0.40	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
27	1	20	0.20	0.50	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
28	1	20	0.20	0.60	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
29	1	20	0.20	0.70	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
30	1	20	0.20	0.10	23	3	1792	0.06	17	0	1792	0.03	240	21	1792	0.00	133	21	1792	0.00
31	1	20	0.30	0.70	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
32	1	20	0.30	0.80	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
33	1	20	0.30	0.10	126	44	1307	0.06	837	283	1307	0.09	413	363	1307	0.00	366	363	1307	0.00
34	1	20	0.30	0.20	4	0	1900	0.03	2	0	1900	0.03	25	0	1900	0.00	7	0	1900	0.00
35	1	20	0.30	0.30	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
36	1	20	0.30	0.40	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
37	1	20	0.30	0.50	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
38	1	20	0.30	0.60	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
39	1	20	0.40	0.40	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
40	1	20	0.40	0.50	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
41	1	20	0.40	0.60	0	0	1900	0.07	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
42	1	20	0.40	0.70	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
43	1	20	0.40	0.80	0	0	1900	0.06	0	0	1900	0.07	0	0	1900	0.00	0	0	1900	0.00
44	1	20	0.40	0.10	235	87	1022	0.06	4487	1499	1022	0.25	1604	1577	1022	0.00	1582	1568	1022	0.00
45	1	20	0.40	0.20	34	2	1854	0.07	8	0	1854	0.03	462	13	1854	0.01	116	12	1854	0.01
46	1	20	0.40	0.30	0	0	1900	0.03	0	0	1900	0.03	0	0	1900	0.00	0	0	1900	0.00
47	1	20	0.50	0.30	20	0	1883	0.03	4	0	1883	0.03	242	1	1883	0.01	41	1	1883	0.01
48	1	20	0.50	0.40	2	0	1900	0.03	1	0	1900	0.03	57	0	1900	0.00	7	0	1900	0.00

Table B.8 . Branch-and-bound performance results: random dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
147	1	40	0.60	0.50	140	0	3900	0.08	1	0	3900	0.12	903	0	3900	0.31	454	0	3900	0.25
148	1	40	0.60	0.60	160	0	3900	0.05	0	0	3900	0.06	422	0	3900	0.12	143	0	3900	0.09
149	1	40	0.60	0.70	0	0	3900	0.03	0	0	3900	0.03	288	0	3900	0.01	9	0	3900	0.01
150	1	40	0.60	0.80	0	0	3900	0.07	0	0	3900	0.08	0	0	3900	0.00	0	0	3900	0.00
151	1	40	0.70	0.70	214	0	3900	0.03	0	0	3900	0.04	468	0	3900	0.07	113	0	3900	0.06
152	1	40	0.70	0.80	0	0	3900	0.02	0	0	3900	0.02	0	0	3900	0.00	0	0	3900	0.00
153	1	40	0.70	0.10	217419011	108700747	-	1800.01	74502459	74052116	1838*	1800.06	1117588453	1072118168	1443*	1800	872784127	872440475	1443*	1800
154	1	40	0.70	0.20	6901145	2774987	-	1800	5184576	5059902	-	1800.01	26652669	0	-	1800	1623925	0	-	1800
155	1	40	0.70	0.30	115945	0	3900	55.09	22831	0	3900	60.48	24214044	0	-	1800	993139	0	3900	390.16
156	1	40	0.70	0.40	4820	0	3900	3.71	215	0	3900	2.17	219409	0	3900	28.61	18487	0	3900	7.66
157	1	40	0.70	0.50	169	0	3900	0.14	32	0	3900	0.19	26467	0	3900	1.63	2165	0	3900	1.05
158	1	40	0.70	0.60	222	0	3900	0.07	0	0	3900	0.07	506	0	3900	0.21	239	0	3900	0.16
159	1	40	0.80	0.60	231	0	3900	0.10	0	0	3900	0.11	586	0	3900	0.40	346	0	3900	0.27
160	1	40	0.80	0.70	307	0	3900	0.05	0	0	3900	0.09	584	0	3900	0.15	207	0	3900	0.13
161	1	40	0.80	0.80	0	0	3900	0.02	0	0	3900	0.02	175	0	3900	0.01	0	0	3900	0.01
162	1	40	0.80	0.10	120741130	55389281	1253	957.14	71650636	71304088	1639*	1800.06	1426598876	1420643204	1253*	1800	1144967486	1144961196	1253*	1800
163	1	40	0.80	0.20	5923167	2846090	-	1800	5546303	5476026	-	1800.01	26117653	0	-	1800	1652194	0	-	1800
164	1	40	0.80	0.30	620922	0	3900	321.47	118946	0	3900	294.24	15569784	0	-	1800	2374333	0	-	1800
165	1	40	0.80	0.40	10981	0	3900	10.13	1065	0	3900	4.56	2435034	0	3900	188.89	61972	0	3900	34.22
166	1	40	0.80	0.50	561	0	3900	0.67	109	0	3900	0.55	67458	0	3900	8.21	6419	0	3900	4.55
167	1	50	0.20	0.40	0	0	4900	0.10	0	0	4900	0.05	0	0	4900	0.00	0	0	4900	0.00
168	1	50	0.20	0.50	0	0	4900	0.10	0	0	4900	0.05	0	0	4900	0.00	0	0	4900	0.00
169	1	50	0.20	0.60	0	0	4900	0.05	0	0	4900	0.05	0	0	4900	0.00	0	0	4900	0.00
170	1	50	0.20	0.70	0	0	4900	0.05	0	0	4900	0.06	0	0	4900	0.00	0	0	4900	0.00
171	1	50	0.20	0.80	0	0	4900	0.05	0	0	4900	0.06	0	0	4900	0.00	0	0	4900	0.00
172	1	50	0.20	0.10	1773554	2	4841	16.64	8706076	0	4841	527.38	1177137958	0	-	1800	94390066	0	-	1800
173	1	50	0.20	0.20	10	0	4900	0.03	1	0	4900	0.04	1178	0	4900	0.12	92	0	4900	0.09
174	1	50	0.20	0.30	0	0	4900	0.03	0	0	4900	0.03	91	0	4900	0.00	0	0	4900	0.00
175	1	50	0.30	0.20	3219	0	4900	1.80	1681	0	4900	4.26	5863676	0	4900	508.99	42297	0	4900	25.31
176	1	50	0.30	0.30	85	0	4900	0.08	19	0	4900	0.13	12223	0	4900	1.13	529	0	4900	0.55
177	1	50	0.30	0.40	44	0	4900	0.04	0	0	4900	0.05	297	0	4900	0.06	61	0	4900	0.06
178	1	50	0.30	0.50	0	0	4900	0.03	0	0	4900	0.03	0	0	4900	0.00	0	0	4900	0.00
179	1	50	0.30	0.60	0	0	4900	0.06	0	0	4900	0.06	0	0	4900	0.00	0	0	4900	0.00
180	1	50	0.30	0.70	0	0	4900	0.04	0	0	4900	0.04	0	0	4900	0.00	0	0	4900	0.00
181	1	50	0.30	0.80	0	0	4900	0.04	0	0	4900	0.04	0	0	4900	0.00	0	0	4900	0.00
182	1	50	0.30	0.10	181452704	81522603	-	1800.01	76974858	74904648	-	1800	858686197	0	-	1800	77398980	0	-	1800
183	1	50	0.40	0.10	184498689	92115158	-	1800	62571612	62058584	-	1800	710470312	0	-	1800	77469780	0	-	1800
184	1	50	0.40	0.20	775053	0	4900	322.20	225805	0	4900	492.12	21018874	0	-	1800	1067955	0	-	1800
185	1	50	0.40	0.30	3945	0	4900	2.99	387	0	4900	2.09	1215974	0	4900	108.00	19419	0	4900	14.01
186	1	50	0.40	0.40	185	0	4900	0.16	30	0	4900	0.19	23452	0	4900	2.34	1054	0	4900	0.96
187	1	50	0.40	0.50	123	0	4900	0.06	0	0	4900	0.07	439	0	4900	0.19	149	0	4900	0.16
188	1	50	0.40	0.60	115	0	4900	0.03	0	0	4900	0.03	396	0	4900	0.06	40	0	4900	0.05
189	1	50	0.40	0.70	0	0	4900	0.04	0	0	4900	0.04	0	0	4900	0.00	0	0	4900	0.00
190	1	50	0.40	0.80	0	0	4900	0.04	0	0	4900	0.04	0	0	4900	0.00	0	0	4900	0.00
191	1	50	0.50	0.60	180	0	4900	0.06	0	0	4900	0.07	558	0	4900	0.21	163	0	4900	0.17
192	1	50	0.50	0.70	0	0	4900	0.02	0	0	4900	0.02	344	0	4900	0.01	5	0	4900	0.01
193	1	50	0.50	0.80	0	0	4900	0.04	0	0	4900	0.03	0	0	4900	0.00	0	0	4900	0.00
194	1	50	0.50	0.10	171332662	85654320	-	1800	55816400	55542747	-	1800.01	633252803	0	-	1800	92784958	0	-	1800
195	1	50	0.50	0.20	4027048	1185966	-	1800	2693828	2579130	-	1800.01	18970055	0	-	1800	875346	0	-	1800

Table B.8 . Branch-and-bound performance results: random dataset. (cont.)

Instances				BB-E			BB-S			Rec-D			BB-D			
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
196	1	50	0.50	0.30	42644	0	4900	31.65	6853	0	4900	29.77	12217386	0	1800	213.54
197	1	50	0.50	0.40	1908	0	4900	2.50	147	0	4900	1.20	84424	0	4900	6.70
198	1	50	0.50	0.50	219	0	4900	0.15	0	0	4900	0.14	1120	0	4900	0.46
199	1	50	0.60	0.40	8411	0	4900	11.67	340	0	4900	4.48	577184	0	4900	24.72
200	1	50	0.60	0.50	287	0	4900	0.45	61	0	4900	0.41	54779	0	4900	2.73
201	1	50	0.60	0.60	367	0	4900	0.12	0	0	4900	0.13	687	0	4900	0.37
202	1	50	0.60	0.70	272	0	4900	0.06	0	0	4900	0.06	661	0	4900	0.16
203	1	50	0.60	0.80	0	0	4900	0.04	0	0	4900	0.04	0	0	4900	0.01
204	1	50	0.60	0.10	150187815	75092282	-	1800	51451070	51262672	2880*	1800.03	534694974	135324739	2055*	1800
205	1	50	0.60	0.20	3989410	1873626	-	1800	3704588	3665253	-	1800.01	19308653	0	1800	1800
206	1	50	0.60	0.30	354900	0	4900	275.31	48174	0	4900	219.34	12976952	0	1800	1800
207	1	50	0.70	0.30	1960981	0	4900	1574.38	343369	0	4900	1481.63	11975012	0	1800	1800
208	1	50	0.70	0.40	17486	0	4900	26.80	2423	0	4900	15.81	8356378	0	4900	126.30
209	1	50	0.70	0.50	1301	0	4900	1.89	174	0	4900	1.49	135896	0	4900	10.33
210	1	50	0.70	0.60	264	0	4900	0.17	0	0	4900	0.23	810	0	4900	0.67
211	1	50	0.70	0.70	135	0	4900	0.08	0	0	4900	0.11	800	0	4900	0.32
212	1	50	0.70	0.80	0	0	4900	0.03	0	0	4900	0.03	305	0	4900	0.01
213	1	50	0.70	0.10	130333637	65166751	-	1800	47237830	47096390	-	1800.01	482758851	246523749	2115*	1800
214	1	50	0.70	0.20	3186361	1589211	-	1800	3575935	3555499	-	1800.04	17070231	0	1800	1800
215	1	50	0.80	0.30	2026939	374450	-	1800	1226418	1178823	-	1800.02	15246861	0	1800	1800
216	1	50	0.80	0.40	66383	0	4900	91.70	11253	0	4900	69.31	7071458	0	1800	548.49
217	1	50	0.80	0.50	5796	0	4900	10.97	308	0	4900	4.36	194236	0	4900	24.80
218	1	50	0.80	0.60	257	0	4900	0.27	5	0	4900	0.35	3325	0	4900	1.41
219	1	50	0.80	0.70	397	0	4900	0.12	0	0	4900	0.19	930	0	4900	0.48
220	1	50	0.80	0.80	137	0	4900	0.03	0	0	4900	0.07	794	0	4900	0.10
221	1	50	0.80	0.10	126437268	63228633	-	1800	42412855	42312758	2762*	1800.02	521357942	0	1800	1800
222	1	50	0.80	0.20	2560908	1280212	-	1800	2938820	2927155	-	1800.07	16715767	0	1800	1800
223	1	75	0.40	0.20	1571533	760493	-	1800.01	1582402	1574618	-	1800.05	7651677	0	1800	1800
224	1	75	0.40	0.40	8758	0	7400	25.04	321	0	7400	11.90	305465	0	7400	74.98
225	1	75	0.40	0.60	445	0	7400	0.25	0	0	7400	0.31	1032	0	7400	1.02
226	1	75	0.40	0.80	0	0	7400	0.12	0	0	7400	0.13	0	0	7400	0.01
227	1	75	0.50	0.20	1081560	540725	-	1800.01	1319176	1315886	-	1800.23	9432442	0	1800	1800
228	1	75	0.50	0.40	47656	0	7400	183.32	3628	0	7400	75.60	7590217	0	1800	768.37
229	1	75	0.50	0.60	649	0	7400	0.57	0	0	7400	0.66	1315	0	7400	2.34
230	1	75	0.50	0.80	0	0	7400	0.04	0	0	7400	0.03	447	0	7400	0.03
231	1	75	0.60	0.80	480	0	7400	0.09	0	0	7400	0.13	1450	0	7400	0.57
232	1	75	0.60	0.10	91600630	45800317	-	1800	21529964	21504339	-	1800.03	430611839	0	1800	1800
233	1	75	0.60	0.20	754248	377126	-	1800	1065797	1064079	-	1800.27	7459503	0	1800	1800
234	1	75	0.60	0.30	689074	258147	-	1800	793433	789793	-	1800.06	8304828	0	1800	1800
235	1	75	0.60	0.40	324206	0	7400	1089.95	39739	0	7400	799.20	2744650	0	1800	1800
236	1	75	0.60	0.50	23452	0	7400	116.51	538	0	7400	33.00	574372	0	7400	183.75
237	1	75	0.60	0.60	775	0	7400	1.25	7	0	7400	1.42	18228	0	7400	8.44
238	1	75	0.60	0.70	435	0	7400	0.46	0	0	7400	0.77	1591	0	7400	2.20
239	1	75	0.70	0.60	615	0	7400	2.42	220	0	7400	5.39	430992	0	7400	58.13
240	1	75	0.70	0.80	632	0	7400	0.31	0	0	7400	0.34	1835	0	7400	1.54
241	1	75	0.70	0.20	553430	276717	-	1800.01	775204	774276	-	1800.42	7202226	0	1800	1800
242	1	75	0.70	0.40	419733	32017	-	1800.01	124440	108760	7400*	1800.15	2682838	0	1800	1800
243	1	75	0.80	0.50	115073	0	7400	983.87	2965	0	7400	247.92	9123601	0	1800	1466.16
244	1	75	0.80	0.60	4795	0	7400	44.63	481	0	7400	22.77	788670	0	7400	182.41

Table B.8 . Branch-and-bound performance results: random dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D				
ID	Rep	$ V(G) $	T	Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
245	1	75	0.80	0.70	713	0	7400	1.91	0	0	7400	1.90	2147	0	7400	8.89	1356	0	7400	6.22
246	1	75	0.80	0.80	898	0	7400	0.54	0	0	7400	0.98	2135	0	7400	3.37	899	0	7400	2.72
247	1	75	0.80	0.10	80103136	40051570	-	1800	17580927	17567124	-	1800.02	398258427	0	-	1800.05	17692682	0	-	1800
248	1	75	0.80	0.20	397956	198980	-	1800.01	621431	620845	-	1800.72	6606948	0	-	1800.05	219731	0	-	1800
249	1	75	0.80	0.30	429734	214754	-	1800	590681	589718	-	1801.13	5500128	0	-	1800.04	92863	0	-	1800
250	1	75	0.80	0.40	328717	63975	-	1800.01	293403	289100	-	1800.02	8182656	0	-	1800.04	66080	0	-	1800
251	1	75	0.20	0.70	0	0	7400	0.12	0	0	7400	0.13	0	0	7400	0.00	0	0	7400	0.00
252	1	75	0.20	0.80	0	0	7400	0.12	0	0	7400	0.13	0	0	7400	0.00	0	0	7400	0.00
253	1	75	0.20	0.10	108059910	52286310	-	1800.01	39896857	39414202	-	1800	776715619	0	-	1800	41225387	0	-	1800
254	1	75	0.20	0.20	11397	0	7400	9.18	1890	0	7400	11.12	11631620	0	-	1800	82651	0	7400	115.88
255	1	75	0.20	0.30	131	0	7400	0.17	19	0	7400	0.27	28708	0	7400	3.81	912	0	7400	1.86
256	1	75	0.20	0.40	154	0	7400	0.07	0	0	7400	0.08	471	0	7400	0.32	90	0	7400	0.23
257	1	75	0.20	0.50	0	0	7400	0.03	0	0	7400	0.03	325	0	7400	0.02	1	0	7400	0.02
258	1	75	0.20	0.60	0	0	7400	0.12	0	0	7400	0.13	0	0	7400	0.00	0	0	7400	0.00
259	1	75	0.30	0.30	9885	0	7400	16.97	1105	0	7400	12.97	6886837	0	7400	1220.63	68845	0	7400	121.20
260	1	75	0.30	0.40	222	0	7400	0.40	49	0	7400	0.64	81471	0	7400	12.82	3330	0	7400	5.87
261	1	75	0.30	0.50	187	0	7400	0.15	0	0	7400	0.16	756	0	7400	0.82	248	0	7400	0.65
262	1	75	0.30	0.60	135	0	7400	0.05	0	0	7400	0.06	684	0	7400	0.26	98	0	7400	0.22
263	1	75	0.30	0.70	0	0	7400	0.12	0	0	7400	0.12	0	0	7400	0.01	0	0	7400	0.01
264	1	75	0.30	0.80	0	0	7400	0.12	0	0	7400	0.12	0	0	7400	0.01	0	0	7400	0.01
265	1	75	0.30	0.10	106528267	53244032	-	1800	32784303	32658572	-	1800.02	665195964	0	-	1800.01	30712854	0	-	1800
266	1	75	0.30	0.20	1709462	261401	-	1800	684730	625764	-	1800.02	10115173	0	-	1800	271541	0	-	1800
267	1	75	0.40	0.30	321151	0	7400	635.32	33320	0	7400	567.00	8755469	0	-	1800.01	110053	0	-	1800
268	1	75	0.40	0.50	424	0	7400	0.48	8	0	7400	0.58	22991	0	7400	3.87	3364	0	7400	4.61
269	1	75	0.40	0.70	340	0	7400	0.04	0	0	7400	0.07	920	0	7400	0.32	145	0	7400	0.28
270	1	75	0.40	0.10	103688100	51842584	-	1800	26214387	26158486	-	1800.01	536369541	0	-	1800.01	28311263	0	-	1800
271	1	75	0.50	0.30	818715	240120	-	1800.01	530768	520049	-	1800.16	3372954	0	-	1800.02	136626	0	-	1800
272	1	75	0.50	0.50	1600	0	7400	5.13	251	0	7400	5.35	353265	0	7400	180.49	32756	0	7400	53.67
273	1	75	0.50	0.70	271	0	7400	0.19	0	0	7400	0.26	1302	0	7400	1.56	471	0	7400	1.08
274	1	75	0.50	0.10	97820641	48910316	-	1800	23351590	23314527	-	1800.03	485111115	0	-	1800.02	26708694	0	-	1800
275	1	75	0.70	0.70	620	0	7400	0.89	0	0	7400	1.16	1870	0	7400	5.37	1028	0	7400	3.85
276	1	75	0.70	0.10	88712630	44356317	-	1800	19387381	19369029	-	1800.06	421948848	0	-	1800.05	20955002	0	-	1800
277	1	75	0.70	0.30	576139	281977	-	1800.01	724604	722830	-	1800.40	3419463	0	-	1800.03	129587	0	-	1800
278	1	75	0.70	0.50	72086	0	7400	466.87	834	0	7400	94.23	931538	0	-	1800	212550	0	7400	448.34
279	1	100	0.30	0.10	69460104	34729973	-	1800	15842840	15818671	-	1800.02	449053286	0	-	1800.02	14625995	0	-	1800
280	1	100	0.30	0.20	806733	400671	-	1800	852031	849288	-	1800.04	5811358	0	-	1800.02	112691	0	-	1800
281	1	100	0.30	0.30	412343	0	9900	1337.67	40384	0	9900	1321.85	3308000	0	-	1800.01	34939	0	-	1800
282	1	100	0.30	0.40	8826	0	9900	41.44	336	0	9900	20.15	516437	0	9900	715.90	57354	0	9900	190.85
283	1	100	0.30	0.50	500	0	9900	0.93	1	0	9900	1.11	5142	0	9900	6.58	2265	0	9900	6.61
284	1	100	0.30	0.60	262	0	9900	0.30	0	0	9900	0.46	1381	0	9900	2.43	512	0	9900	2.03
285	1	100	0.30	0.70	1	0	9900	0.06	0	0	9900	0.08	1154	0	9900	0.42	108	0	9900	0.38
286	1	100	0.30	0.80	0	0	9900	0.12	0	0	9900	0.16	0	0	9900	0.02	0	0	9900	0.02
287	1	100	0.40	0.60	622	0	9900	1.35	0	0	9900	1.71	1882	0	9900	8.97	1129	0	9900	6.78
288	1	100	0.40	0.80	0	0	9900	0.08	0	0	9900	0.06	642	0	9900	0.06	0	0	9900	0.06
289	1	100	0.40	0.20	499921	249962	-	1800	650137	649182	-	1800.02	5373285	0	-	1800.04	107740	0	-	1800
290	1	100	0.40	0.40	80534	0	9900	653.95	8990	0	9900	339.84	4012186	0	-	1800.03	89108	0	-	1800
291	1	100	0.50	0.40	256719	29630	-	1800.01	114030	107731	-	1800.20	1190566	0	-	1800.05	18595	0	-	1800
292	1	100	0.50	0.60	779	0	9900	4.07	66	0	9900	5.15	177254	0	9900	39.72	19316	0	9900	52.05
293	1	100	0.50	0.80	884	0	9900	0.39	0	0	9900	0.46	2243	0	9900	2.70	531	0	9900	2.37

Table B.8 . Branch-and-bound performance results: random dataset. (cont.)

Instances				BB-E				BB-S				Rec-D				BB-D			
ID	Rep	$ V(G) $	T Ω	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)	Explored	Active	Best UB	CPU(s)
294	1	100	0.50 0.20	315624	157814	-	1800.01	461059	460649	-	1800.37	4072097	0	-	1800.06	98157	0	-	1800
295	1	100	0.60 0.30	203146	101572	-	1800.02	284845	284600	-	1803.46	3672897	0	-	1800.08	30526	0	-	1800
296	1	100	0.60 0.40	159845	27025	-	1800	237245	236774	9900*	1800.36	6185925	0	-	1800.08	14036	0	-	1800
297	1	100	0.60 0.50	112893	460	-	1800.01	1490	0	9900	551.73	2198111	0	-	1800.06	76925	0	-	1800
298	1	100	0.60 0.60	2441	0	9900	22.30	486	0	9900	31.12	1342255	0	9900	1333.14	130172	0	9900	446.49
299	1	100	0.60 0.70	966	0	9900	3.30	0	0	9900	4.05	2872	0	9900	18.97	1737	0	9900	14.46
300	1	100	0.60 0.80	987	0	9900	0.90	0	0	9900	1.29	2855	0	9900	8.15	1117	0	9900	6.22
301	1	100	0.60 0.10	58901738	29450871	-	1800	10359811	10354044	-	1800.24	343964195	0	-	1800.10	10153689	0	-	1800
302	1	100	0.60 0.20	203026	101515	-	1800.01	328334	328131	-	1800.80	5092208	0	-	1800.09	98101	0	-	1800
303	1	100	0.70 0.20	142918	71461	-	1800.02	255294	255172	-	1800.51	4341955	0	-	1800.12	88961	0	-	1800
304	1	100	0.70 0.40	149709	66705	-	1800	213012	212772	-	1800.15	2070664	0	-	1800.23	41920	0	-	1800
305	1	100	0.70 0.60	41757	0	9900	790.75	1007	0	9900	175.87	894968	0	-	1800.10	349390	0	9900	1354.08
306	1	100	0.70 0.80	1302	0	9900	2.39	0	0	9900	2.73	3362	0	9900	14.45	1582	0	9900	11.21
307	1	100	0.80 0.20	110928	55466	-	1800.01	196499	196422	-	1800.08	3390789	0	-	1800.16	107150	0	-	1800
308	1	100	0.80 0.30	108194	54099	-	1800.02	179994	179901	-	1800.71	1630737	0	-	1800.17	50892	0	-	1800
309	1	100	0.80 0.40	129332	64448	-	1800.02	181861	181710	-	1800.56	1153542	0	-	1800.16	25698	0	-	1800
310	1	100	0.80 0.50	106344	34053	-	1800.01	109091	108681	9900*	1800.70	1286929	0	-	1800.15	18753	0	-	1800
311	1	100	0.80 0.60	75050	911	-	1800.01	1492	0	9900	475.85	574227	0	-	1800.16	31019	0	-	1800
312	1	100	0.80 0.70	1111	0	9900	9.54	12	0	9900	13.82	36064	0	9900	72.81	18149	0	9900	67.55
313	1	100	0.80 0.80	1088	0	9900	3.71	0	0	9900	4.89	3860	0	9900	23.97	2082	0	9900	18.44
314	1	100	0.80 0.10	53249024	26624514	-	1800	9218076	9214557	-	1800.21	272870798	0	-	1800.18	9973786	0	-	1800
315	1	100	0.20 0.30	1937	0	9900	4.75	167	0	9900	5.27	250464	0	9900	214.29	21607	0	9900	57.93
316	1	100	0.20 0.40	235	0	9900	0.30	0	0	9900	0.37	914	0	9900	1.94	570	0	9900	1.73
317	1	100	0.20 0.50	163	0	9900	0.12	0	0	9900	0.14	847	0	9900	0.82	166	0	9900	0.65
318	1	100	0.20 0.60	0	0	9900	0.03	0	0	9900	0.03	292	0	9900	0.02	0	0	9900	0.02
319	1	100	0.20 0.70	0	0	9900	0.15	0	0	9900	0.16	0	0	9900	0.01	0	0	9900	0.01
320	1	100	0.20 0.80	0	0	9900	0.14	0	0	9900	0.17	0	0	9900	0.01	0	0	9900	0.01
321	1	100	0.20 0.10	76691670	38289606	-	1800	20524411	20457042	-	1800.01	512256457	0	-	1800.01	14878061	0	-	1800
322	1	100	0.20 0.20	312043	0	9900	549.39	70942	0	9900	983.02	5254921	0	-	1800	142921	0	-	1800
323	1	100	0.40 0.70	627	0	9900	0.64	0	0	9900	0.74	1865	0	9900	5.86	717	0	9900	3.88
324	1	100	0.40 0.10	68603630	34301817	-	1800	12425663	12413954	-	1800.06	358196988	0	-	1800.04	11327686	0	-	1800
325	1	100	0.40 0.30	435434	201145	-	1800.01	361868	359949	-	1800.47	1840661	0	-	1800.04	44862	0	-	1800
326	1	100	0.40 0.50	3975	0	9900	27.97	403	0	9900	23.42	751762	0	9900	829.54	76042	0	9900	245.60
327	1	100	0.50 0.50	60155	0	9900	668.07	889	0	9900	162.90	660870	0	-	1800.04	252238	0	9900	1042.41
328	1	100	0.50 0.70	843	0	9900	1.64	0	0	9900	2.01	2377	0	9900	9.53	1213	0	9900	8.14
329	1	100	0.50 0.10	64810784	32405394	-	1800	10489317	10482209	-	1800.03	310663491	0	-	1800.06	10068397	0	-	1800
330	1	100	0.50 0.30	288559	143079	-	1800.02	346404	345910	-	1800.67	3703791	0	-	1800.05	64709	0	-	1800
331	1	100	0.70 0.30	146928	73466	-	1800.01	209785	209650	-	1800.85	3115333	0	-	1800.12	46684	0	-	1800
332	1	100	0.70 0.50	83615	18272	-	1800.01	36125	33302	9900*	1800.15	2907165	0	-	1800.13	34512	0	-	1800
333	1	100	0.70 0.70	1258	0	9900	6.18	0	0	9900	7.56	3367	0	9900	39.69	2489	0	9900	28.40
334	1	100	0.70 0.10	53696664	26848334	-	1800	10356510	10351717	-	1800.20	264992064	0	-	1800.14	8260306	0	-	1800
Average				9840897	4685119	4511	383.03	5206718	4960160	4599	400.03	56701221	16343373	4294	510.95	18456655	15640383	4360	447.19