ABSTRACT

SAFFARI, SAEED. Set Covering Problem with Conflict Constraints. (Under the direction of Dr. Yahya Fathi.)

In this dissertation we consider the set covering problem with conflict constraints, i.e., from each conflicting pair of columns (subsets) at most one column can appear in any feasible solution. After discussing various settings and potential applications in which this problem may arise, we formulate the problem as an Integer Programming (IP) model. Subsequently we develop various strategies to solve this IP model effectively. We introduce several families of Valid Inequalities (VIs) and pre-processing procedures in this context. Some of these valid inequalities and pre-processing procedures are adaptations of those previously employed in the context of the set covering problem or the node packing problem (i.e., the stable set problem). But others are new and a direct result of the presence of both groups of constraints simultaneously.

In particular, we discuss two families of VIs in the latter group, namely the Covering-Packing Valid Inequalities (CP-VIs) and the Covering-Cliques Valid Inequalities (CC-VIs). We analyze the facetial properties of these VIs and show that, although these VIs are not facetial for all instances of the problem, they can be facetial if the problem has a special structure. We also show that in practice these VIs can be quite effective in enhancing the efficiency of the branch-and-bound algorithm for some large instances of the problem.

We then discuss an appropriate strategy for including all the said VIs and pre-processing procedures in the context of a branch-and-cut algorithm for solving the IP model, and carry out a comprehensive computational experiment to demonstrate their effectiveness. It turns out that our proposed method results in significantly enhancing the effectiveness of the branch-and-bound algorithm, and hence allowing us to solve larger instances of IP model within practical time limits.

In the second part of this dissertation we introduce a bi-criteria set covering problem with conflict constraints in which we are interested in simultaneously maximizing the number of covered rows (elements) and minimizing the number of selected columns (subsets). In this context we introduce and define the problem, and propose two algorithms for obtaining the non-dominated frontier for this bi-criteria optimization problem. Subsequently we propose and discuss several classes of valid inequalities for the IP model of this problem and, through a limited computational experiment, we demonstrate their effectiveness in the context of a branch-and-cut algorithm. We also develop a local search heuristic algorithm for solving larger instances of this problem, and demonstrate its effectiveness through a numeric experiment.

© Copyright 2019 by Saeed Saffari

All Rights Reserved

Set Covering Problem with Conflict Constraints

by
Saeed Saffari

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

Industrial Engineering

Raleigh, North Carolina

2019

APPROVED BY:

Dr. Michael G. Kay

Dr. Osman Ozaltin

Dr. Golbarg Kazemi Tutunchi

Dr. Yahya Fathi
Chair of Advisory Committee

DEDICATION

To my beloved parents and my lovely wife who made this possible.

BIOGRAPHY

Saeed Saffari is a Ph.D. student in the Industrial Engineering Graduate Program at North Carolina State University. He was born in Sabzevar, Iran. He has received his Bachelor of Science degree in Industrial Engineering from Sharif University of Technology in May 2013. He then went to Turkey in August 2013, and he received his Master of Science degree in Industrial Engineering from Koç University in May 2015.

To further his education, he came to the United States in August 2015 and joined the Industrial Engineering program at North Carolina State University. Currently, Saeed is working as a decision scientist at Grubhub Holding Inc. in New York City, NY.

ACKNOWLEDGEMENTS

This dissertation would have not been completed without the guidance, help, and support I received from my mentors, friends, and my family.

I was very fortunate to have Dr. Yahya Fathi as my adviser for this research. Dr. Fathi offered me his guidance and time throughout my research, keeping me positive through tough times. This dissertation would not have been complete without his patience and encouragement.

I would like to thank my committee members Dr. Michael G. Kay, Dr. Osman Ozaltin, and Dr. Golbarg Kazemi Tutunchi for their constant guidance throughout my research, which pushed me to set higher standards for my work.

I would like to extend my thanks to all the faculties and staff of the ISE department at North Carolina State University who contributed to my education. Also a big thanks to all my friends for making the Ph.D. life more fun and interesting.

My special thanks go to my beloved wife, Samira, without her support and help I could not have made it through the hard moments of my Ph.D. studies. I am sincerely grateful to my parents, my sister, and my in-laws for their love and warm support.

TABLE OF CONTENTS

List of Tables	viii
List of Figures	x
Chapter 1 Introduction	1
1.1 Statement of the Problem	1
1.2 Applications	3
1.3 Contributions	4
Chapter 2 Background	5
2.1 Set Covering Problem	5
2.1.1 Integer Programming Formulation	5
2.1.2 Solution Methodologies for the Set Covering Problem	6
2.2 Maximum Stable Set Problem	8
2.2.1 Integer Programming Formulation	8
2.2.2 Brief Overview of Solution Methodologies for the Maximum Stable Set Problem	9
2.3 Other Problems with Conflict Constraints	9
Chapter 3 Pre-processing Procedures	11
3.1 Row Reduction	12
3.2 Column Reduction	12
3.3 A Computational Experiment	13
3.3.1 Data sets	13
3.3.2 A Comparative Study	17
Chapter 4 Valid Inequalities	20
4.1 Maximal Clique Valid Inequalities	21
4.2 Odd-Cycle Valid Inequalities	21
4.3 Covering-Packing Valid Inequalities	21
4.3.1 Definitions and Properties	22
4.3.2 Facet-defining Properties of CP-VIs	22
4.3.3 Generalized Covering-Packing Valid Inequalities	27
4.4 Covering-Clique Valid Inequalities	28
4.4.1 Facet-defining Properties of CC-VIs	29
4.5 Conflicting-Set Valid Inequalities	33
4.6 Column-Domination Valid Inequalities	35
Chapter 5 Computational Effectiveness of Valid Inequalities	36
5.1 Covering-Packing Valid Inequalities	36
5.1.1 Computational Experiments	39
5.1.2 Generalized Covering-Packing Valid Inequalities	47
5.2 Covering-Clique Valid Inequalities	47

5.2.1	Computational Experiments	47
5.3	Maximal Clique Valid Inequalities	55
5.3.1	Maximal Clique Finder Algorithm	56
5.3.2	Adding Maximal Clique Valid Inequalities to the Model	57
5.3.3	A Computational Experiment	58
5.4	Column-Domination Valid Inequalities and Conflicting-Set Valid Inequalities . .	62
5.4.1	Impact of Adding Column-Domination Valid Inequalities to the Model . .	62
5.4.2	Impact of Adding Conflicting-Set Valid Inequalities to the Model	62
5.4.3	Adding CD-VIs and CS-VIs before MC-VIs	63
5.5	Odd-Cycle Valid Inequalities	65
5.6	Including All Valid Inequalities Within a B&B Framework	66
5.6.1	A Computational Experiment	67
Chapter 6	Bi-criteria Set Covering Problem With Conflict Constraints	70
6.1	Statement of the Problem	70
6.1.1	Terminology	71
6.1.2	Efficient Frontier: An Illustrative Example	72
6.2	Strategies for Obtaining the Non-dominated Points	73
6.2.1	The ϵ -constraint Approach	74
6.3	A Computational Experiment	76
Chapter 7	Exact Methods for Solving Models $IP-\epsilon_1$ and $IP-\epsilon_2$	80
7.1	Bicriteria-Covering-Packing Valid Inequalities	80
7.2	Bicriteria-Covering-Clique Valid Inequalities	82
7.3	Bicriteria-Conflicting-Set Valid Inequalities	85
7.4	MC-VIs and CD-VIs	86
7.5	Other Valid Inequalities	86
7.6	Computational Effectiveness of Valid Inequalities	87
7.7	A Strategy for Adding the Valid Inequalities	88
7.7.1	First Experiment	90
7.7.2	Second Experiment	93
7.8	Concluding Remarks	96
Chapter 8	Inexact Methods for Solving the Model $IP-\epsilon_1$	97
8.1	A Local Search Heuristic	97
8.1.1	Computational Experiments	98
8.2	A Lagrangian Heuristic	100
Chapter 9	Conclusions and Future Research	103
9.1	Future research	104
References		105
APPENDICES		111
Appendix A	Clarifying The Maximal Clique Finder Algorithm	112
Appendix B	Impact of CP-VIs in Presence of CPLEX Cuts	115

Appendix C	Impact of CC-VIs in Presence of CPLEX Cuts	118
Appendix D	Data for Instance Explained in Section 6.1.2	121
Appendix E	A Lagrangian Heuristic Procedure for Solving the Model $IP-\epsilon_1$. . .	125

LIST OF TABLES

Table 3.1	The information of demand-service data sets I and II	14
Table 3.2	The parameters of instances in GEO-10D data set	15
Table 3.3	The parameters of instances in RAN data set	16
Table 3.4	The parameters of instances in OR-L data set	16
Table 3.5	Data sets instances reduction using the pre-processing procedures	18
Table 5.1	Results of adding CP-VIs to the model <i>SCC-IP</i> for instances in data set A with $m = 400$ and $n = 1300$	41
Table 5.2	Results of adding CP-VIs to the model <i>SCC-IP</i> for RAN, GEO-10D, and OR-L instances	42
Table 5.3	Results of adding CC-VIs to the model <i>SCC-IP</i> for instances in data set B with $m = 400$ and $n = 1000$	50
Table 5.4	Results of adding CC-VIs to the model <i>SCC-IP</i> for RAN, GE-10D, and OR-L instances	51
Table 5.5	Results of the MCF algorithm on GEO-10D, RAN, and OR-L instances	59
Table 5.6	Results of adding MC-VIs to the model <i>SCC-IP</i>	60
Table 5.7	Results of adding CD-VIs and CS-VIs to the model <i>SCC-IP</i>	64
Table 5.8	Results of adding all VIs in the context of six-phase method to the model <i>SCC-IP</i>	68
Table 6.1	Values of the evaluation criteria corresponding to the efficient points of the illustrative example	72
Table 6.2	Execution time (seconds) of the two algorithms	77
Table 6.3	Execution times of the model $IP-\epsilon_1$ for instance G10.17	78
Table 6.4	Execution times of the model $IP-\epsilon_2$ for instance G10.17	79
Table 7.1	Selected instances from data sets GEO-10D and RAN	89
Table 7.2	Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_1$ for GEO-10D instances	91
Table 7.3	Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_1$ for RAN instances	92
Table 7.4	Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_2$ for GEO-10D instances	94
Table 7.5	Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_2$ for RAN instances	95
Table 8.1	The computational results of solving the $IP-\epsilon_1$ via the exact and inexact methods for GEO-10D instances	101
Table 8.2	The computational results of solving the $IP-\epsilon_1$ via the exact and inexact methods for RAN instances	102
Table B.1	Results of adding CP-VIs to the model <i>SCC-IP</i> for GEO-10D, RAN, and OR-L instances in the presence of CPLEX cuts	116

Table B.2	Results of adding CP-VIs to the model <i>SCC-IP</i> for instances in data set A in the presence of CPLEX cuts	117
Table C.1	Results of adding CC-VIs to the model <i>SCC-IP</i> for GEO-10D, RAN, and OR-L instances in the presence of CPLEX cuts	119
Table C.2	Results of adding CC-VIs to the model <i>SCC-IP</i> for instances in data set B in the presence of CPLEX cuts	120
Table D.1	Set covering matrix data for instance explained in Section 6.1.2	122
Table D.2	Conflict relations data for instance explained in Section 6.1.2	123

LIST OF FIGURES

Figure 5.1	Results of adding CP-VIs to the model $SCC-IP$ for the instances in data sets GEO-10D, RAN, and OR-L	45
Figure 5.2	Results of adding CP-VIs to the model $SCC-IP$ for the instances in data set A	45
Figure 5.3	Results of adding CC-VIs to the model $SCC-IP$ for the instances in data sets GEO-10D, RAN, and OR-L	53
Figure 5.4	Results of adding CC-VIs to the model $SCC-IP$ for the instances in data set B	54
Figure 6.1	Efficient frontier corresponding to the illustrative example	73
Figure 8.1	The pseudocode for local search heuristic procedure for solving the model $IP-\epsilon_1$	99
Figure A.1	The conflict graph G of Example 12	112

Chapter 1

Introduction

In this research we consider the set covering problem with conflict constraints, i.e., from each conflicting pair of columns (subsets) at most one column can appear in any feasible solution. We formulate the problem as an Integer Programming (IP) model, and develop various strategies to solve this IP model effectively. We introduce several families of Valid Inequalities (VIs) and pre-processing procedures in this context. We then discuss an appropriate strategy for including all the VIs and pre-processing procedures in the context of a branch-and-cut algorithm for solving the IP model, and carry out a comprehensive computational experiment to demonstrate their effectiveness.

Later in this research we introduce a bi-criteria set covering problem with conflict constraints in which we are interested in simultaneously maximizing the number of covered rows (elements) and minimizing the number of selected columns (subsets). In this context we introduce and define the problem, and propose two algorithms for obtaining the non-dominated frontier for this bi-criteria optimization problem. Subsequently we propose and discuss several classes of valid inequalities for the IP model of this problem and, through a limited computational experiment, we demonstrate their effectiveness in the context of a branch-and-cut algorithm. We also develop a local search heuristic algorithm for solving larger instances of this problem, and demonstrate its effectiveness through a numeric experiment.

1.1 Statement of the Problem

In this research, we consider the Set Covering problem with Conflict constraints (SCC). In this context, we have an m by n matrix $A = (a_{ij})$, where each entry a_{ij} is either 0 or 1, and the i -th row of this matrix (a^i) is said to be covered by column j (a_j) if $a_{ij} = 1$. We also have a collection F of pairs of columns (j, k) , and we say that columns j and k represent a conflict if $(j, k) \in F$. There is a cost c_j ($c_j \geq 0$) associated with each column a_j , and we seek to identify

conflict-free collection of columns of A to cover all rows of A at minimum cost. Following is an integer programming (IP) model for the problem SCC.

$$\text{minimize } \sum_{j=1}^n c_j x_j \quad (SCC-IP)$$

$$\text{subject to } \sum_{j=1}^n a_{ij} x_j \geq 1 \quad \forall i = 1, \dots, m \quad (1.1a)$$

$$x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F \quad (1.1b)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (1.1c)$$

Constraints in group 1.1a ensure that each row of A must be covered by at least one selected column of A . These constraints are same as those used in the well-known set covering problem. Constraints in group 1.1b represent a collection of conflicts between various pairs of columns in A . These constraints are similar to the constraints of the well-known maximum stable set (node packing) problem. We can represent these symmetric conflict relations by means of an undirected graph $G := (V, E)$, where the set of vertices is $V = \{v_j : j = 1, \dots, n\}$, i.e., each vertex v_j corresponds uniquely to a column j of matrix A (for all $j = 1, \dots, n$), and E corresponds to the set of edges of G so that for each two columns j and k ($j, k = 1, \dots, n$), $(v_j, v_k) \in E$ when there is a conflict between columns j and k , i.e., $(j, k) \in F$. We refer to this graph as the *conflict graph* for the problem SCC.

This IP model with either group of constraints alone (i.e., either 1.1a or 1.1b) has been previously studied in the open literature. But, to the best of our knowledge, the problem with simultaneous presence of both groups of constraints is not yet addressed.

Our first goal in this research is to study the structural properties of the polyhedra associated with the set covering problem, the stable set problem, and the intersection of the two as we have in this problem, and introduce appropriate valid inequalities and pre-processing techniques that can be effective for solving the problem SCC. Our second goal in this research is to introduce this problem in a bi-criteria context in which we are interested in simultaneously maximizing the number of covered rows of A and also minimizing the number of selected columns of A , and introduce appropriate exact and inexact methodologies that can be effective for solving the bi-criteria SCC problem. We carry out several computational experiments to assess the effectiveness of the proposed methods on an empirical basis.

In the remainder of this chapter we discuss various applications and settings where considering the problem with both groups of constraints (i.e., set covering constraints and stable set constraints) would be appropriate. We also give a summary of our contributions in this context.

In Chapter 2 we briefly introduce the set covering problem and the maximum stable set

problem, and give a brief review of the relevant literature on each problem. In Chapter 3 we discuss several techniques that can be employed to remove redundancies and tighten up the mathematical model. We refer to these techniques as the *pre-processing procedures*. In Chapter 4 we discuss six families of valid inequalities for the problem SCC, and in Chapter 5 we evaluate the effectiveness of these valid inequalities for solving the model *SCC-IP* through several computational experiments, and propose a strategy for adding these VIs to the model *SCC-IP*.

In Chapter 6 we introduce the bi-criteria set covering problem with conflict constraints, present a bi-criteria mathematical model for this problem, and discuss two strategies for solving this problem. Later, in Chapter 7 we propose appropriate valid inequalities for the models discussed in Chapter 6, and employ these valid inequalities to devise a branch-and-cut algorithm for solving the corresponding models. Subsequently, in Chapter 8 we introduce a local search heuristic procedure to solve larger instances of the bi-criteria problem. Finally, in Chapter 9 we provide a brief summary and several concluding remarks, and point out a few avenues for future research.

1.2 Applications

Many of the known applications of the set covering problem can potentially include conflict constraints. We give two examples here.

A practical application of SCC could be in the wireless communication services. In this application, a number of signal towers need be constructed to send signals to customer regions. Each signal tower can send signals to all customer regions within a certain distance, and each customer region must receive signals from (be covered by) at least one signal tower. Furthermore, in order to avoid electromagnetic interference, the distance between each pair of signal towers must be no less than a given distance. Such a situation can be represented by means of a set covering problem with conflict constraints, considering each customer region as a row and each signal tower location as a column of the zero-one matrix. The corresponding model, i.e., set covering problem with conflict constraints, seeks to identify conflict-free collection of signal tower locations to cover all customer regions at minimum cost.

Another practical application could be in multi-period staff-scheduling problem in which the set covering constraints are used to ensure that each task is assigned to at least one worker at a time period, and the conflict constraints represent prohibited co-assignments due to legal and/or technical considerations (e.g., same worker assigned to consecutive night shifts). In this example, the model *SCC-IP* seeks the least expensive way to assign all tasks to workers at different time periods with no conflicts.

1.3 Contributions

As mentioned earlier, in this dissertation, we study the set covering problem with conflict constraints in a single-objective context as well as in a bi-criteria context, and propose several pre-processing procedures and several families of valid inequalities for the corresponding IP models. To the best of our knowledge, the set covering problem with conflict constraints has not been previously studied in the open literature. Our specific contributions are as follows:

1. We introduce appropriate pre-processing techniques to remove redundancies, remove unnecessary variables, and tighten up the set of constraints.
2. We investigate and extend a family of valid inequalities that has been previously proposed for this model (i.e., covering-packing valid inequalities) and demonstrate their effectiveness. We show that when the constraint set of the model *SCC-IP* has a special structure, these VIs can be facet-defining for the convex hull of the feasible solutions for the IP model. We also demonstrate their effectiveness through a computational experiment.
3. We propose a new family of valid inequalities for the model *SCC-IP* that we refer to as the covering-clique valid inequalities (CC-VIs). We show that for *SCC-IP* models with a special structure, these VIs are also facet-defining, and we demonstrate their effectiveness through a computational experiment.
4. We propose an algorithm to find a subset of the maximal cliques of the conflict graph G so as to include every conflicting pair, and we present a method to add these maximal clique constraints to the model *SCC-IP*. We also propose column-domination valid inequalities and conflicting-set valid inequalities, and we show that by adding these VIs to the model *SCC-IP*, we can enhance the effectiveness of the maximal clique constraints.
5. We introduce the bi-criteria set-covering problem with conflict constraints, and propose an IP model to formulate this problem. We then introduce two strategies for solving this problem, and present the effectiveness of these strategies through a numerical experiment.
6. We propose appropriate valid inequalities for the IP models discussed for the bi-criteria problem, and using these valid inequalities we propose a branch-and-cut algorithm for solving this problem. We demonstrate the effectiveness of the proposed branch-and-cut algorithm through a computational experiment.
7. We develop a local search heuristic approach to solve relatively large instances of the bi-criteria set-covering problem with conflict constraints, and demonstrate the efficiency of this algorithm through a numerical experiment.

Chapter 2

Background

In this chapter, we discuss the set covering problem and the stable set problem separately, and give a brief review of the corresponding literature.

2.1 Set Covering Problem

The set covering problem (SCP) is a classical combinatorial optimization problem. Its goal is to find a subset of columns that cover all rows of a zero-one matrix at minimal cost. The SCP has numerous real-life applications including those involving crew scheduling [21, 4, 46, 39], facility location [9, 1, 61, 74, 73], vehicle routing [64, 19], selection of portfolio [54], and conservation biology [49]. Various algorithms are proposed for solving this problem. In the next subsections, we give the mathematical model of SCP and a brief overview of the solution methods.

2.1.1 Integer Programming Formulation

In this subsection, we present the integer programming model for the SCP in the literature. We have an m by n matrix $A = (a_{ij})$, where each entry a_{ij} is either 0 or 1, and the i -th row of this matrix (a^i) is said to be covered by column j (a_j) if $a_{ij} = 1$. Now, for each column a_j , we define the decision variable $x_j = 1$ if column a_j is selected to be in the optimal cover, and $x_j = 0$ otherwise. Following is an IP model for the problem SCP.

$$\text{minimize} \quad \sum_{j=1}^n c_j x_j \quad (SCP-IP)$$

$$\text{subject to} \quad \sum_{j=1}^n a_{ij} x_j \geq 1 \quad \forall i = 1, \dots, m \quad (2.1a)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (2.1b)$$

The objective function is to minimize cost of all selected columns of A . Constraints in group 2.1a ensures that each row of A must be covered by at least one column of A . The last constraint implies that the decision variables are binary.

2.1.2 Solution Methodologies for the Set Covering Problem

In this subsection, we review different solution methods proposed in the open literature for the SCP. Exact algorithms are mostly based on branch-and-bound (B&B) and branch-and-cut (B&C). However, these algorithms are time consuming and can solve instances of relatively moderate size. That is why many research efforts have been worked on designing heuristics in order to find near-optimal solutions for large instances within a reasonable execution time. We first explain exact algorithms, and then we briefly review several effective heuristic algorithms.

2.1.2.1 Exact Algorithms

The most effective exact methods for solving the model $SCP-IP$ are B&B algorithms in which using LP relaxation of the model $SCP-IP$, lower bounds are computed. To the best of our knowledge, the algorithm of Nobili and Sassamo [55] is the only one that could obtain lower bounds that are stronger than the value of LP relaxation of the model $SCP-IP$. In their algorithm, they investigate the structural properties of the SCP polytope, and find several cutting planes, and add them to their B&C algorithm.

Beasley [11] proposes an algorithm that solves LP relaxation only at the root node of the B&B tree. At all other nodes of the tree, his algorithm finds the lower bounds using Lagrangian relaxation along with subgradient optimization which starts with the Lagrangian multipliers that are associated with the parent node. At the root node, Lagrangian multipliers are calculated by applying a dual ascent procedure and a subgradient procedure. Then a reduction procedure is applied, and then the LP relaxation of the reduced problem is solved to optimality by dual simplex method. Feasible solutions are computed by applying simple greedy heuristics and several dominance properties to reduce the number of rows and columns of the set covering matrix A . The branching rule chooses the uncovered row with the highest Lagrangian multiplier, and fixes the variable corresponding to the column with smallest Lagrangian cost that covers the selected uncovered row to zero and to one.

Beasley and Jornsten [14] give an enhancement of the algorithm of Beasley [11]. A main difference is that Beasley and Jornsten [14] add some extra constraints to the problems. Two of these constraints are added to exclude the best feasible solution found at the end of the subgradient procedure at the root node. They impose these constraints so that at least one of the selected columns in this solution must be unselected. After adding these constraints, they solve the LP relaxation as in [11], and they then add classical Gomory fractional cuts. Another

enhancement as regards [11] is that [14] uses the Lagrangian heuristic of [16] and a different branching rule. They computationally show that their proposed algorithm performs better than the algorithm proposed in [11] in most of the benchmark instances with respect to execution time.

Balas and Carrera [5] propose the most recent exact algorithm for solving the model *SCP-IP*. Their algorithm is based on inserting a dynamic subgradient optimization into a B&B algorithm. They use two binary branching strategies based on dual information corresponding to Lagrangian multipliers in order to find a better (larger) lower bound at both descendant nodes of the tree. They finally show that their proposed algorithm is faster than the algorithm developed in [11] and [14] in all benchmark instances.

Recently, Caprara et al. [22] present an experimental comparison of all above-mentioned algorithms and general ILP solvers on the benchmark instances, and show that if we give the instance directly to the solvers CPLEX 4.0.8 and MINTO 2.3, the average execution time will be less than the execution time of all previously proposed algorithms. According to their computational experiment, the average execution time is about 3000 seconds for the algorithm in [11], 2000 seconds for the algorithm in [14], 1500 seconds for the algorithm in [5], 700 seconds for CPLEX, and 750 seconds for MINTO, on several classes of instances in the Beasley's OR Library [12].

2.1.2.2 Inexact Algorithms

Various heuristic approaches are proposed to find near-optimal solutions of the model *SCP-IP*.

Yagiura et al. [78] propose a local search algorithm for the SCP with three characteristics: 3-flip neighborhood, searching the infeasible region, and using the information from the Lagrangian relaxation to reduce the size of the problem. Kinney et al. [42] present a reactive tabu search algorithm with variable clustering for the unicast SCP. Then, Caserta [23] proposes a dynamic primal-dual algorithm and applies the tabu search based on a primal intensive scheme with a Lagrangian-based dual intensive scheme.

Different relaxation heuristics are designed for the SCP [72]. Lan et al. [44] develop a heuristic called Meta-RaPS (Meta-heuristic for Randomized Priority Search) for solving the model *SCP-IP*. In their algorithm, they generate feasible solutions by introducing random factors into a construction method. Then they try to improve the feasible solutions by applying an improvement heuristic. In addition to applying the basic Meta-RaPS, the heuristic developed herein integrates the elements of randomizing the selection of priority rules, penalizing the worst columns when the searching space is highly condensed, and defining the core problem to speed up the algorithm. Bautista and Pereira [10] propose a GRASP (Greedy randomized adaptive search procedure) algorithm incorporating a local improvement procedure based on

the heuristics to solve the model *SCP-IP*.

One of the most successful and most recent heuristic approaches is proposed by Wang et al. [75]. They enhance the electromagnetism metaheuristic method (EM), proposed in [52], so that the algorithm first generates a pool of solutions, and then it applies a fixed number of local search and movement iterations, and it also uses mutations to further escape the local optima. Their results show that their proposed method outperforms most previous heuristic approaches.

Other approaches for solving the model *SCP-IP* include ABC (Artificial Bee Colony) [69, 24], ACO (Ant Colony Optimization) [50, 63], GA (Genetic Algorithm) [13].

2.2 Maximum Stable Set Problem

A stable set in a graph $G(V, E)$ is a subset S of the node set V such that no two nodes of S are adjacent. We call two nodes adjacent if there is an edge in E connecting them. The maximum stable set problem (MSS) asks for the stable set of maximum cardinality. In the open literature, the MSS is also called as independent set problem or vertex packing problem. If there is a distinct weight assigned to each node in V , then the graph is weighted. In this case, maximum stable set problem asks for the stable set of maximum weight. Garey and Johnson [30] show that it is NP-hard to determine the maximum stable set in an arbitrary graph. Practical applications of the maximum stable set problem include economics [3], computer vision [7], fault tolerance [15], coding theory [45, 68], and aligning DNA and protein sequences [47]. In the next subsections, we give the mathematical model of MSS and a brief overview of the solution methods.

2.2.1 Integer Programming Formulation

In this subsection, we present the integer programming model for the problem MSS in the literature. Considering the graph $G(V, E)$, for each node $v_j \in V$, we define the decision variable $x_j = 1$ if node v_j is selected to be in the optimal subset, and $x_j = 0$ otherwise. Following is an IP model for the problem MSS.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^n w_j x_j & (MSS-IP) \end{aligned}$$

$$\text{subject to} \quad x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in E \quad (2.2a)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (2.2b)$$

The objective function is to maximize weight of all selected nodes of G . Constraints in group 2.2a ensure that for each edge in E , at most one endnode can be in a stable set. The last constraint implies that the decision variables are binary.

2.2.2 Brief Overview of Solution Methodologies for the Maximum Stable Set Problem

Over the past few decades, various heuristic and exact solution methods are proposed for solving the maximum stable set problem.

The popular exact methods include branch and bound [58, 35], branch and price [76], continuous formulation approaches [34], global optimization [57], Lagrangian relaxation [20], semidefinite programming [26], and branch and cut [62].

As an alternative approach, a number of heuristic methods have been proposed to find near-optimal solutions to the MSS problem. Those approaches include local search [8, 38], tabu search [32], variable depth search [41], quadratic optimization [17], and genetic search [67].

For a more detailed categorization of exact and inexact algorithms for solving the model *MSS-IP*, see [77] in which a comprehensive review on algorithms for solving the model *MSS-IP* is provided by Wu and Hao in 2015.

2.3 Other Problems with Conflict Constraints

Conflict relations arise in many contexts of combinatorial optimization and integer programming so that various problems have been studied under conflict constraints. To the best of our knowledge, Balas [6] in 1979 introduced Disjunctive Programming framework. Nevertheless, classic problems with conflict constraints were studied recently.

Atamtork et al. [2] investigate the use of conflict graphs in solving integer programming problems. They introduce how to construct and manage dynamically changing conflict graphs, and discuss pre-processing and cut generation methods in the presence of conflict graphs.

Bin packing problem with conflict graphs (BPCG) is firstly introduced by Jansen and Ohring [40] in 1995. The authors propose seven different approximation algorithms and analyze worst-case approximation ratios for special graph classes. In 2004, Gendreau et al. [31] present six heuristics for BPCG and perform extensive computational experiments. In 2008, Epstein and Levin [28] improve one of the algorithms of [40] which is designed for perfect graphs. In 2010, Muritiba et al. [51] give the first exact algorithm for solving BPCG in a branch and price framework. They compute lower bounds based on the clique structure, on a matching and on a surrogate relaxation over all constraints, and upper bounds by applying a population based heuristic framework with a tabu list. Sadykov and Vanderbeck [65], in 2012, propose

a new branch and price algorithm for solving BPCG in which they solve conflict-constrained knapsack subproblems with dynamic programming.

Knapsack problem with conflict constraints (KCG) is introduced by Yamada et al. [79] in 2002, and the authors propose a greedy algorithm for solving the problem. Hifi and Michrafy [36], in 2006, present a reactive local search algorithm combined with a tabu list for the solving KCG, and in 2007 [37], they propose a branch and bound algorithm where they apply LP relaxation to compute upper bounds and heuristics of [36] to compute lower bounds. They also propose reduction techniques to eliminate redundant items before starting the branch and bound algorithm. In 2009, Pferschy and Schauer [59] present pseudopolynomial algorithms for three special classes of conflict graphs in KCG.

Minimum spanning tree problems with conflict constraints (MSTCC) is introduced by Darmann et al. [25] in 2009. In 2011, Zhang et al. [80] propose special cases in which the MSTCC is polynomially solvable, and present two heuristic algorithms for solving the problem. In 2015, Samer and Urrutia [66] present a branch and cut algorithm for solving the MSTCC, and they computationally show that their algorithm outperforms the algorithm of [80] on all the benchmark instances.

In 2013, Pferschy and Schauer [60] discuss the complexity of maximum flow problems with conflict and forcing constraints. In the same year, Oncan et al. [56] propose a Lagrangean relaxation scheme and heuristics for minimum cost perfect matchings with conflict constraints, and investigate the polynomially solvable cases.

Chapter 3

Pre-processing Procedures

In this chapter, we discuss several procedures to remove redundancies, remove unnecessary variables, and tighten up the set of constraints for the IP model *SCC-IP*. We refer to these procedures as pre-processing. The only row reduction and the first column reduction that we employ are similar to those previously proposed for the SCP, and we employ it without change. We also introduce and employ two new column reductions for the problem SCC. We then carry out a comprehensive computational experiment to assess the efficiency of these pre-processing procedures.

For convenience and ease of discussions, we refer to each row of the matrix A as an element and to each column of this matrix as a node. Element i is covered by node j if $a_{ij} = 1$. We use $M = \{1, 2, \dots, m\}$ to represent the set of all elements and the set $N = \{1, 2, \dots, n\}$ to represent the set of all nodes. We also define the subsets N_i of N for each $i \in M$, subsets M_j of M for each $j \in N$, and subsets G_j of N for each $j \in N$ as follows.

- $N_i = \{j : a_{ij} = 1, j \in N\}$
- $M_j = \{i : a_{ij} = 1, i \in M\}$
- $G_j = \{k : (j, k) \in F, k \in N\}$

N_i is the collection of nodes that cover element i . We refer to N_i as the covering node-set for element i . Similarly, M_j is the collection of elements covered by node j , and we refer to it as the cover-set of node j . G_j is the collection of nodes that are in conflict with node j .

A conflict-free collection of nodes form a *feasible cover* if the union of their corresponding cover-sets is the entire set M . Every feasible solution of the problem SCC is associated with a feasible cover (nodes j with the corresponding value of $x_j = 1$).

3.1 Row Reduction

To do a row reduction, we use the first row reduction method defined for the SCP in [11], as follows.

- **Row reduction #1:** For each element i ($i \in M$), if there exists another element r ($r \in M, r \neq i$) such that $N_r \subseteq N_i$, then a^i (i^{th} row of the matrix A) can be deleted from the problem, since any column used to cover a^r automatically covers a^i .

3.2 Column Reduction

In this subsection, we discuss three column reduction methods. The column reduction #1 is defined for the SCP in [11], and we explain it here. The column reductions #2 and #3 are new column reductions that we define for the SCC.

- **Column reduction #1:** For each element i ($i \in M$), if it is only covered by a single node j , then the node j must be in every feasible cover. Computationally, if we identify node j that has to be in every feasible cover, all the rows corresponding to the elements in M_j are covered hence need not be considered (i.e., the corresponding rows can be removed from the constraint set). Note therefore that we need to add c_j to the optimal objective function value, after the problem is solved. Also, the variables corresponding to nodes in G_j must be set to zero, since node j is now selected to be in every feasible cover, and those nodes that are in conflict with it cannot be in any feasible cover anymore. The collection F must be updated accordingly.
- **Column reduction #2:** For each node j ($j \in N$), if there exists another node k ($k \in N, k \neq j$) such that $M_j \subseteq M_k$, i.e., all elements covered by node j are also covered by node k , and $c_j \geq c_k$, and $[G_k \setminus \{j\}] \subseteq [G_j \setminus \{k\}]$, i.e., each node that is in conflict with node k is also in conflict with node j , then we fix the variable x_j to zero. This is due to the fact that under these conditions for every solution in which $x_j = 1$ there exists another solution with equal or smaller objective function value in which $x_j = 0$ and $x_k = 1$ (keeping all other variables unchanged).
- **Column reduction #3:** For each node j ($j \in N$), if there exists an element i ($i \in M$) such that $N_i \subseteq G_j$, i.e., each node that covers element i is in conflict with node j , we fix the variable x_j to zero, since choosing node j to be in any feasible cover would make it impossible to cover element i .

3.3 A Computational Experiment

In this section we present the results of a computational experiment in which we evaluate the effectiveness of the pre-processing procedures on the computational requirements of the branch and bound algorithm for solving the model *SCC-IP*. In Subsection 3.3.1, we describe the data sets that we use in this experiment and in Subsection 3.3.2, we present our findings.

In this dissertation, we write all programs in C#, and for solving the problem SCC, the callback mechanism in the Concert API of CPLEX 12.6.1 is used. We consider a numerical precision of 10^{-5} , even when looking for violated constraints. Experiments are carried on a machine with an Intel Core TM2 (2.66GHz) CPU, with 4GB of RAM. We set the absolute MIP gap tolerance parameter of CPLEX to 0.9999.

In this experiment, we turn off all pre-processing, heuristics and cut generation options of CPLEX, so as to assess the effectiveness of the proposed pre-processing procedures without interference with the effects of other methods.

3.3.1 Data sets

An instance of the SCC is identified by the set covering matrix A , the conflict collection F , and cost vector c . The set covering matrix A has m rows and n columns, and each entry of the matrix is either 0 or 1. We construct and solve instances of the problem with different values of m and n . For each instance in our experiment, we construct the set covering matrix A and the conflict collection F randomly as we describe below.

To construct the cost vector c , we can fix every entry of the cost vector c to 1, and refer to the resulting instances as the unicast instances. Alternatively, we can use randomly generated numbers as entries of the cost vector c (weighted instances). Our pilot studies indicated that unicast instances are harder instances compared to weighted instances with respect to the execution time of the B&B and thus we construct unicast instances in our first two data sets. For our third data set, we use the weighted instances that are already constructed in the open literature.

We construct three collections of instances of the SCC as follows.

Data set 1: 10-D geometrical instances (GEO-10D)

In this data set, we have a collection of m demand centers (rows) and a collection of n potential service centers (columns). We assume that there is a 10-dimensional characteristic vector associated with each demand center or potential service center. Let $x_1^i, x_2^i, \dots, x_{10}^i$ denote the characteristics of demand center i and $y_1^j, y_2^j, \dots, y_{10}^j$ denote the characteristics of potential service center j . We obtain the value of each element of the characteristic vector for each demand

center or service center using a uniform distribution in the interval $[0, 1000]$. We then assume that a demand center i is covered by a potential service center j , if their corresponding Euclidean distance $d_{ij} = \sqrt{\sum_{l=1}^{10} (x_l^i - y_l^j)^2}$ is less than a given parameter r_s . For each demand center i and each potential service center j , we have $a_{ij} = 1$ if $d_{ij} \leq r_s$, and $a_{ij} = 0$ otherwise, for $i = 1, \dots, m$ and $j = 1, \dots, n$. This defines the set covering matrix A .

For the conflict collection F , we assume that two potential service centers j and k represent a conflict if their corresponding distance $d_{jk} = \sqrt{\sum_{l=1}^{10} (y_l^j - y_l^k)^2}$ is less than a given parameter r_p , i.e., $(j, k) \in F$ if $d_{jk} \leq r_p$, and $(j, k) \notin F$ otherwise for each $j = 1, \dots, n-1$ and each $k = j+1, \dots, n$. For any value of r_p , this defines all elements of the conflict collection F .

This data set is inspired by the application of SCC in the context of the wireless communication location problem that we described in Chapter 1. We use the 10-dimensional space for this data set, since our preliminary experiments showed that the corresponding models are relatively more challenging to solve, as compared with the data set in the 2-dimensional space.

Within this data set, we first construct characteristic vectors for a collection of 100 demand centers, a collection of 300 demand centers, and a collection of 1000 potential service centers, and we denote to these collections of characteristic vectors as demand collection D100, demand collection D300, and service collection S1000, respectively. We then construct two pairs of demand and service collections as follows; first, we consider the demand collection D100 and the service collection S1000 as a pair of demand and service collections, and we refer to this pair as demand-service data set I; next, we consider the demand collection D300 and the service collection S1000 as a pair of demand and service collections, and we refer to this pair as demand-service data set II.

For each demand-service data set, we determine the distance between every demand center and every potential service center ($\frac{mn}{2}$ values). In columns r_{ds}^{min} , r_{ds}^{med} , and r_{ds}^{max} of Table 3.1, we report the minimum, the median, and the maximum among these values. Similarly, for every pair of potential service centers in each demand-service data set, we determine the corresponding distance ($\frac{n(n-1)}{2}$ values), and in columns r_{ss}^{min} , r_{ss}^{med} , and r_{ss}^{max} of Table 3.1, we report the minimum, the median, and the maximum values as obtained.

Table 3.1: The information of demand-service data sets I and II

Demand-Service Data Set	Demand Collection	Service Collection	r_{ds}^{min}	r_{ds}^{med}	r_{ds}^{max}	r_{ss}^{min}	r_{ss}^{med}	r_{ss}^{max}
I	D100	S1000	298.2	1278.2	2239.1	276.1	1272.9	2281.1
II	D300	S1000	262.2	1268.8	2270.7	276.1	1272.9	2281.1

Now, we generate several instances of the problem SCC by considering demand centers and potential service centers in each demand-service data set and changing the values of parameters r_s and r_p . We report all parameter values of the instances that we construct in this data set in Table 3.2.

Table 3.2: The parameters of instances in GEO-10D data set

ID	Demand-Service Data Set	m	n	r_s	r_p
G10.1	I	100	1000	850	700
G10.2				850	850
G10.3				850	950
G10.4				850	1050
G10.5				950	700
G10.6				950	850
G10.7				950	950
G10.8				950	1050
G10.9				1050	850
G10.10				1050	1050
G10.11	II	300	1000	850	700
G10.12				850	850
G10.13				950	700
G10.14				950	850
G10.15				950	950
G10.16				1050	850
G10.17				1050	950
G10.18				1050	1050

Data set 2: Random instances (RAN)

Instances in this data set are constructed as follows. Each entry of the $m \times n$ matrix A is randomly chosen as 1 with the probability of s and 0 with the probability of $1 - s$, where s is a given parameter in the interval $[0, 1]$. For the conflict collection F , for each $j = 1, \dots, n - 1$ and each $k = j + 1, \dots, n$, we add (j, k) to collection F with the probability of p , where p is a given parameter in the interval $[0, 1]$.

We report all parameter values for the instances that we construct in this data set in Table 3.3.

Table 3.3: The parameters of instances in RAN data set

ID	m	n	s	p
R.1	100	300	0.05	0.01
R.2			0.05	0.05
R.3			0.05	0.1
R.4			0.1	0.05
R.5			0.1	0.1
R.6			0.1	0.2
R.7	100	1000	0.01	0.01
R.8			0.01	0.05
R.9			0.05	0.01
R.10			0.05	0.05
R.11			0.1	0.01

Data set 3: OR-Library instances (OR-L)

In this data set, we use two instances that are already constructed for the set covering problem in the OR-Library [12]. For each of these two instances, we have the set covering matrix and the cost vector from the OR-Library [12], and we construct several conflict collections randomly similar to the data set RAN.

We report all parameter values for the instances that we construct in this data set in Table 3.4. The first column shows the ID that we define for each instance, and the second and third columns show the size of each instance. The fourth column shows the index of the instance from OR-Library [12], and the fifth column shows the density of the set covering matrix in the corresponding instance. The last column represents the probability of each pair of nodes being added to the conflict collection in the corresponding instance.

Table 3.4: The parameters of instances in OR-L data set

ID	m	n	OR-L Index	s	p
L.1	200	1000	scp41	0.02	0.02
L.2	200	1000	scp61	0.05	0.02
L.3					0.03
L.4					0.04
L.5					0.05

3.3.2 A Comparative Study

We study the impact of pre-processing procedures on the computational effort required to solve the model *SCC-IP* for the instances that we described above. To this end, we consider two scenarios. In the first scenario, we solve the instance via B&B method (i.e., without imposing pre-processing procedures). In the second scenario, we first impose pre-processing procedures on the instance, and then we solve the reduced instance via B&B method. Then we compare the two scenarios with respect to their corresponding execution times. In Table 3.5, for a representative selection of these instances, we report these results along with the information for each reduced instance (such as number of eliminated rows and fixed columns). Results for the remaining instances follow the same pattern, and for brevity, we refrain from reporting the results for the remaining instances. In this table the first column shows the instance ID, which is consistent with Tables 3.2 and 3.3; the next four columns show the parameter values of each instance, the sixth column shows the total number of conflict constraints for each corresponding instance, the seventh column (Total time 1) corresponds to the execution time of the B&B when solving the model *SCC-IP* (pre-processing procedures are not imposed). The eighth and ninth columns show the number of rows that are eliminated and the number of columns that are fixed after imposing pre-processing procedures, respectively. The tenth column (IP time 2) shows the execution time of the IP model after imposing pre-processing procedures, in seconds, and the last column (Total time 2) shows the summation of execution time of imposing pre-processing procedures and the B&B tree for solving the reduced instance, in seconds. In this experiment, we impose a time limit of 7200 seconds on all instances, and if the execution time of an instance exceeds this time limit, we abort the program.

We can make several observations from the results reported in Table 3.5;

1. **Impact of number of conflicts.** For fixed values of m , n , and r_s in GEO-10D instances, number of fixed columns tends to increase as value of r_p grows, i.e., as the number of conflicts increases, the pre-processing procedures performs a stronger reduction. Same observation is made for RAN instances, as value of p grows, and values of m , n , and s are fixed. For the OR-L instances, the pre-processing procedures could not fix any column. The execution time of solving the model *SCC-IP* with and without pre-processing procedures also tend to increase significantly as we increase the number of conflicts while other parameters are fixed, for instances in all data sets.
2. **Impact of covering percentage (r_s and s).** For fixed values of m , n , and r_p in GEO-10D instances, the number of eliminated rows and the number of fixed columns decrease as value of r_s grows, which means that the pre-processing procedures perform a weaker reduction. Also, for fixed values of m , n , and p , the number of eliminated rows and fixed

Table 3.5: Data sets instances reduction using the pre-processing procedures

ID	m	n	s or r_s	p or r_p	# of confs.	B&B	B&B w/ Prep.			
						Total time 1 (sec)	# of elim. rows	# of fixed cols	IP time 2 (sec)	Total time 2 (sec)
G10.2	100	1000	850	850	22,780	37.6	0	36	17.1	17.7
G10.3			850	950	49,061	299.1	0	46	276.4	277.4
G10.4			850	1050	92,397	2,722.2	0	111	2,228.2	2,230.0
G10.6			950	850	22,780	5.8	0	3	4.9	5.6
G10.11	300	1000	850	700	5,366	60.7	12	1	55.8	56.5
G10.12			850	850	22,780	-	12	13	-	-
G10.14			950	850	22,780	793.8	3	0	605.2	606.1
R.1	100	300	0.05	0.01	466	2.6	0	4	2.3	2.3
R.2			0.05	0.05	2,312	32.3	0	3	32.4	32.5
R.3			0.05	0.1	4,615	339.3	0	3	362.3	362.4
R.7	100	1000	0.01	0.01	5,118	0.7	0	378	0.3	0.7
R.8			0.01	0.05	25,393	1,759.4	0	370	499.7	500.3
R.9			0.05	0.01	5,118	441.6	0	6	424.1	424.6
R.10			0.05	0.05	25,393	3,591.0	0	6	3,449.0	3,449.8
L.1	200	1000	0.02	0.02	10,175	147.1	0	0	147.1	147.3
L.2			0.05	0.02	10,175	15.7	0	0	15.7	16.0
L.3			0.05	0.03	15,242	80.6	0	0	80.6	81.0

columns decrease as value of s grows in RAN instances. For the OR-L instances, the pre-processing procedures could not eliminate any row from the problem. The execution time of solving the model *SCC-IP* with and without pre-processing procedures also tend to decrease significantly as we increase r_s and s (covering percentage) while other parameter values are fixed for the instances in all data sets. In other words, as the set covering matrix A becomes more dense, the corresponding instance becomes somewhat easier to solve, and the impact of the pre-processing procedures get smaller.

3. **Impact of ratio of n to m .** In the GEO-10D instances, for fixed values of n , r_p , and r_s , the number of fixed columns decreases as value of m grows, which means that the pre-processing procedures performs a weaker reduction. Same observation is valid for RAN instances, as value of n decreases and values of m , p , and s are fixed.
4. **Execution time comparison.** For 12 out 14 instances in data sets GEO-10D and RAN, total execution time of the second scenario, where we perform the pre-processing procedures, is smaller than the execution time of the model *SCC-IP* without pre-processing, but the difference between execution times is not significant in most of these 12 instances. On average, we observe a 20.5% reduction in execution time of the model *SCC-IP* after performing pre-processing procedures, in GEO-10D and RAN instances. We also observe that for the OR-L instances, the pre-processing procedures are not effective.

Chapter 4

Valid Inequalities

Feasible region of the model $SCC-IP$ consists of the integer points in the intersection of two well-known polyhedra. One polyhedron is associated with the constraints of the set covering problem (set covering polyhedron) and the other polyhedron is associated with the conflict constraints (stable set polyhedron). These polyhedra are separately studied in the open literature, and appropriate valid inequalities (VIs) for each polyhedron are introduced. Naturally, these VIs are also valid for the model $SCC-IP$. In Sections 4.1 and 4.2, we describe two such families of VIs that we found to be effective in the context of the problem SCC.

The presence of both groups of constraints, in the model $SCC-IP$, however, allows for introduction of new families of VIs that are potentially more effective in this context. To the best of our knowledge, only two articles in the open literature address this subject. One article is by Kou et al. [43] in which the authors study the polyhedral structure of the mixed set covering, packing, and partitioning problem and develop new classes of VIs. They then identify sufficient conditions for their proposed VIs to be facet-defining. Our pilot studies showed that those VIs are not effective in solving the model $SCC-IP$. So we do not explain those VIs in this context. The second article is by Kazemi Tutunchi and Fathi [71] in which a family of VIs for this problem is introduced. They introduce these VIs in the context of solving a bi-objective discrete optimization problem, in order to enhance the computational effectiveness of the B&B algorithm in that context. In Sections 4.3 to 4.6 we discuss these VIs along with several families of new VIs that we introduce in this context.

In Chapter 5, through several limited computational experiments, we evaluate the effectiveness of these VIs in solving the model $SCC-IP$.

4.1 Maximal Clique Valid Inequalities

In this section, we describe a family of valid inequalities called maximal clique valid inequalities (MC-VIs) for the model *SCC-IP*. Maximal clique VIs are defined for the stable set problem on the conflict graph of the problem SCC (as defined in Chapter 1), as follows.

Proposition 1. *For each $q \in V$, where q is a maximal clique in G , the constraint*

$$\sum_{j:v_j \in q} x_j \leq 1 \tag{4.1}$$

*is a valid inequality for the model *SCC-IP*.*

Rebennack et al. [62] show that clique inequalities are valid cuts for the maximum stable set polytope, and the proof will not be repeated here. It is also shown that when the clique is maximal, the clique inequality is facet-defining for the convex hull of the maximum stable set polytope. Hence MC-VIs can be added to the constraints set of model *SCC-IP* as valid inequalities to tighten the LP feasible region.

4.2 Odd-Cycle Valid Inequalities

The following proposition describes a family of valid inequalities called odd-cycle valid inequalities for the model *SCC-IP*.

Proposition 2. *For each $C \subseteq G$, where $C := (V_C, E_C)$ is an odd-cycle in G , the constraint*

$$\sum_{j:v_j \in V_C} x_j \leq \frac{|V_C| - 1}{2} \tag{4.2}$$

*is a valid inequality for the model *SCC-IP*.*

Odd-cycle valid inequalities (OCI) are shown to be valid inequalities for the stable set polytope in Gerards and Schrijver [33], and they are shown to be effective. Hence, the constraint set #4.2 is a set of valid inequalities for the model *SCC-IP*.

4.3 Covering-Packing Valid Inequalities

In this section, we explain a family of valid inequalities called Covering-Packing Valid Inequalities (CP-VIs) proposed by Kazemi Tutunchi and Fathi [71] for the model *SCC-IP*, and we discuss their facetial properties. We then introduce another set of valid inequalities which we refer to as generalized CP-VIs.

4.3.1 Definitions and Properties

Let us define $N_i^k = \{j : j \in N_i, (j, k) \notin F\}$ for each element $i \in M$ and node $k \in N \setminus N_i$. The following proposition explains the CP-VIs.

Proposition 3. *For each element $i \in M$ and node $k \in N \setminus N_i$, the constraint*

$$\sum_{j:j \in N_i^k} x_j \geq x_k \quad (4.3)$$

is a valid inequality for the model $SCC-IP$.

The CP-VI states that for each element i and each node k that does not cover element i , if node k is selected to be in a feasible cover, at least one of those nodes that cover element i and do not have conflict with node k must be selected to be in that feasible cover. Kazemi Tutunchi and Fathi [71] show that the CP-VIs can remove a segment of the feasible region of the LP relaxation for the model $SCC-IP$, and through a computational experiment, they showed that these VIs can be quite effective in solving some instances of this problem.

4.3.2 Facet-defining Properties of CP-VIs

In general a CP-VI is not necessarily a facet-defining VI for the convex hull of the feasible solutions for the model $SCC-IP$. But this VI can be facet-defining if the constraints set of the problem satisfies a sufficient condition, i.e., it has a special structure.

To show that a CP-VI is not always a facet-defining VI for the convex hull of the feasible solutions for the model $SCC-IP$, we construct a small instance of the problem SCC, as follows.

Example 1. Consider an instance of the problem SCC shown by the following IP model with $m = 1$ covering inequalities and one conflict constraint.

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^3 x_j && (EX1-IP) \\ & \text{subject to} && x_1 + x_2 \geq 1 \\ & && x_2 + x_3 \leq 1 \\ & && x_j \in \{0, 1\} \quad \forall j = 1, \dots, 3 \end{aligned}$$

The model $EX1-IP$ has only 4 feasible solutions, as follows.

$$\mathbf{t}^1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \quad \mathbf{t}^2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$$

$$\mathbf{t}^3 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T \quad \mathbf{t}^4 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T$$

One can easily verify that the vectors $\mathbf{t}^1, \dots, \mathbf{t}^4$ are affinely independent. Thus the convex hull of the feasible solutions for the model *EX1-IP* is full-dimensional.

The model *EX1-IP* has only one CP-VI as follows.

$$x_1 \geq x_3 \tag{4.4}$$

In order for the inequality 4.4 to be a facet-defining VI for the model *EX1-IP*, there must exist 3 feasible and affinely independent vectors that satisfy the inequality 4.4 at equality. But we observe that only $\mathbf{t}^2$ and $\mathbf{t}^4$ satisfy the inequality 4.4 at equality. Therefore, the inequality 4.4 is not a facet-defining VI for the model *EX1-IP*. $\square$

Through Example 1, we showed that in general a CP-VI is not a facet-defining VI for the convex hull of the feasible solutions for the model *SCC-IP*. But as mentioned before, this VI can be facet-defining if the constraints set of the problem has a special structure. To show this, we construct another small instance of the problem SCC, as follows.

Example 2. Consider an instance of the problem SCC shown by the following IP model with $m = 4$ covering inequalities and two conflict constraint.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^7 x_j && (EX2-IP) \\ \text{subject to} \quad & x_1 + x_3 + x_4 \geq 1 && (\text{element 1}) \\ & x_2 + x_3 + x_4 \geq 1 && (\text{element 2}) \\ & x_5 + x_6 \geq 1 && (\text{element 3}) \\ & x_5 + x_7 \geq 1 && (\text{element 4}) \\ & x_3 + x_5 \leq 1 \\ & x_4 + x_5 \leq 1 \\ & x_j \in \{0, 1\} \quad \forall j = 1, \dots, 7 \end{aligned}$$

According to the definition of CP-VIs, and by setting $i = 1$ and $k = 5$, we get the following CP-VI,

$$x_1 \geq x_5, \tag{4.5}$$

and by setting $i = 2$ and $k = 5$, we get the following CP-VI.

$$x_2 \geq x_5. \tag{4.6}$$

Observation 1. *The inequalities 4.5 and 4.6 are facet-defining VIs for the model EX2-IP.*

Proof. First, we show that the convex hull of the feasible solutions for the model EX2-IP is full-dimensional. To this end, consider the following 8 vectors.

$$\begin{aligned} \mathbf{y}^1 &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}^T & \mathbf{y}^2 &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}^T \\ \mathbf{y}^3 &= \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}^T & \mathbf{y}^4 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T \\ \mathbf{y}^5 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^T & \mathbf{y}^6 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T \\ \mathbf{y}^7 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T & \mathbf{y}^8 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T \end{aligned}$$

Let $Y = \{\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3, \mathbf{y}^4, \mathbf{y}^5, \mathbf{y}^6, \mathbf{y}^7, \mathbf{y}^8\}$. One can easily verify that the 8 vectors in Y are feasible for the model EX2-IP, and that these vectors are also affinely independent. Since the solution space for this model is 7-dimensional, it follows that the convex hull of the feasible solutions for Example 2 is full-dimensional.

We observe that with the exception of $\mathbf{y}^7$, every vector in Y satisfies the CP-VI 4.5 at equality. Thus we have 7 feasible solutions for the model EX2-IP that are affinely independent and satisfy the inequality 4.5 at equality, hence the inequality 4.5 is a facet-defining VI for the model EX2-IP.

Similarly, to show that the inequality 4.6 is a facet-defining VI for the model EX2-IP, consider the vectors in Y with the exception of $\mathbf{y}^8$. Thus we have 7 feasible solutions for the model EX2-IP that are affinely independent and satisfy the inequality 4.6 at equality, hence the inequality 4.6 is also a facet-defining VI for the model EX2-IP. $\square$

Using the structure of constraints in Example 2, we can construct larger instances for which CP-VIs are facet-defining VIs. To illustrate, we construct a larger instance of the SCC with 12 elements and 21 nodes, and we then show that CP-VIs are facet-defining VIs for its IP model, as follows.

Example 3. Consider an instance of the problem SCC shown by the following IP model with $m = 12$ covering inequalities and 12 conflict constraints.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^{21} x_j && (EX3-IP) \\
& \text{subject to} && x_1 + x_3 + x_4 \geq 1 && \text{(element 1)} \\
& && x_2 + x_3 + x_4 \geq 1 && \text{(element 2)} \\
& && x_5 + x_6 \geq 1 && \text{(element 3)} \\
& && x_5 + x_7 \geq 1 && \text{(element 4)} \\
& && x_3 + x_5 \leq 1 \\
& && x_4 + x_5 \leq 1 && x_4 + x_{11} \leq 1 \\
& && && x_5 + x_{10} \leq 1 \\
& && x_8 + x_{10} + x_{11} \geq 1 && \text{(element 5)} && x_3 + x_{12} \leq 1 \\
& && x_9 + x_{10} + x_{11} \geq 1 && \text{(element 6)} \\
& && x_{12} + x_{13} \geq 1 && \text{(element 7)} \\
& && x_{12} + x_{14} \geq 1 && \text{(element 8)} \\
& && x_{10} + x_{12} \leq 1 \\
& && x_{11} + x_{12} \leq 1 && x_{11} + x_{18} \leq 1 \\
& && x_{15} + x_{17} + x_{18} \geq 1 && \text{(element 9)} && x_{12} + x_{17} \leq 1 \\
& && x_{16} + x_{17} + x_{18} \geq 1 && \text{(element 10)} && x_{10} + x_{19} \leq 1 \\
& && x_{19} + x_{20} \geq 1 && \text{(element 11)} \\
& && x_{19} + x_{21} \geq 1 && \text{(element 12)} \\
& && x_{17} + x_{19} \leq 1 \\
& && x_{18} + x_{19} \leq 1 \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, 21
\end{aligned}$$

We get the following six CP-VIs for this model.

$$x_1 \geq x_5 \quad \text{for element } i = 1 \text{ and node } k = 5 \quad (4.7)$$

$$x_2 \geq x_5 \quad \text{for element } i = 2 \text{ and node } k = 5 \quad (4.8)$$

$$x_8 \geq x_{12} \quad \text{for element } i = 5 \text{ and node } k = 12 \quad (4.9)$$

$$x_9 \geq x_{12} \quad \text{for element } i = 6 \text{ and node } k = 12 \quad (4.10)$$

$$x_{15} \geq x_{19} \quad \text{for element } i = 9 \text{ and node } k = 19 \quad (4.11)$$

$$x_{16} \geq x_{19} \quad \text{for element } i = 10 \text{ and node } k = 19 \quad (4.12)$$

Observation 2. *The inequalities 4.7 to 4.12 are facet-defining VIs for the model EX3-IP.*

Proof. First, we show that the convex hull of the feasible solutions for the model EX3-IP is full-dimensional. To this end, we need to find 22 feasible vectors that are affinely independent. Using the 8 vectors that we defined in Observation 1, we construct the following 22 vectors.

$$\begin{aligned}
\mathbf{u}^1 &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T \\
\mathbf{u}^2 &= [\mathbf{y}^{2T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^9 &= [\mathbf{y}^{1T} \ \mathbf{y}^{2T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{16} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{2T}]^T \\
\mathbf{u}^3 &= [\mathbf{y}^{3T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{10} &= [\mathbf{y}^{1T} \ \mathbf{y}^{3T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{17} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{3T}]^T \\
\mathbf{u}^4 &= [\mathbf{y}^{4T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{11} &= [\mathbf{y}^{1T} \ \mathbf{y}^{4T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{18} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{4T}]^T \\
\mathbf{u}^5 &= [\mathbf{y}^{5T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{12} &= [\mathbf{y}^{1T} \ \mathbf{y}^{5T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{19} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{5T}]^T \\
\mathbf{u}^6 &= [\mathbf{y}^{6T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{13} &= [\mathbf{y}^{1T} \ \mathbf{y}^{6T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{20} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{6T}]^T \\
\mathbf{u}^7 &= [\mathbf{y}^{7T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{14} &= [\mathbf{y}^{1T} \ \mathbf{y}^{7T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{21} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{7T}]^T \\
\mathbf{u}^8 &= [\mathbf{y}^{8T} \ \mathbf{y}^{1T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{15} &= [\mathbf{y}^{1T} \ \mathbf{y}^{8T} \ \mathbf{y}^{1T}]^T & \mathbf{u}^{22} &= [\mathbf{y}^{1T} \ \mathbf{y}^{1T} \ \mathbf{y}^{8T}]^T
\end{aligned}$$

Let U denote the collection of vectors $\mathbf{u}^1, \dots, \mathbf{u}^{22}$, i.e., $U = \{\mathbf{u}^1, \dots, \mathbf{u}^{22}\}$. One can easily verify that all the vectors in U are feasible for the model EX3-IP, and that these vectors are also affinely independent. Thus the convex hull of the feasible solutions for the model EX3-IP is full-dimensional.

We observe that with the exception of $\mathbf{u}^7$, every vector in U satisfies the inequality 4.7 at equality. Thus we have 21 feasible solutions for the model EX3-IP that are affinely independent and satisfy the inequality 4.7 at equality, hence the inequality 4.7 is a facet-defining VI for the model EX3-IP.

Similarly, for each of the inequalities 4.8 to 4.12, we can show that it is a facet-defining VI for the model EX3-IP by selecting 21 of the 22 vectors $\mathbf{u}^1$ through $\mathbf{u}^{22}$ specifically and showing that these affinely independent vectors satisfy this VI at equality (for inequalities 4.8, 4.9, 4.10, 4.11, and 4.12, consider the vectors in $U \setminus \{\mathbf{u}^8\}$, $U \setminus \{\mathbf{u}^{14}\}$, $U \setminus \{\mathbf{u}^{15}\}$, $U \setminus \{\mathbf{u}^{21}\}$, and $U \setminus \{\mathbf{u}^{22}\}$, respectively). $\square$

Using the structure and pattern described in Examples 2 and 3, it is easy to construct larger instances of SCC with similar structure and pattern for which CP-VIs are facet-defining VIs. Therefore, although in general a CP-VI is not always a facet-defining VI for the convex hull of

the feasible solutions for the model $SCC-IP$, it can be a facet-defining VI if the constraints set of the problem has a special structure.

4.3.3 Generalized Covering-Packing Valid Inequalities

In this subsection, we introduce a related family of VIs for the model $SCC-IP$ that we refer to as the *generalized* CP-VIs, or GCP-VIs for short. The following proposition explains the GCP-VIs.

Proposition 4. *The constraints*

$$\sum_{j:j \in N_i^{k_1} \cap N_i^{k_2}} x_j \geq x_{k_1} + x_{k_2} - 1, \quad \forall i \in M \text{ and } \forall k_1, k_2 \in N \setminus N_i \quad (4.13)$$

are valid inequalities for the model $SCC-IP$.

Proof. If neither node k_1 nor k_2 is selected to be in a feasible cover, this inequality reduces to

$$\sum_{j:j \in N_i^{k_1} \cap N_i^{k_2}} x_j \geq -1$$

which is trivially true at every feasible solution of the model $SCC-IP$. Similarly, if only one of the nodes k_1 or k_2 is selected to be in a feasible cover, this inequality reduces to

$$\sum_{j:j \in N_i^{k_1} \cap N_i^{k_2}} x_j \geq 0$$

which is still trivially true at every feasible solution of the model $SCC-IP$.

On the other hand, If both k_1 and k_2 are selected to be in a feasible cover, from the conflict constraints it follows that at every corresponding feasible solution for the model $SCC-IP$, we have $x_j = 0$ for every node in the covering node-set of element i that is in conflict with either k_1 or k_2 . It follows that the set covering constraint

$$\sum_{j:j \in N_i} x_j \geq 1$$

reduces to

$$\sum_{j:j \in N_i^{k_1} \cap N_i^{k_2}} x_j \geq 1$$

□

4.4 Covering-Clique Valid Inequalities

In this section we introduce another family of valid inequalities for the model *SCC-IP* that we refer to as Covering-Clique Valid Inequalities (CC-VIs), and we discuss its properties. The following proposition explains CC-VIs.

Proposition 5. *Given a clique q of size k , let i_j be any element of the cover-set of the j^{th} node of the clique, for $j = 1, \dots, k$. If no node in the clique q appears in every covering node-set for elements $i_1, i_2, \dots, i_k$, (i.e., in every one of the covering node-sets $N_{i_1}, N_{i_2}, \dots, N_{i_k}$) then the constraint*

$$\sum_{j \in (N_{i_1} \cup N_{i_2} \cup \dots \cup N_{i_k}) \setminus q} x_j \geq 1, \quad (4.14)$$

*is a valid inequality for the model *SCC-IP*.*

Proof. If none of the nodes in the clique q is selected to be in a feasible cover, this inequality follows, since at least one node in each of the sets $N_{i_1}, N_{i_2}, \dots, N_{i_k}$ must be selected to be in that feasible cover.

On the other hand, if any node $h \in q$ is selected to be in a feasible cover, it follows that there is at least one covering node-set N_{i_l} such that $h \notin N_{i_l}$. Since node h is in the clique q , no other node in q can be in that feasible cover. So the set covering constraint

$$\sum_{j: j \in N_{i_l}} x_j \geq 1$$

reduces to

$$\sum_{j \in N_{i_l} \setminus q} x_j \geq 1,$$

and therefore the inequality 4.14 follows. $\square$

Considering each conflict constraint as a clique of size 2, we derive the following corollary from Proposition 5.

Corollary 1. *For each conflicting pair of nodes $(r, s) \in F$, if we have two elements i_1 and i_2 such that each of the nodes r and s appears in only one of their corresponding covering node-sets N_{i_1} and N_{i_2} , then the constraint*

$$\sum_{j \in (N_{i_1} \cup N_{i_2}) \setminus \{r, s\}} x_j \geq 1, \quad (4.15)$$

*is a valid inequality for the model *SCC-IP*.*

4.4.1 Facet-defining Properties of CC-VIs

In general a CC-VI is not necessarily a facet-defining VI for the convex hull of the feasible solutions for the model *SCC-IP*. But this VI can be facet-defining if the constraints set of the problem satisfies a sufficient condition, i.e., it has a special structure.

To show that a CC-VI is not always a facet-defining VI for the convex hull of the feasible solutions for the model *SCC-IP*, we construct a small instance of the problem SCC, as follows.

Example 4. Consider an instance of the problem SCC shown by the following IP model with $m = 2$ covering inequalities and one conflict constraint.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^4 x_j && (EX4-IP) \\
& \text{subject to} && x_1 + x_2 \geq 1 \\
& && x_3 + x_4 \geq 1 \\
& && x_1 + x_3 \leq 1 \\
& && x_j \in \{0, 1\} \quad \forall j = 1, \dots, 4
\end{aligned}$$

The model *EX4-IP* has only 5 feasible solutions, as follows.

$$\begin{aligned}
\mathbf{d}^1 &= \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T & \mathbf{d}^2 &= \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}^T & \mathbf{d}^3 &= \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T \\
\mathbf{d}^4 &= \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}^T & \mathbf{d}^5 &= \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}^T
\end{aligned}$$

One can easily verify that the vectors $\mathbf{d}^1, \dots, \mathbf{d}^5$ are affinely independent. Thus the convex hull of the feasible solutions for the model *EX4-IP* is full-dimensional.

The model *EX4-IP* has only one CC-VI as follows.

$$x_2 + x_4 \geq 1 \tag{4.16}$$

In order for the inequality 4.16 to be a facet-defining VI for the model *EX4-IP*, there must exist 4 feasible and affinely independent vectors that satisfy the inequality 4.16 at equality. But we observe that only $\mathbf{d}^1$ and $\mathbf{d}^4$ satisfy the inequality 4.16 at equality. Therefore, the inequality 4.16 is not a facet-defining VI for the model *EX4-IP*. $\square$

Through Example 4, we showed that in general a CC-VI is not a facet-defining VI for the convex hull of the feasible solutions for the model *SCC-IP*. But as mentioned before, this VI can be facet-defining if the constraints set of the problem has a special structure. To show this, we construct another small instance of the problem SCC, as follows.

Example 5. Consider an instance of the problem SCC shown by the following IP model with $m = 4$ covering inequalities and one conflict constraint.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^6 x_j && (EX5-IP) \\
& \text{subject to} && x_1 + x_2 + x_3 \geq 1 && (\text{element 1}) \\
& && x_1 + x_2 + x_4 \geq 1 && (\text{element 2}) \\
& && x_3 + x_5 \geq 1 && (\text{element 3}) \\
& && x_4 + x_6 \geq 1 && (\text{element 4}) \\
& && x_3 + x_4 \leq 1 \\
& && x_j \in \{0, 1\} \quad \forall j = 1, \dots, 6
\end{aligned}$$

The only clique this instance has corresponds to the conflict between node 3 and node 4. The cover sets of nodes 3 and 4 are as follows.

- $M_3 = \{1, 3\}$
- $M_4 = \{2, 4\}$

Considering element $1 \in M_3$ and element $2 \in M_4$, and their corresponding covering-node sets $N_1 = \{1, 2, 3\}$ and $N_2 = \{1, 2, 4\}$, we find the following CC-VI for this numeric example.

$$x_1 + x_2 \geq 1 \tag{4.17}$$

Observation 3. *The inequality 4.17 is a facet-defining VI for the model EX5-IP.*

Proof. First, we show that the convex hull of the feasible solutions for the model $EX5-IP$ is full-dimensional. To this end, we need to find 7 feasible vectors that are affinely independent. Consider the following 7 vectors. First, we show that the convex hull of the feasible solutions for the model $EX5-IP$ is full-dimensional. To this end, consider the following 7 vectors.

$$\begin{aligned}
\mathbf{v}^1 &= \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}^T & \mathbf{v}^2 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}^T \\
\mathbf{v}^3 &= \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}^T & \mathbf{v}^4 &= \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}^T \\
\mathbf{v}^5 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T & \mathbf{v}^6 &= \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}^T \\
\mathbf{v}^7 &= \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}^T
\end{aligned}$$

One can easily verify that the vectors $\mathbf{v}^1, \dots, \mathbf{v}^7$ are feasible for the model *EX5-IP*, and that these vectors are also affinely independent. Since the solution space for this model is 6-dimensional, it follows that the convex hull of the feasible solutions for the model *EX5-IP* is full-dimensional.

To show that the inequality 4.17 is a facet-defining VI for the model *EX5-IP*, consider the vectors $\mathbf{v}^1, \dots, \mathbf{v}^6$. Clearly each of these vectors satisfies the inequality 4.17 at equality. Since these vectors are affinely independent and feasible for the model *EX5-IP*, it follows that the inequality 4.17 is a facet-defining VI for the model *EX5-IP*. $\square$

Using the structure of constraints in Example 5, we can construct larger instances for which CC-VIs are facet-defining VIs. To illustrate, we construct a larger instance of the SCC with 14 elements and 18 nodes, and we then show that CC-VIs are facet-defining VIs for its IP model, as follows.

Example 6. Consider an instance of the problem SCC shown by the following IP model with $m = 14$ covering inequalities and 9 conflict constraints.

$$\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{18} x_j \quad (EX6-IP) \\
\text{subject to} & x_1 + x_2 + x_3 \geq 1 \quad (\text{element 1}) \\
& x_1 + x_2 + x_4 \geq 1 \quad (\text{element 2}) \\
& x_3 + x_5 \geq 1 \quad (\text{element 3}) \\
& x_4 + x_6 \geq 1 \quad (\text{element 4}) \\
& x_3 + x_4 \leq 1 \\
& x_7 + x_8 + x_9 \geq 1 \quad (\text{element 5}) \\
& x_7 + x_8 + x_{10} \geq 1 \quad (\text{element 6}) \\
& x_9 + x_{11} \geq 1 \quad (\text{element 7}) \\
& x_{10} + x_{12} \geq 1 \quad (\text{element 8}) \\
& x_9 + x_{10} \leq 1 \\
& x_{13} + x_{14} + x_{15} \geq 1 \quad (\text{element 9}) \\
& x_{13} + x_{14} + x_{16} \geq 1 \quad (\text{element 10}) \\
& x_{15} + x_{17} \geq 1 \quad (\text{element 11}) \\
& x_{16} + x_{18} \geq 1 \quad (\text{element 12}) \\
& x_{15} + x_{16} \leq 1 \\
& x_j \in \{0, 1\} \quad \forall j = 1, \dots, 18 \\
& x_1 + x_{11} + x_{12} \geq 1 \quad (\text{element 13}) \\
& x_1 + x_7 \leq 1 \\
& x_3 + x_9 \leq 1 \\
& x_6 + x_{12} \leq 1 \\
& x_7 + x_{17} + x_{18} \geq 1 \quad (\text{element 14}) \\
& x_7 + x_{13} \leq 1 \\
& x_9 + x_{15} \leq 1 \\
& x_{12} + x_{18} \leq 1
\end{array}$$

Considering the conflict between nodes 3 and 4, and the covering-node sets N_1 and N_2 , we find the following CC-VI for this numeric example.

$$x_1 + x_2 \geq 1 \quad (4.18)$$

Similarly considering the conflict between nodes 9 and 10, and the covering-node sets N_5 and N_6 , we find the following CC-VI for this numeric example.

$$x_7 + x_8 \geq 1 \quad (4.19)$$

Finally considering the conflict between nodes 15 and 16, and the covering-node sets N_9 and N_{10} , we find the following CC-VI for this numeric example.

$$x_{13} + x_{14} \geq 1 \quad (4.20)$$

Observation 4. *The inequalities 4.18, 4.19, and 4.20 are facet-defining VIs for the model EX6-IP.*

Proof. First, we show that the convex hull of the feasible solutions for the model EX6-IP is full-dimensional. To this end, we need to find 19 feasible vectors that are affinely independent. Using the 7 vectors that we defined in Observation 3, we construct the following 19 vectors.

$$\begin{aligned} \mathbf{w}^1 &= [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \\ \mathbf{w}^2 &= [\mathbf{v}^2{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^8 = [\mathbf{v}^1{}^T \ \mathbf{v}^2{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{14} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^2{}^T]^T \\ \mathbf{w}^3 &= [\mathbf{v}^3{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^9 = [\mathbf{v}^1{}^T \ \mathbf{v}^3{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{15} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^3{}^T]^T \\ \mathbf{w}^4 &= [\mathbf{v}^4{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{10} = [\mathbf{v}^1{}^T \ \mathbf{v}^4{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{16} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^4{}^T]^T \\ \mathbf{w}^5 &= [\mathbf{v}^5{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{11} = [\mathbf{v}^1{}^T \ \mathbf{v}^5{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{17} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^5{}^T]^T \\ \mathbf{w}^6 &= [\mathbf{v}^6{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{12} = [\mathbf{v}^1{}^T \ \mathbf{v}^6{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{18} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^6{}^T]^T \\ \mathbf{w}^7 &= [\mathbf{v}^7{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{13} = [\mathbf{v}^1{}^T \ \mathbf{v}^7{}^T \ \mathbf{v}^1{}^T]^T \quad \mathbf{w}^{19} = [\mathbf{v}^1{}^T \ \mathbf{v}^1{}^T \ \mathbf{v}^7{}^T]^T \end{aligned}$$

Let W denote the collection of vectors $\mathbf{w}^1, \dots, \mathbf{w}^{19}$, i.e., $W = \{\mathbf{w}^1, \dots, \mathbf{w}^{19}\}$. One can easily verify that all the vectors in W are feasible for the model EX6-IP, and that these vectors are also affinely independent. Thus the convex hull of the feasible solutions for the model EX6-IP is full-dimensional.

To show that inequality 4.18 is a facet-defining VI for the IP model $EX6-IP$, consider the collection of 18 vectors in $W \setminus \{\mathbf{w}^7\}$. Clearly each of these vectors satisfies the inequality 4.18 at equality. Since these vectors are affinely independent and feasible for the model $EX6-IP$, it follows that the inequality 4.18 is a facet-defining VI for the model $EX6-IP$.

Similarly, to show that inequalities 4.19 and 4.20 are facet-defining VIs for the model $EX6-IP$, we consider the vectors in $W \setminus \{\mathbf{w}^{13}\}$ and the vectors in $W \setminus \{\mathbf{w}^{18}\}$, respectively. Thus for each of the inequalities 4.19 and 4.20, we have 18 affinely independent vectors that are feasible for the model $EX6-IP$ and satisfy the corresponding inequality at equality. Hence, inequalities 4.19 and 4.20 are also facet-defining VIs for the model $EX6-IP$. $\square$

Using the structure and pattern described in Examples 5 and 6, it is easy to construct larger instances of SCC with similar structure and pattern for which CC-VIs are facet-defining VIs. Therefore, although in general a CC-VI is not always a facet-defining VI for the convex hull of the feasible solutions for the model $SCC-IP$, it can be a facet-defining VI if the constraints set of the problem has a special structure.

4.5 Conflicting-Set Valid Inequalities

In this section, we introduce another family of valid inequalities for the model $SCC-IP$ which we refer to as Conflicting-Set Valid Inequalities (CS-VIs). The following proposition explains CS-VIs.

Proposition 6. *For every element $i \in M$, if there exists a set $\bar{N} \subseteq N - N_i$ such that $N_i \subseteq \bigcup_{j \in \bar{N}} G_j$, then the constraint*

$$\sum_{j \in \bar{N}} x_j \leq |\bar{N}| - 1 \quad (4.21)$$

is a valid inequality for the model $SCC-IP$.

Proof. If we select all nodes in $\bar{N}$ to be in a cover, then none of the nodes in $\bigcup_{j \in \bar{N}} G_j$ can be in that cover since each node in $\bar{N}$ is in conflict with at least one node in $\bigcup_{j \in \bar{N}} G_j$. Since $N_i \subseteq \bigcup_{j \in \bar{N}} G_j$, it follows that none of the nodes in N_i is in the corresponding cover, hence element i cannot be covered, i.e. the corresponding cover is not feasible. Thus in every feasible cover, at least one of the nodes in $\bar{N}$ must be not present. Hence, the inequality 4.21 follows. $\square$

We derive the following corollary from Proposition 6.

Corollary 2. For every element $i \in M$, if there exists two nodes $j, k \in N - N_i$ such that $N_i \subseteq G_j \cup G_k$, then the constraint

$$x_j + x_k \leq 1, \quad (4.22)$$

is a valid inequality for the model *SCC-IP*.

Through the following numeric example, we show that the proposed CS-VIs can eliminate part of the feasible region of the LP relaxation of the model *SCC-IP* (i.e., they are not redundant).

Example 7. Consider an instance of the problem SCC shown by the following IP model with $m = 6$ covering inequalities and four conflict constraints.

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^8 x_j && (EX7-IP) \\ & \text{subject to} && x_1 + x_2 + x_3 + x_4 \geq 1 \\ & && x_3 + x_5 + x_7 \geq 1 && x_3 + x_5 \leq 1 \\ & && x_4 + x_5 \geq 1 && x_4 + x_5 \leq 1 \\ & && x_6 + x_7 \geq 1 && x_1 + x_6 \leq 1 \\ & && x_6 + x_8 \geq 1 && x_2 + x_6 \leq 1 \\ & && x_3 + x_4 + x_8 \geq 1 \\ & && x_j \in \{0, 1\} \quad \forall j = 1, \dots, 8 \end{aligned}$$

We solve the linear programming relaxation of this model via simplex algorithm and obtain the following optimal solution:

$$\begin{aligned} x_6^* &= 1 \\ x_1^* &= x_2^* = x_7^* = x_8^* = 0 \\ x_3^* &= x_4^* = x_5^* = 0.5 \end{aligned}$$

The corresponding optimal value of the objective function is $z^* = 2.5$.

For $i = 2, 3, 4, 5, 6$, there does not exist any set $\overline{N} \subseteq N - N_i$ such that $N_i \subseteq \bigcup_{j \in \overline{N}} G_j$. For element 1, considering $\overline{N} = \{5, 6\}$, $G_5 = \{3, 4\}$ and $G_6 = \{1, 2\}$, we have:

$$N_1 = \{1, 2, 3, 4\} \subseteq \bigcup_{j \in \overline{N}} G_j = \{1, 2, 3, 4\} \quad (4.23)$$

Now $|\overline{N}| = 2$, and we find the following CS-VI for the IP model of this particular instance.

$$x_5 + x_6 \leq 1 \quad (4.24)$$

We observe that this CS-VI is violated at x^* . □

This example shows that the CS-VIs can remove a segment of the feasible region of the LP relaxation for the model *SCC-IP*. Hence they can be effective in the context of B&B for solving this problem.

4.6 Column-Domination Valid Inequalities

The Column-Domination valid inequalities (CD-VIs) are based on comparing the cover-sets associated with various columns of matrix A , as described below.

Proposition 7. *For every $j, k \in N$ such that $M_j \subseteq M_k$, the constraint*

$$x_j + x_k \leq 1, \tag{4.25}$$

*is a valid inequality for the set of optimal solutions of the model *SCC-IP*.*

Proof. For every two nodes j and k ($j, k \in N$), if $M_j \subseteq M_k$, i.e., all the elements covered by node j are also covered by node k , this inequality states that any feasible cover including both nodes j and k cannot be an optimal cover. The reason is that, since $c_j > 0 \quad \forall j \in N$, by removing node j from such feasible cover, we reduce the objective function value while maintaining the feasibility of the problem. □

Chapter 5

Computational Effectiveness of Valid Inequalities

In the previous chapter, we introduced several families of valid inequalities and showed that these inequalities are valid for the model *SCC-IP*. In this chapter, we evaluate the effectiveness of these VIs in solving the model *SCC-IP* through several computational experiments.

5.1 Covering-Packing Valid Inequalities

One strategy to use Covering-Packing valid inequalities (CP-VIs) in the context of solving the model *SCC-IP* would be to identify and add an appropriate subset of these constraints to the model at the root node of the B&B tree, and solve the resulting (larger) model via the B&B algorithm. The key idea is that by adding these constraints to the corresponding IP model, we can potentially obtain a better (higher) lower bound for its LP relaxation. This, in turn, allows the B&B algorithm to be more effective in solving the corresponding IP model (i.e., it fathoms various branches faster and thus it requires less number of nodes in the B&B tree).

The experiments performed by Kazemi Tutunchi and Fathi [71] indicate that the number of CP-VIs could be relatively large even for the instances of small and medium size. Of course, adding these additional constraints could also have a potentially negative impact on the computation requirements of solving the LP relaxation of the model *SCC-IP*, i.e., it could increase the corresponding execution time for solving the corresponding LP model.

In order to reduce the potential negative impact of the large number of additional constraints, Kazemi Tutunchi and Fathi [71] suggest using a common practice in this field, which is identifying and iteratively adding only those VIs that are violated at the optimal solution of the LP relaxation of the original model (i.e. add only the VIs that cut off the optimal solution of the LP relaxation of the model). To find such violated CP-VIs, we can use an iterative cutting

plane algorithm, following a well established practice in this field [53], as follows.

At each iteration of this algorithm, we identify and add a group of CP-VIs that are violated by the optimal solution of the LP relaxation of the model at the previous iteration. We repeat the process until we cannot identify any such violated CP-VI anymore (see Remark 3). We then add all or only a subset of the CP-VIs that we obtained in this manner to the model $SCC-IP$ and solve the resulting (larger) model via the branch and bound method.

We now describe the implementation of the cutting plane algorithm (CPA) for adding CP-VIs in more details.

Let us use the symbol H to denote the collection of all CP-VIs for a given model $SCC-IP$, and denote each constraint in the collection H by the symbol $\pi^t x \geq 0$, where $x = (x_1, \dots, x_n)^t$ is the vector of variables and $\pi = (\pi_1, \dots, \pi_n)^t$ is the vector of CP-VIs' coefficients. Following is the linear programming (relaxation) associated with the model $SCC-IP$ with the additional constraints; we refer to this model as $\overline{SCC}(H)$.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n c_j x_j && (\overline{SCC}(H)) \\
& \text{subject to} && \sum_{j=1}^n a_{ij} x_j \geq 1 && \forall i = 1, \dots, m \\
& && x_j + x_k \leq 1 && \forall j, k = 1, \dots, n : (j, k) \in F \\
& && \pi^t x \geq 0 && \forall \pi \in H \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, n
\end{aligned}$$

We now describe various steps of the iterative algorithm for solving $\overline{SCC}(H)$;

Initialization: set $\omega = 1$, Let S_R^1 be the feasible region of the LP relaxation of the model $SCC-IP$, i.e.,

$$\begin{aligned}
S_R^1 = \{ & \sum_{j=1}^n a_{ij} x_j \geq 1 \quad \forall i = 1, \dots, m, \\
& x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F, \\
& 0 \leq x_j \leq 1 \quad \forall j = 1, \dots, n \}
\end{aligned}$$

Iterative Steps: At iteration ω do;

1. Solve the following relaxation of $\overline{SCC}(H)$.

$$Z_R^\omega = \min\{z = \sum_{j=1}^n c_j x_j : x \in S_R^\omega\} \quad \overline{SCC}(H)^\omega$$

and let x^ω be the optimal solution for this problem.

2. Optimality test: If $\pi^t x^\omega \geq 0$ for all $\pi \in H$, then x^ω is an optimal solution for $\overline{SCC}(H)$; stop. Otherwise, go to next step.
3. Refinement: Identify a collection of (one or more) CP-VIs in the family H that are violated at x^ω , i.e., identify $H^\omega \subseteq H$ with the property that for each $\pi \in H^\omega$ we have $\pi^t x^\omega < 0$. Let $S_R^{\omega+1} = S_R^\omega \cap \{x \in \mathbb{R}^n : \pi^t x \geq 0 \quad \forall \pi \in H^\omega\}$.
4. $\omega \leftarrow \omega + 1$

Termination Criterion: Terminate the process either in step 2 or in step 3 when no violated CP-VI can be identified (see Remark 3).

Remark 1. In steps 2 and 3, given the solution x^ω to the relaxation $\overline{SCC}(H)^\omega$, we must either show that x^ω is a feasible solution for $\overline{SCC}(H)$ or find all or a subset of valid inequalities in the set H that are violated at x^ω . This problem is commonly known as the separation problem in the literature of IP. One way to solve this separation at each iteration is to simply substitute the vector x^ω in every CP-VI for the model SCC -IP and identify the violated CP-VIs, if any.

Remark 2. Our preliminary experiments indicated that the number of violated CP-VIs in step 3 is usually not very large for the instances of small and medium size; so for these instances we choose to add all violated CP-VIs in this step.

Remark 3. We choose to terminate our iterative cutting plane algorithm when there is no violated CP-VI anymore. Alternatively, we could terminate this iterative process after a certain number of iterations, or when the level of improvement in the lower bound is below a given threshold.

This algorithm clearly terminates after a finite number of iterations. This follows from the fact that at each iteration we add at least one CP-VI to the model and there is only a finite number of such constraints for any given model SCC -IP.

We can now combine this cutting plane algorithm with the branch and bound algorithm in the following two-phase manner;

Phase 1: Solve $\overline{SCC}(H)$ via the CPA as described above. At termination let x^ω be an optimal solution for $\overline{SCC}(H)$. If $x^\omega \in \mathbb{B}^n$, stop. x^ω solves the model SCC -IP. Otherwise go to phase 2.

Phase 2: Let $H' \subseteq H$ be the set of all cuts generated in phase 1, that is $H' = \bigcup_{s=1}^{\omega} H^s$. Add all CP-VIs associated with the set H' to the model $SCC-IP$ and solve the reformulation of the IP model (as stated below) by the B&B algorithm.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n c_j x_j && (SCC(H')) \\
& \text{subject to} && \sum_{j=1}^n a_{ij} x_j \geq 1 && \forall i = 1, \dots, m \\
& && x_j + x_k \leq 1 && \forall j, k = 1, \dots, n : (j, k) \in F \\
& && \pi^t x \geq 0 && \forall \pi \in H' \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, n
\end{aligned}$$

We refer to this procedure as the two-phase-CPVI method.

5.1.1 Computational Experiments

In this section, we present the results of two computational experiments in which we evaluate the effectiveness of adding CP-VIs to the model $SCC-IP$ via the two-phase-CPVI approach.

In our preliminary experiments, we noted that the effectiveness of CP-VIs varies significantly for different instances. In Subsection 5.1.1.1, we construct a collection of instances for which adding CP-VIs to the model $SCC-IP$ is particularly effective.

In Subsection 5.1.1.2, we use the data set that we constructed in Subsection 5.1.1.1 as well as the GEO-10D and RAN data sets as described in Section 3.3.1 of Chapter 3. Using these instances, we carry out a computational experiment in which we turn off all pre-processing, heuristics and cut generation options of CPLEX in order to observe the impact of CP-VIs without interference with the effects of other methods.

Subsequently, in Subsection 5.1.1.3, we use the above-mentioned data sets again, and we perform another computational experiment in which we turn on all cut generation options of CPLEX in order to observe the impact of CP-VIs in the presence of CPLEX cuts.

5.1.1.1 Data set A: CPVI-specific instances

This data set, which we refer to as data set A, is inspired by the instances with special structure and pattern that we constructed and described in Section 4.3.2 of Chapter 4. We now describe how we construct instances in this data set.

An instance of the problem SCC is identified by the set covering matrix A , the conflict collection F , and cost vector c . For an instance in this data set, we construct the set covering

matrix A randomly as follows. Each entry of the $m \times n$ matrix A is randomly chosen; it is equal to 1 with the probability of s and, equal to 0 with the probability of $1 - s$, where s is a given parameter in the interval $[0, 1]$. Note that this is the same method that we use to construct this matrix for the RAN instances, as described earlier. Then, for each column j of the matrix A , we find its corresponding set M_j , and for each row i of the matrix A , we find its corresponding set N_i .

In order to obtain the conflict collection F , let us define collection of columns R_j for each $j \in N$ as follows.

$$R_j = \bigcup_{i \in M_j} N_i$$

Subsequently, for each $j = 1, \dots, n - 1$ and each $k = j + 1, \dots, n$, we define the parameter Δ_{jk} as below.

$$\Delta_{jk} = |M_j| \cdot |M_k| / \max\{|R_j \cap R_k|, 1\}$$

We then use the value of Δ_{jk} for all pairs (j, k) to construct a collection of ρ conflict constraints, where ρ is a given integer parameter in the interval $[0, n(n - 1)/2]$. To this end, we use the method of roulette wheel selection as described in [48]. This method is so that the probability of the pair (j, k) being selected and added to the conflict collection F is proportional to the value of Δ_{jk} , i.e., the larger value of Δ_{jk} implies there is a higher chance that we select (j, k) and add it to the conflict collection F .

To construct the cost vector c , we fix every entry of the cost vector c to 1.

Within this collection, we construct instances with $m = 400$ and $n = 1300$ and different values for the parameters s and ρ . We carried other similar experiments with instances with other values of m and n , but the pattern of the observations was similar. Hence, for brevity, we only use the instances with $m = 400$ and $n = 1300$, in the next two experiments.

5.1.1.2 First Experiment

In this subsection, we carry out a computational experiment to assess the computational impact of adding CP-VIs to the model $SCC-IP$ via the two-phase-CPVI approach where we turn off all pre-processing, heuristics and cut generation options of CPLEX. In this experiment, we solve the model $SCC-IP$ for all instances from data set A as described in previous subsection as well as a selected group of instances from GEO-10D, RAN, and OR-L data sets as described in Chapter 3. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the IP model for each instance twice; once we solve each model via B&B method (i.e., without adding CP-VIs) and once with the two-phase-CPVI (i.e., with CP-VIs added to the model). In this experiment, we impose a time limit of 7200 seconds on all instances, and if the execution time of an instance exceeds this time limit, we abort the

program. In Tables 5.1 and 5.2, we report the number of nodes upon termination of the B&B method and the corresponding execution times along with the total number of CP-VIs added to the model when imposing the two-phase-CPVI method.

Table 5.1: Results of adding CP-VIs to the model *SCC-IP* for instances in data set A with $m = 400$ and $n = 1300$

ID	s	ρ	B&B Method		Two-Phase-CPVI Method			
			# of B&B Nodes	Total time 1 (sec)	# of CP-VIs	# of B&B Nodes	IP time 2 (sec)	Total time 2 (sec)
CP.1	0.0040	1250	22,749	86.4	102	59	2.2	11.6
CP.2		2200	33,312	242.8	111	556	9.0	17.6
CP.3		2500	73,019	660.5	116	700	13.2	23.0
CP.4		2800	44,628	470.2	117	857	16.7	31.6
CP.5		3100	108,475	1,226.8	127	3,109	56.1	71.1
CP.6		3750	223,167	3,236.2	128	16,535	247.5	263.0
CP.7	0.0043	1250	10,837	66.8	94	649	7.5	13.0
CP.8		2200	40,528	362.8	103	1,129	18.1	27.2
CP.9		2500	139,802	1,416.8	109	2,520	34.7	41.4
CP.10		2800	289,706	3,873.3	113	2,998	59.3	67.8
CP.11		3100	-	-	109	41,765	606.2	615.4
CP.12		3750	-	-	120	294,645	5,362.3	5,373.9
CP.13	0.0046	1250	10,440	75.2	83	4,025	35.0	42.4
CP.14		2200	43,678	494.0	85	4,617	55.6	62.1
CP.15		2500	61,180	649.0	82	11,197	151.9	158.6
CP.16		2800	-	-	78	46,340	644.5	651.5
CP.17		3100	-	-	81	166,986	2,460.8	2,468.6
CP.18		3750	-	-	88	91,623	2,023.2	2,031.3

We make the following observations from Table 5.1;

1. **Execution time comparison.** We observe that adding CP-VIs significantly impacts the execution time of the model *SCC-IP* by reducing the number of nodes in the branch and bound tree in these instances. More specifically, we can solve only 13 out of 18 instances directly with the IP solver without adding any VIs within the pre-specified time limit of 2 hours. By adding CP-VIs to the model *SCC-IP* in the context of the two-phase method we can solve all of these instances and we observe significant improvements in reducing the execution time and the number of nodes for these instances. More explicitly, for the

Table 5.2: Results of adding CP-VIs to the model *SCC-IP* for RAN, GEO-10D, and OR-L instances

					B&B Method		Two-Phase-CPVI Method			
ID	m	n	s or r_s	p or r_p	# of B&B Nodes	Total time 1 (sec)	# of CP-VIs	# of B&B Nodes	IP time 2 (sec)	Total time 2 (sec)
G10.4	100	1000	850	1050	10,601	2,230.0	977	4,335	2,178.7	2,184.8
G10.5			950	700	41	1.7	1	35	1.2	1.9
G10.6			950	850	15	5.6	39	23	7.4	8.6
G10.7			950	950	328	85.6	156	44	49.4	51.9
G10.8			950	1050	2,230	714.6	510	101	196.4	199.8
G10.10			1050	1050	19	19.7	283	0	11.3	14.6
G10.13	300	1000	950	700	3,012	31.6	3	3,116	31.8	37.8
G10.14			950	850	11,325	606.1	92	9,804	569.2	574.9
G10.15			950	950	-	-	430	-	-	-
R.3	100	300	0.05	0.1	71,651	362.4	40	35,736	211.7	212.1
R.7	100	1000	0.01	0.01	0	0.7	4	0	0.4	0.8
R.8			0.01	0.05	33,454	500.3	74	28,430	460.9	462.3
R.9			0.05	0.01	97,988	424.6	0	97,988	424.6	425.1
L.3	200	1000	0.05	0.03	6,375	81.0	18	4,734	67.0	69.5
L.4			0.05	0.04	20,641	376.3	28	17,297	329.8	333.3
L.5			0.05	0.05	36,621	1,035.7	48	36,474	1,077.1	1,081.6

13 instances that the IP solver without CP-VIs terminated within our time limit, the execution time of the two-phase method is 93.5% smaller than that of the B&B method, on average.

2. **Impact of value of ρ on number of CP-VIs.** We observe that for a given value of s as the value of ρ increases, the number of added CP-VIs in the two-phase method tends to increase. As we see in this table, the instance CP.6 has the largest number of added CP-VIs in the two-phase method. Among all 18 instances, this instance has the largest ρ value and smallest s value. The underlying reason behind this phenomenon is that as we increase value of ρ , i.e., the number of conflicts, the chance of CP-VIs becoming more effective increases, since for each element $i \in M$ and node $k \in N \setminus N_i$, as the value of ρ increases, the set N_i^k becomes smaller, and therefore the corresponding CP-VIs get tighter and the chance of these CP-VIs being violated becomes larger.
3. **Impact of value of s on number of CP-VIs.** We observe that for each value of ρ , as the value of parameter s decreases, i.e., as the set covering matrix A becomes more sparse, the number of the CP-VIs tends to increase.

We make the following observations from Table 5.2;

1. **Execution time comparison.** We observe that for 8 out of 16 instances, adding CP-VIs to the model *SCC-IP* could reduce the total execution time by 17.5% on average. We also observe a drop in the number of B&B nodes after adding CP-VIs to the model *SCC-IP* for these instances. Instance G10.15 could not be solved by either the B&B method or the two-phase-CPVI method. In the remaining instances, there is no significant difference between execution time of B&B method and the two-phase-CPVI method.
2. **Impact of covering percentage and number of conflicts on number of CP-VIs.** We observe that for given values of s and r_s as the values of p and r_p , i.e., number of conflict constraints, increases, the number of added CP-VIs in the two-phase method tends to increase. We also observe that for each value of p and r_p as the value of parameters s and r_s decreases, the number of the CP-VIs tends to increase.
3. **Number of CP-VIs in different data sets.** The number of CP-VIs added to the IP model for GEO-10D instances is significantly larger than RAN and OR-L instances, on average.

In summary based on our limited experiment, we observe that while adding the CP-VIs to the IP model *SCC-IP* for the instances in data set A is significantly effective, its effect is relatively small for the instances in data sets GEO-10D, RAN, and OR-L. We also observe a

larger number of CP-VIs for the instances with larger number of conflicts and more sparse set covering matrices.

5.1.1.3 Second Experiment

In this subsection, in order to observe the impact of CP-VIs in the presence of CPLEX cuts, we perform another computational experiment. In this experiment, we solve the model *SCC-IP* for a selected group of instances from GEO-10D, RAN, and OR-L data sets as described in Chapter 3 as well as all instances in data set A that we constructed in Subsection 5.1.1.1. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the model *SCC-IP* for those instances, considering the following four different scenarios. Note that the first and third scenarios are the ones that we used in the previous experiment. We include them here so as to have a better comparison of all the cases together.

- **Scenario 1:** We solve the model via B&B method (i.e., without adding CP-VIs) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 2:** We solve the model via B&B method (i.e., without adding CP-VIs) where we turn on all cut generation options of CPLEX.
- **Scenario 3:** We solve the model with the two-phase-CPVI method (i.e., with CP-VIs added to the model). We turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 4:** We solve the model with the two-phase-CPVI method (i.e., with CP-VIs added to the model). We turn on all cut generation options of CPLEX.

We solve each of the selected instances with each of the above mentioned scenarios and we show the corresponding execution times in Figures 5.1 and 5.2. In these Figures, we labeled scenarios 1,2,3, and 4 with "IP", "IP + CPLEX cuts", "IP + CPVI", and "IP + CPVI + CPLEX cuts", respectively.

We report all results of this experiment, including execution times along with the total number of CPLEX cuts added to the B&B tree in Scenarios 2 and 4 as well as total number of CP-VIs added to the model in Scenarios 3 and 4 in Tables B.1 and B.2, in Appendix B. We give a summary of the results in Figures 5.1 and 5.2, respectively. In Figure 5.1, we show the execution time for solving each instance in data sets GEO-10D, RAN, and OR-L under each different scenario. Similarly, in Figure 5.2, we show the execution time for solving each instance in data set A under each different scenario.

In Figure 5.1, we observe that for 3 out of 3 instances in OR-L data set, the model *SCC-IP* with CPLEX cuts (scenario 2) has a better performance than the model *SCC-IP* without any

Figure 5.1: Results of adding CP-VIs to the model $SCC-IP$ for the instances in data sets GEO-10D, RAN, and OR-L

Figure 5.2: Results of adding CP-VIs to the model $SCC-IP$ for the instances in data set A

cuts (scenario 1), but for 8 out of 15 instances in GEO-10D and RAN data sets, the model *SCC-IP* without any cuts (scenario 1) has a smaller execution time than the IP model with CPLEX cuts (scenario 2), which shows that CPLEX cuts might have a negative impact on the execution time of the IP model for the instances in these data sets. In this figure, we also observe that for 10 out of 15 instances in GEO-10D, RAN, and OR-L data sets, the IP model with CP-VIs (scenario 3) has a better performance than the IP model with CPLEX cuts (scenario 2) with respect to CPU time. It shows that for the instances in these data sets, although adding CP-VIs has not a major impact on the execution time of model *SCC-IP*, but still adding CP-VIs is a better choice than adding CPLEX cuts to the model *SCC-IP*.

In Figure 5.2, we observe that for 15 out of 18 instances in data set A, the IP model with CPLEX cuts (scenario 2) has a significantly smaller execution time than the IP model without cuts (scenario 1), which shows that CPLEX cuts can have a remarkable positive impact on the execution time of the B&B tree for these structured instances. In this figure we also observe that for 15 out of 18 instances in data set A, the IP model with CP-VIs and CPLEX cuts (scenario 4) has a considerably smaller CPU time than the IP model with CPLEX cuts (scenario 2). More specifically, for these 15 instances, adding CP-VIs to the IP model with CPLEX cuts could reduce the execution time by 64%, on average. For the remaining 3 instances, there is no significant difference between execution time of the IP model with CPLEX cuts (scenario 2) and the IP model with CP-VIs and CPLEX cuts (scenario 4).

In summary based on our limited experiment, we observe that for the instances in data set A, adding CPLEX cuts to the model has a significant effect in reducing the execution time of the model. We also observe that for each instance in this data set, adding the CP-VIs to the model in the presence of CPLEX cuts significantly reduces the execution time of the model. For the instances in data sets GEO-10D, RAN, and OR-L, adding CPLEX cuts to the model might increase the execution time of the model, and adding the CP-VIs in the presence of CPLEX cuts cannot significantly reduce the execution time of the model.

5.1.1.4 Concluding Remarks

The experiments that we performed in previous subsections showed that while adding CP-VIs to the model *SCC-IP* can be quite effective for some instances of the problem, its effect varies significantly among different instances. For the instances in data sets GEO-10D, RAN, and OR-L, the impact is relatively small, but the impact for the instances in data set A is quite significant. Our study in this dissertation is not aimed at identifying all situations where adding these CP-VIs can be beneficial. Instead, we focused on presenting their impact on a few selected instances only to demonstrate that this impact can be significant.

5.1.2 Generalized Covering-Packing Valid Inequalities

Based on the Proposition 4 in Chapter 4, Generalized Covering-Packing valid inequalities (GCP-VIs) can be added to the constraints set of the model $SCC-IP$ as valid inequalities to tighten the corresponding LP feasible region. To add these VIs to the model $SCC-IP$, we employed a cutting plane algorithm similar to what we introduced for adding CP-VIs to the model $SCC-IP$, and performed a preliminary experiment to assess the impact of adding GCP-VIs to the model $SCC-IP$. In our experiment, we solved the model $SCC-IP$ with and without adding GCP-VIs for a collection of small and medium size instances from GEO-10D, RAN, and OR-L libraries as described in Chapter 3.

We observed that in most of the instances of the problem in our pilot study, zero or only a few GCP-VIs are violated at the root node of the B&B tree, and adding these GCP-VIs to the model $SCC-IP$ does not have a measurable impact on the execution time of the model $SCC-IP$.

5.2 Covering-Clique Valid Inequalities

Our preliminary experiments indicated that the total number of Covering-Clique valid inequalities (CC-VIs) could be excessively large even for the instances of small and medium size. Thus adding all CC-VIs to the model $SCC-IP$ cannot positively affect the computational requirement, since the size of the corresponding LP relaxation will be excessively large. We note that even identifying all CC-VIs of a given instance of medium size takes a relatively long time and an excessively large amount of memory.

Consequently we consider only those CC-VIs that are associated with cliques of size 2. These CC-VIs are explained in Corollary 1 in Chapter 4. For adding these CC-VIs to the model $SCC-IP$, we use a cutting plane algorithm (CPA) similar to what we explained in Section 5.1. At each iteration of this CPA, we identify and add only those CC-VIs that are associated with cliques of size 2 and are violated by the optimal solution of the LP relaxation of the model at the previous iteration. We repeat the process until we cannot identify any such violated CC-VI anymore. We then add all the CC-VIs that we obtained in this manner to the model $SCC-IP$ and solve the resulting (larger) model via B&B method, similar to the two phase approach that we discussed in Section 5.1. We refer to this procedure as the two-phase-CCVI method.

5.2.1 Computational Experiments

In this section, we present the results of two computational experiments in which we evaluate the effectiveness of adding CC-VIs to the model $SCC-IP$ via the two-phase-CCVI approach.

In our preliminary experiments, we noted that the effectiveness of CC-VIs varies significantly

for different instances. In Subsection 5.2.1.1, we construct a collection of instances for which adding CC-VIs to the model *SCC-IP* is particularly effective.

In Subsection 5.2.1.2, we use the data set that we constructed in Subsection 5.2.1.1 as well as the GEO-10D, RAN, and OR-L data sets as described in Section 3.3.1 of Chapter 3. Using these instances, we carry out a computational experiment in which we turn off all pre-processing, heuristics and cut generation options of CPLEX in order to observe the impact of CC-VIs without interference with the effects of other methods.

Subsequently, in Subsection 5.2.1.3, we use the above-mentioned data sets again, and we perform another computational experiment in which we turn on all cut generation options of CPLEX in order to observe the impact of CC-VIs in the presence of CPLEX cuts.

5.2.1.1 Data set B: CCVI-specific instances

This data set, which we refer to as data set B, is inspired by the instances with special structure and pattern that we constructed and described in Section 4.4.1 of Chapter 4. We now describe how we construct instances in this data set.

As stated earlier, an instance of the SCC problem is identified by the set covering matrix A , the conflict collection F , and cost vector c . For an instance in this data set, we construct the set covering matrix A randomly as follows. Each entry of the $m \times n$ matrix A is randomly chosen; it is equal to 1 with the probability of s and, equal to 0 with the probability of $1 - s$, where s is a given parameter in the interval $[0, 1]$. Note that this is the same method that we use to construct this matrix for the RAN instances, as described earlier. Then, for each column j of the matrix A , we find its corresponding set M_j , and for each row i of the matrix A , we find its corresponding set N_i .

In order to obtain the conflict collection F , let us define the parameter Δ'_{jk} for each $j = 1, \dots, n - 1$ and each $k = j + 1, \dots, n$, as below.

$$\Delta'_{jk} = |M_j| \cdot |M_k| \cdot |R_j \cap R_k|$$

Subsequently, similar to the data set A, we use the value of Δ'_{jk} for all pairs (j, k) to construct a collection of ρ conflict constraints, where ρ is a given integer parameter in the interval $[0, n(n - 1)/2]$. To this end, we use the method of roulette wheel selection as described in [48]. This method is so that the probability of the pair (j, k) being selected and added to the conflict collection F is proportional to the value of Δ'_{jk} , i.e., the larger value of Δ'_{jk} implies there is a higher chance that we select (j, k) and add it to the conflict collection F .

To construct the cost vector c , we fix every entry of the cost vector c to 1.

Within this collection, we construct instances with $m = 400$ and $n = 1000$ and different values for the parameters s and ρ . We carried other similar experiments with instances with

other values of m and n , but the pattern of the observations was similar. Hence, for brevity, we only use the instances with $m = 400$ and $n = 1000$, in the next two experiments.

5.2.1.2 First Experiment

In this subsection, we carry out a computational experiment to assess the computational impact of adding CC-VIs to the model *SCC-IP* via the two-phase-CCVI approach where we turn off all pre-processing, heuristics and cut generation options of CPLEX. In this experiment, we solve the model *SCC-IP* for all instances from data set B as described in previous subsection as well as a selected group of instances from GEO-10D, RAN, and OR-L data sets as described in Chapter 3. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the IP model for each instance twice; once we solve each model via B&B method (i.e., without adding CC-VIs) and once with the two-phase-CCVI (i.e., with CC-VIs added to the model). In this experiment, we impose a time limit of 7200 seconds on all instances, and if the execution time of an instance exceeds this time limit, we abort the program. In Tables 5.3 and 5.4, we report the number of nodes upon termination of the B&B method and the corresponding execution times along with the total number of CC-VIs added to the model when imposing the two-phase-CCVI method.

In Table 5.3, we observe that adding CC-VIs significantly impacts the execution time of the model *SCC-IP* by reducing the number of nodes in the branch and bound tree in these instances. More specifically, we can solve only 10 out of 20 instances directly with the IP solver without adding any VIs within the pre-specified time limit of 2 hours. By adding the CC-VIs to the model *SCC-IP* in the context of the two-phase method, we can solve 17 of these instances and we observe significant improvements in reducing the execution time and the number of B&B nodes for these instances. More explicitly, for the 10 instances that the IP solver without CC-VIs terminated within our time limit, the execution time of the two-phase method is 83.7% smaller than that of the B&B method, on average. We do not observe any specific trend in number of CC-VIs when the value of s increases and the value of ρ is fixed or when the value of ρ increases and the value of s is fixed, in these instances.

We also make the following observations from Table 5.4;

1. **Execution time comparison.** We observe that for 7 out of 17 instances in GEO-10D, RAN, and OR-L data sets, adding CC-VIs to the model *SCC-IP* could reduce the total execution time by 19.8% on average. We also observe a drop in the number of B&B nodes after adding CC-VIs to the model *SCC-IP* for these instances. For the remaining instances, there is no significant difference between execution time of B&B method and the two-phase-CCVI method.
2. **Impact of number of conflicts on number of CC-VIs.** We observe that for given

Table 5.3: Results of adding CC-VIs to the model *SCC-IP* for instances in data set B with $m = 400$ and $n = 1000$

ID	s	ρ	B&B Method		Two-Phase-CCVI Method			
			# of B&B Nodes	Total time 1 (sec)	# of CC-VIs	# of B&B Nodes	IP time 2 (sec)	Total time 2 (sec)
CC.1	0.0044	1100	7,481,477	4,764.7	39	15,243	8.7	9.0
CC.2		1210	-	-	40	30,130	21.4	21.7
CC.3		1430	-	-	45	176,442	124.0	124.2
CC.4		1650	-	-	36	436,787	355.0	355.4
CC.5		1760	-	-	53	933,166	975.4	975.9
CC.6		1870	-	-	-	-	-	-
CC.7	0.0048	1100	10,284	12.5	38	311	1.2	1.5
CC.8		1650	148,519	229.5	50	3,729	9.2	9.5
CC.9		1870	315,441	585.0	60	9,473	23.6	24.1
CC.10		2090	1,009,604	3,108.6	72	110,776	402.3	403.2
CC.11		2310	1,111,119	3,270.9	93	362,185	1,471.0	1,472.3
CC.12		2530	-	-	89	1,476,895	6,736.2	6,737.3
CC.13		2640	-	-	-	-	-	-
CC.14	0.0052	1100	11,581	47.9	66	2,370	12.2	12.9
CC.15		1540	35,577	169.2	63	4,492	25.7	26.2
CC.16		1760	63,486	329.4	55	13,486	78.5	79.3
CC.17		1980	111,583	756.5	60	16,281	124.5	125.1
CC.18		2200	-	-	79	465,869	3,260.0	3,261.1
CC.19		2420	-	-	63	769,803	6,103.4	6,104.0
CC.20		2530	-	-	-	-	-	-

Table 5.4: Results of adding CC-VIs to the model *SCC-IP* for RAN, GE-10D, and OR-L instances

					B&B Method		Two-Phase-CCVI Method			
ID	m	n	s or r_s	p or r_p	# of B&B Nodes	Total time 1 (sec)	# of CC-VIs	# of B&B Nodes	IP time 2 (sec)	Total time 2 (sec)
G10.1	100	1000	850	700	0	1.3	17	0	1.4	1.8
G10.2			850	850	77	17.7	123	36	11.7	13.2
G10.3			850	950	1,670	277.4	199	1,364	201.9	205.8
G10.4			850	1050	10,601	2,230.0	228	6,014	1,897.9	1,902.8
G10.6			950	850	15	5.6	55	19	11.7	13.2
G10.9			1050	850	12	3.8	37	19	4.8	8.3
G10.14	300	1000	950	850	11,325	606.1	107	12,145	421.3	445.3
G10.16			1050	850	1,142	81.5	17	1,247	49.3	52.8
G10.17			1050	950	569	205.7	37	466	159.9	169.9
R.1	100	300	0.05	0.01	1,413	2.3	12	1,514	2.5	2.6
R.2			0.05	0.05	12,676	32.5	45	7,910	23.7	23.8
R.3			0.05	0.1	71,651	362.4	70	36,266	233.9	234.6
R.10	100	1000	0.05	0.05	99,723	3,449.8	3	107,587	3,431.6	3,432.1
L.1	200	1000	0.02	0.02	8,725	147.3	27	10,212	131.0	131.8
L.2			0.05	0.02	1,172	16.0	21	640	12.8	13.4
L.4			0.05	0.04	20,641	376.3	51	15,601	304.9	306.3
L.5			0.05	0.05	36,621	1,035.7	59	38,471	1,028.9	1,030.6

values of s and r_s , as the values of p and r_p , i.e., number of conflict constraints, increases, the number of added CC-VIs in the two-phase method tends to increase. The underlying reason behind this phenomenon is that as we increase the value of ρ , i.e., the number of conflicts, a larger number of CC-VIs can be generated, since we write CC-VIs for each conflicting pair of nodes $(r, s) \in F$, and each two elements i_1 and i_2 in M . Thus we have a larger number of CC-VIs and there is a higher chance of having larger number of violated CC-VIs.

3. **Impact of covering percentage on number of CC-VIs.** We also observe that for each value of p and r_p , as the value of parameters s and r_s decreases, i.e., as the set covering matrix A becomes more sparse, the number of the CC-VIs tends to increase.
4. **Number of CC-VIs in different data sets.** The number of CC-VIs added to the IP model for GEO-10D instances is significantly larger than RAN and OR-L instances, on average.

In summary based on our limited experiment, we observe that while adding the CC-VIs to the IP model *SCC-IP* for the instances in data set B is significantly effective, its effect is relatively small for the instances in data sets GEO-10D, RAN, and OR-L. We also observe a larger number of CC-VIs for the instances in data sets GEO-10D, RAN, and OR-L with larger number of conflicts and more sparse set covering matrices.

5.2.1.3 Second Experiment

In this subsection, in order to observe the impact of CC-VIs in the presence of CPLEX cuts, we perform another computational experiment. In this experiment, we solve the model *SCC-IP* for a selected group of instances from GEO-10D, RAN, and OR-L libraries as described in Chapter 3 as well as all instances in data set B that we constructed in Subsection 5.2.1.1. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the model *SCC-IP* for those instances, considering the following four different scenarios. Note that the first and third scenarios are the ones that we used in the previous experiment. We include them here so as to have a better comparison of all the cases together.

- **Scenario 1:** We solve the model via B&B method (i.e., without adding CC-VIs) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 2:** We solve the model via B&B method (i.e., without adding CC-VIs) where we turn on all cut generation options of CPLEX.

- **Scenario 3:** We solve the model with the two-phase-CCVI method (i.e., with CC-VIs added to the model). We turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 4:** We solve the model with the two-phase-CCVI method (i.e., with CC-VIs added to the model). We turn on all cut generation options of CPLEX.

We solve each of the selected instances with each of the above mentioned scenarios and we show the corresponding execution times in Figures 5.3 and 5.4. In these Figures, we labeled scenarios 1,2,3, and 4 with "IP", "IP + CPLEX cuts", "IP + CPVI", and "IP + CPVI + CPLEX cuts", respectively.

We report all results of this experiment including execution times along with total number of CPLEX cuts added to the B&B tree in Scenarios 2 and 4 as well as total number of CC-VIs added to the model in Scenarios 3 and 4 Tables C.1 and C.2 in Appendix C. We give a summary of the results in Figures 5.3 and 5.4, respectively. In Figure 5.3, we show the execution time for solving each instance in data sets GEO-10D, RAN, and OR-L under each different scenario. Similarly, in Figure 5.4, we show the execution time for solving each instance in data set A under each different scenario.

Figure 5.3: Results of adding CC-VIs to the model *SCC-IP* for the instances in data sets GEO-10D, RAN, and OR-L

Figure 5.4: Results of adding CC-VIs to the model *SCC-IP* for the instances in data set B

In Figure 5.3, we observe no significant difference between execution time of the IP model without cuts (scenario 1) and the IP model with CPLEX cuts (scenario 2). More specifically, from Table C.1 in Appendix C, we find that the average execution time for these selected instances in GEO-10D, RAN, and OR-L data sets is 520.7 seconds for the IP model without cuts (scenario 1) and 567.7 seconds for the IP model with CPLEX cuts (scenario 2). It shows that CPLEX cuts do not have any considerable impact on the IP model for the instances in these data sets (in fact, turning on these features of CPLEX results in a slight increase in the average execution time). In this figure, we also observe that the IP model with CC-VIs (scenario 3) has a better performance than the IP model with CPLEX cuts (scenario 2) with respect to CPU time for these selected instances in GEO-10D, RAN, and OR-L data sets. We also observe that the average execution time of the IP model with CC-VIs (scenario 3) for these instances is 475.8 seconds. It shows that for the instances in these data sets, although adding CC-VIs has not a major impact on the execution time of model *SCC-IP*, but still adding CC-VIs is a better choice than adding CPLEX cuts to the model *SCC-IP*.

In Figure 5.4, we observe that for all 20 instances in data set B, the IP model with CPLEX cuts (scenario 2) has a significantly smaller execution time than the IP model without cuts (scenario 1) which shows that CPLEX cuts have a remarkable positive impact on the execution time of the B&B tree for these structured instances. In this figure and Table C.2 in Appendix C, we also observe that for the one instance (CC.20) in data set B, the IP model with CC-VIs and CPLEX cuts (scenario 4) has a considerably smaller CPU time than the IP model with CPLEX cuts (scenario 2). For the remaining 19 instances, there is no significant difference between execution time of the IP model with CPLEX cuts (scenario 2) and the IP model with CC-VIs and CPLEX cuts (scenario 4).

In summary based on our limited experiment, we observe that for the instances in data set B, adding CPLEX cuts to the model has a significant effect in reducing the execution time of the model. We also observe that for one instance in this data set, adding the CC-VIs to the model in the presence of CPLEX cuts significantly reduces the execution time of the model. For the instances in data sets GEO-10D, RAN, and OR-L, adding CPLEX cuts to the model might increase the execution time of the model, and adding the CC-VIs in the presence of CPLEX cuts cannot significantly reduce the execution time of the model.

5.2.1.4 Concluding Remarks

The experiments that we performed in previous subsections showed that while adding CC-VIs to the model *SCC-IP* can be quite effective for some instances of the problem, its effect varies significantly among different instances. For the instances in data sets GEO-10D, RAN, and OR-L, the impact is relatively small, but the impact for the instances in data set B is quite significant. Our study in this dissertation is not aimed at identifying all situations where adding these CC-VIs can be beneficial. Instead, we focused on presenting their impact on a few selected instances only to demonstrate that this impact can be significant.

5.3 Maximal Clique Valid Inequalities

One way to add Maximal Clique valid inequalities (MC-VIs) to the model *SCC-IP* is to identify all maximal cliques of a given conflict graph G (all MC-VI constraints of a given model *SCC-IP*), and add them to the model *SCC-IP*. We performed a pilot study, and we employed the algorithm proposed by Tomita et al. [70] to find all maximal cliques of the conflict graph G . To the best of our knowledge, for finding all maximal cliques of a given graph, the algorithm of Tomita et al. [70] outperforms other existing algorithms in the open literature with respect to the execution time. For the details of their algorithm, we refer the reader to [70]. We observed that the model *SCC-IP* after adding all MC-VIs takes significantly longer time to solve compared to the model *SCC-IP* without adding MC-VIs, and the reason is that the number of all MC-VIs of a given model *SCC-IP* is relatively large even for the instances of small and medium size. Our pilot study also showed that even identifying all MC-VIs takes a relatively large amount of time for the small to medium size instances of the problem SCC that we consider in our experiments.

Consequently, we choose to introduce only an appropriate subset of maximal clique VIs. More specifically, we select a collection of maximal cliques so that each edge of the conflict graph G is in at least one of these maximal cliques. We then add the collection of MC-VIs associated with these maximal cliques to the model *SCC-IP*, and remove all original conflict

constraints from the model (since each conflict constraint will be dominated by at least one of these MC-VIs). We then solve the resulting model via B&B method. To identify a collection of maximal cliques with this property, we design an iterative algorithm which we refer to as the Maximal Clique Finder algorithm (MCF), and we describe this algorithm in the next subsection.

5.3.1 Maximal Clique Finder Algorithm

At each iteration of the Maximal Clique Finder algorithm (MCF), we introduce a global set q including vertices that constitute a complete subgraph found up to this iteration. The algorithm begins by letting q be an empty set, and expands q step by step by applying a recursive procedure to V and its succeeding induced subgraphs to search for larger and larger complete subgraphs until they reach maximal cliques. We terminate the algorithm when each edge of the graph is in at least one of the identified maximal cliques. We now describe this algorithm in more details.

For convenience, let the set of neighbors of node $v_j \in V$ be defined as $N(v_j) = \{v_k : (v_j, v_k) \in E, k \in N\}$, i.e., $N(v_j)$ includes all nodes that are connected to v_j in the conflict graph G . Let $E' = \emptyset$ where E' includes all edges that are in an identified maximal clique up to this time. Let the updated set of neighbors of node $v_j \in V$ be defined as $N'(v_j) = \{v_k : v_k \in N(v_j), (v_j, v_k) \notin E'\}$, and let the updated degree of each node $v_j \in V$ is defined as $d'(v_j) = |N'(v_j)|$, i.e., number of edges that are not in any of the identified maximal cliques so far and are connected to v_j . Let us use the symbol q^t to denote the t -th maximal clique found by the algorithm, and let us use Q to denote the collection of all maximal cliques that are found by the algorithm. The MCF algorithm is as follows. In steps 1.2 and 1.4 where the algorithm selects a node, if there is a tie, it chooses the node with smaller index.

Initialization Step: Set $Q = \emptyset$.

Iterative Step: At iteration t do;

1. Finding a maximal clique:
 - 1.1. Create a set q^t and set it as $q^t = \emptyset$.
 - 1.2. Find node v_j which has the maximum updated degree ($d'(v_j)$) in G . Add node v_j to the set q^t .
 - 1.3. Find the intersection of set of neighbors of each node in the current set q^t , i.e., $\cap_{v_k \in q^t} N(v_k)$. If $\cap_{v_k \in q^t} N(v_k) = \emptyset$, go to step 2. Otherwise, go to next step.
 - 1.4. Find the node $v_l \in \cap_{v_k \in q^t} N(v_k)$ which has the maximum updated degree (d') in G among all nodes in $\cap_{v_k \in q^t} N(v_k)$. Add node v_l to the current set q^t . Go to step 1.3.

2. Updating the algorithm's parameters:

2.1. Add q^t to the collection Q .

2.2. Update E' , i.e., let $E' \leftarrow E' \cup E(q^t)$ where $E(q^t)$ is the set of all edges of q^t . If $E' = E$, terminate this algorithm. Otherwise, go to next step.

2.3. Update d' for all nodes of the graph G . Let $t \leftarrow t + 1$.

2.4. Go to step 1.1.

This algorithm clearly terminates after a finite number of iterations, since at each iteration of the algorithm, we add at least one edge of the conflict graph G to the set E' , and there are only a finite number of edges in a given conflict graph G .

To clarify the above definitions and the algorithm, we give a simple numeric example in the context of a specific instance of the SCC in Appendix A.

5.3.2 Adding Maximal Clique Valid Inequalities to the Model

In this subsection, we combine the MCF algorithm with the branch and bound algorithm in the following two-phase manner;

Phase 1: Apply the MCF algorithm to the conflict graph G , and find a subset of maximal cliques of G , and call it Q .

Phase 2: Add the MC-VIs corresponding to maximal cliques in Q to the model $SCC-IP$, and remove all conflict constraints from the model $SCC-IP$. (see Remark 4). Then solve the reformulation of the IP model (as stated below) by the branch and bound algorithm.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n c_j x_j && (SCC(Q)) \\
& \text{subject to} && \sum_{j=1}^n a_{ij} x_j \geq 1 && \forall i = 1, \dots, m \\
& && \sum_{j: v_j \in q} x_j \leq 1 && \forall q \in Q \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, n
\end{aligned}$$

We refer to this procedure as the two-phase-MCVI method.

Remark 4. *In the MCF algorithm, we continue searching for maximal cliques until $E' = E$, i.e., each edge of the conflict graph G is in at least one of the identified maximal cliques. Each of the identified maximal cliques dominates every conflict constraint corresponding to its edges, thus in the presence of the identified maximal clique constraints in the model $SCC-IP$, all conflict constraints are redundant and we can remove them from the model $SCC-IP$.*

5.3.3 A Computational Experiment

In this section, we carry out a computational experiment to assess the computational impact of adding MC-VIs to the model *SCC-IP* via the two-phase-MCVI method that we introduced in previous section. In our experiment, we first impose pre-processing procedures (explained in Chapter 3) on all instances from GEO-10D, RAN, and OR-L data sets as described in Chapter 3, and then we solve the model *SCC-IP* for those instances, considering the following three different scenarios:

- **Scenario 1:** We solve the model via B&B method (i.e., without adding MC-VIs) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 2:** We solve the model via B&B method (i.e., without adding MC-VIs) where we turn on the clique cut generation option of CPLEX. All other cut generation options of CPLEX are turned off.
- **Scenario 3:** We solve the model with the two-phase-MCVI method (i.e., with MC-VIs added to the model) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.

In scenario 2, we turn on only the clique cut generation option of CPLEX in order to compare our proposed two-phase-MCVI method for adding clique constraints with the method of CPLEX for adding similar constraints to the model in the B&B tree.

We solve each instance with each of the above mentioned scenarios and we compare the number of nodes upon termination of the B&B method and the corresponding execution times. In Table 5.6, We report these observations along with total number of clique cuts added to the B&B tree in scenario 2 as well as total number of MC-VIs added to the model in scenario 3. In Table 5.5, We report the results of the MCF algorithm that we employ as the first phase of the two-phase-MCVI method in scenario 3. We impose a time limit of 7200 seconds on all instances in this experiment.

We make the following observations from Tables 5.5 and 5.6;

1. **Impact of number of conflicts on number of MC-VIs.** In Table 5.5, we observe that as we increase values of r_p and p while other parameters are fixed, the number of maximal cliques tends to increase. The underlying reason is that as we increase values of r_p and p , i.e., number of conflicts, the conflict graph G becomes more dense, and therefore more maximal cliques can be found by the MCF algorithm.
2. **Ratio of number of maximal cliques to conflicts.** In Table 5.5, we also observe that the ratio of number of maximal cliques to the number of conflicts is larger for RAN and

Table 5.5: Results of the MCF algorithm on GEO-10D, RAN, and OR-L instances

ID	m	n	s or r_s	p or r_p	# of Conflicts	# of Maximal Cliques
G10.1	100	1000	850	700	5,138	2,026
G10.2			850	850	21,898	3,519
G10.3			850	950	45,654	4,104
G10.4			850	1050	70,352	3,807
G10.5			950	700	5,338	2,067
G10.6			950	850	22,754	3,671
G10.7			950	950	49,006	4,266
G10.8			950	1050	92,279	4,430
G10.9			1050	850	22,780	3,667
G10.10			1050	1050	92,397	4,449
G10.11	300	1000	850	700	5,360	2,079
G10.12			850	850	21,636	3,550
G10.13			950	700	5,366	2,080
G10.14			950	850	22,780	3,667
G10.15			950	950	49,061	4,326
G10.16			1050	850	22,780	3,667
G10.17			1050	950	49,061	4,326
G10.18			1050	1050	92,397	4,449
R.1	100	300	0.05	0.01	453	447
R.2			0.05	0.05	2,261	1,567
R.3			0.05	0.1	4,526	2,047
R.4			0.1	0.05	2,312	1,601
R.5			0.1	0.1	4,615	2,081
R.6			0.1	0.2	9,042	2,427
R.7	100	1000	0.01	0.01	1,901	1,818
R.8			0.01	0.05	9,963	5,683
R.9			0.05	0.01	5,049	4,708
R.10			0.05	0.05	25,068	12,613
R.11			0.1	0.01	5,118	4,769
L.1	200	1000	0.02	0.02	10,175	8,105
L.2			0.05	0.02	10,175	8,105
L.3			0.05	0.03	15,242	10,122
L.4			0.05	0.04	20,294	11,552
L.5			0.05	0.05	25,393	12,743

Table 5.6: Results of adding MC-VIs to the model *SCC-IP*

	B&B Method		B&B Method with CPLEX Clique Cuts			Two-Phase-MCVI Method			
ID	# of B&B Nodes	Total time 1 (sec)	# of Cuts	# of B&B Nodes	Total time 2 (sec)	# of MC-VIs	# of B&B Nodes	IP time 3 (sec)	Total time 3 (sec)
G10.1	0	1.3	1	0	1.1	2,026	0	0.9	1.0
G10.2	77	17.7	10	510	43.1	3,519	179	6.5	7.1
G10.3	1,670	277.4	35	1,479	327.5	4,104	2,202	33.0	35.3
G10.4	10,601	2,230.0	71	9,584	2,693.9	3,807	7,243	133.4	137.7
G10.5	41	1.7	0	41	2.1	2,067	39	1.2	1.2
G10.6	15	5.6	3	6	7.3	3,671	7	1.9	2.5
G10.7	328	85.6	10	167	72.8	4,266	111	6.4	9.0
G10.8	2,230	714.6	22	2,306	756.6	4,430	2,982	40.4	47.4
G10.9	12	3.8	1	0	1.7	3,667	0	0.6	1.2
G10.10	19	19.7	9	32	66.5	4,449	0	1.7	9.9
G10.11	4,559	56.5	1	10,924	119.2	2,079	6,901	53.5	53.6
G10.12	-	-	-	-	-	3,550	328,984	3,804.8	3,805.3
G10.13	3,012	31.6	0	3,091	35.3	2,080	2,873	21.6	21.7
G10.14	11,325	606.1	4	10,238	544.9	3,667	4,664	58.3	58.9
G10.15	-	-	-	-	-	4,326	87,068	1,084.8	1,087.5
G10.16	1,142	81.5	0	1,204	80.5	3,667	1,271	19.0	19.6
G10.17	569	205.7	3	504	157.0	4,326	422	16.4	19.2
G10.18	2,704	1,162.5	11	2,813	1,207.0	4,449	8,289	116.2	123.5
R.1	1,413	2.3	0	1,323	2.6	447	1,342	2.1	2.1
R.2	12,676	32.5	1	8,553	26.0	1,567	6,630	18.2	18.2
R.3	71,651	362.4	1	36,701	223.0	2,047	38,510	151.0	151.0
R.4	7,655	29.8	0	7,353	27.8	1,601	6,698	19.9	20.0
R.5	11,466	65.1	0	3,624	28.2	2,081	3,783	16.6	16.7
R.6	15,298	175.4	0	76,571	683.4	2,427	79,446	347.7	347.9
R.7	0	0.7	0	0	0.7	1,818	0	0.3	0.3
R.8	33,454	500.3	10	38,115	562.8	5,683	24,026	273.0	273.2
R.9	97,988	424.6	0	2,131	27.3	4,708	2,165	25.6	25.8
R.10	99,723	3,449.8	0	98,089	3,221.7	12,613	97,708	1,592.6	1,593.4
R.11	831,164	5,513.6	0	886,094	5,011.9	4,769	915,260	5,309.9	5,310.0
L.1	8,725	147.3	3	12,338	194.5	8,105	14,650	133.4	133.7
L.2	1,172	16.0	1	512	12.0	8,105	545	11.6	11.9
L.3	6,375	81.0	2	5,010	67.1	10,122	4,540	65.0	65.5
L.4	20,641	376.3	3	19,542	362.0	11,552	15,339	308.6	309.2
L.5	36,621	1,035.7	5	41,149	1,156.6	12,743	42,139	1,053.5	1,054.3

OR-L instances compared to GEO-10D instances. More specifically, the ratio of number of maximal cliques to the number of conflicts is 0.67 for instances in RAN and OR-L data sets while this ratio is 0.17 for instances in GEO-10D data set. We can also observe this phenomenon in Table 5.6 where for 8 out of 16 instances in RAN and OR-L data sets, no CPLEX clique cut has been added to the B&B tree while for most of the instances in GEO-10D data set, at least one CPLEX clique cut has been added to the B&B tree.

3. **Scenario 1 versus Scenario 3.** In Table 5.6, we observe that adding MC-VIs significantly affects the execution time of the model *SCC-IP* in these instances. More specifically, we cannot solve 2 instances (G10.12 and G10.15) directly with the IP solver without adding any VIs within the pre-specified time limit of 2 hours, while after adding the MC-VIs to the model *SCC-IP* in the context of the two-phase method we can solve these two instances. In the remaining 32 instances that the IP solver without adding any VIs could solve in 2 hours, we observe 90%, 27%, and 5% reduction in execution time when we add MC-VIs to the model in GEO-10D, RAN, and OR-L instances, on average, respectively. It shows that the impact of MC-VIs is more significant in geometric instances than in random instances. It also shows that MC-VIs are more effective in solving RAN instances than they are for OR-L instances.
4. **Scenario 2 versus Scenario 3.** In Table 5.6, we also observe that the two-phase-MCVI method could solve all instances significantly faster than the B&B method with CPLEX clique cuts. More specifically, in 32 instances that both methods could solve in 2 hours, we observe 91%, 21%, and 12% reduction in execution time of GEO-10D, RAN, and OR-L instances, on average, respectively, when we use two-phase-MCVI method rather than B&B method with CPLEX clique cuts.
5. **Scenario 1 versus Scenario 2.** In Table 5.6, we observe that in 15 out of 34 instances, the execution time of B&B method with CPLEX clique cuts is larger than the B&B method without cuts. This observation shows that adding CPLEX clique cuts does not necessarily improve the solving process. The average execution time of B&B method without cuts on these instances is 553.6 seconds while the average execution time of B&B method with CPLEX clique cuts on these instances is 553.9 seconds. It shows that the B&B method with and without CPLEX cliques cuts have a similar performance on these instances, on average.

5.4 Column-Domination Valid Inequalities and Conflicting-Set Valid Inequalities

In this section, we discuss the impact of adding Column-Domination valid inequalities (CD-VIs) and Conflicting-Set valid inequalities (CS-VIs) to the model *SCC-IP*.

5.4.1 Impact of Adding Column-Domination Valid Inequalities to the Model

For adding CD-VIs to the model *SCC-IP*, we cannot employ a cutting plane algorithm similar to what we introduced in Sections 5.1 and 5.2. This is due to the fact that CD-VIs do not remove any of the optimal solutions of the LP relaxation for the model *SCC-IP*, hence no CD-VI can be violated at the optimal solution of the LP relaxation of the model *SCC-IP* at the root node of the B&B tree. Yet, these VIs can be effective in the context of B&B algorithm because they remove some non-optimal feasible solutions for the model *SCC-IP*.

One way to add CD-VIs to the model *SCC-IP* is to identify and add all these constraints to the model *SCC-IP*. In our pilot studies, we observed that the execution time of solving the resulting model after adding all CD-VIs is not significantly different from the execution time of solving the model *SCC-IP* without CD-VIs. Therefore, we conclude that adding CD-VIs to the model *SCC-IP* in absence of other VIs is not positively impactful on the computational effort required to solve the model *SCC-IP*.

Yet, we observed that adding these VIs before adding MC-VIs to the model can be helpful in the effectiveness of MC-VIs, since CD-VIs can be considered as new conflicts in the conflict graph G and therefore can result in maximal cliques of larger size, which can positively affect the execution time required for solving the model *SCC-IP*. We evaluate the impact of this approach in Section 5.4.3.

5.4.2 Impact of Adding Conflicting-Set Valid Inequalities to the Model

Our preliminary experiments showed that the number of all CS-VIs, even for an instance of relatively small size, can be excessively large, and identifying all CS-VIs can take a large amount of time and memory. Thus identifying and adding all these constraints to the model *SCC-IP* does not positively affect the computational effort required to solve the model *SCC-IP*.

As a result, to use CS-VIs for solving model *SCC-IP*, we then considered identifying and adding only an appropriate subset of these constraints to the model *SCC-IP*. To this end, we considered identifying all CS-VIs with RHS (right-hand side) equal to 1, and adding them to the model *SCC-IP*. These CS-VIs are explained in Corollary 2 in Chapter 4. The result of our pilot study showed that, on average, the model after adding these CS-VIs takes less time

to solve, compared to the model without adding CS-VIs, but the overall difference between execution times is not significant.

Similar to CD-VIs, if we add CS-VIs (with RHS equal to 1) to the model *SCC-IP* before adding MC-VIs, these VIs can be considered as new conflicts in the conflict graph G and therefore can positively affect the execution time required for solving the model *SCC-IP*. In the next section, we discuss the impact of this approach.

5.4.3 Adding CD-VIs and CS-VIs before MC-VIs

As discussed in previous two subsections, an approach for adding CD-VIs and CS-VIs to the model *SCC-IP* is to add these VIs before adding MC-VIs. To evaluate the impact of this approach on the computational effort required to solve the model *SCC-IP*, we here perform a computational experiment in which we consider the following two scenarios;

- **Alt0:** Solve the model *SCC-IP* with the two-phase-MCVI method, explained in Section 5.3.
- **Alt1:** Add all CS-VIs with RHS equal to 1 as well as all CD-VIs to the model *SCC-IP*; Then, apply the two-phase-MCVI method on the resulting model.

In the Alt1 scenario, after adding CD-VIs and CS-VIs, we consider all conflicts corresponding to the identified CD-VIs and CS-VIs as new edges of the conflict graph G , and then apply the MCF algorithm on updated graph G , and find and add MC-VIs to the model. Then, as explained in Section 5.3, we remove all original conflict constraints as well as CD-VIs and CS-VIs from the model *SCC-IP*, since they are redundant in the presence of MC-VIs.

In this experiment, we solve the model *SCC-IP* for a collection of small and medium size instances from GEO-10D, RAN, and OR-L libraries as described in Chapter 3. Each model is defined by the instance ID from the corresponding library, and also the instance size m , n , and the values of the parameters s or r_s and p or r_p .

In this experiment, we turn off all pre-processing, heuristics and cut generation options of CPLEX in order to observe the impact of CD-VIs and CS-VIs. We first impose pre-processing procedures (explained in Chapter 3) on each instance, and then for IP model of each instance, we apply each of the above-mentioned scenarios. In Table 5.7, we report the total number of VIs added, the total execution time of each scenario, and the number of nodes subsequently used by B&B method.

As expected, we observe that for 6 out of 15 instances, adding CD-VIs and CS-VIs before adding MC-VIs to the model *SCC-IP*, as explained in the Alt1 scenario, can reduce the computational effort required to solve the model *SCC-IP* compared to the Alt0 scenario where we do not add CD-VIs and CS-VIs to model *SCC-IP*. For the remaining instances, the total

Table 5.7: Results of adding CD-VIs and CS-VIs to the model *SCC-IP*

ID	m	n	s or r_s	p or r_p	Policy	No. of CD-VI	No. of CS-VI	No. of MC-VI	No. of B&B Nodes	Total Time (sec)
G10.2	100	1000	850	850	Alt0	-	-	3,519	179	7.1
					Alt1	4,821	552	3,784	70	6.3
G10.3	100	1000	850	950	Alt0	-	-	4,104	2,202	35.3
					Alt1	3,544	4,013	4,442	1,440	32.8
G10.4	100	1000	850	1050	Alt0	-	-	3,807	7,243	137.7
					Alt1	1,993	21,007	4,843	3,684	121.5
G10.6	100	1000	950	850	Alt0	-	-	3,671	7	2.5
					Alt1	3,039	2	3,920	5	2.6
G10.15	300	1000	950	950	Alt0	-	-	4,326	87,068	1,087.5
					Alt1	138	100	4,339	83,482	1,064.5
G10.18	300	1000	1050	1050	Alt0	-	-	4,449	8,289	123.5
					Alt1	23	31	4,437	3,079	67.1
R.4	100	300	0.1	0.05	Alt0	-	-	1,601	6,698	20.0
					Alt1	1	0	1,601	6,612	20.7
R.5	100	300	0.1	0.1	Alt0	-	-	2,081	3,783	16.7
					Alt1	1	21	2,076	3,536	15.1
R.6	100	300	0.1	0.2	Alt0	-	-	2,427	79,446	347.9
					Alt1	1	43	2,401	74,268	314.8
R.7	100	1000	0.01	0.01	Alt0	-	-	1,818	0	0.3
					Alt1	2,864	6	2,237	58	1.7
R.8	100	1000	0.01	0.05	Alt0	-	-	5,683	24,026	273.2
					Alt1	2,817	217	5,791	21,459	264.1
R.11	100	1000	0.1	0.01	Alt0	-	-	4,769	915,260	5,310.0
					Alt1	22	0	4,785	817,456	5,134.1
L.1	200	1000	0.02	0.02	Alt0	-	-	8,105	14,650	133.7
					Alt1	908	0	8,432	9,607	103.2
L.2	200	1000	0.05	0.02	Alt0	-	-	8,105	545	11.9
					Alt1	7	0	8,109	965	15.8
L.3	200	1000	0.05	0.03	Alt0	-	-	10,122	4,540	65.5
					Alt1	7	0	10,127	3,765	58.8

execution time of Alt1 is similar to Alt0 which shows that even if adding CD-VIs and CS-VIs before adding MC-VIs to the model *SCC-IP* has not a positive impact on the execution time of the model *SCC-IP* in some instance of the problem, it also has not a negative impact on it.

We also observe that as we increase values of p and r_p while values of the parameters are fixed, the number of added CD-VIs decreases and the number of added CS-VIs increases. Note that the seventh and eighth columns of Table 5.7 represent the number of CD-VIs and CS-VIs that are added to the problem, respectively, not total number of these VIs.

By the definition of CD-VIs, we know that total number of these VIs for an instance of the problem with given values of m , n , s or r_s is fixed regardless of number of conflicts. As we increase number of conflicts, a larger number of CD-VIs become redundant, and it is the reason that less number of these VIs are added to the problem.

By the definition of CS-VIs, it is clear that after adding conflicts to the problem, total number of CS-VIs will increase or remain the same. As we increase number of conflicts, there is a higher chance that a larger number of CS-VIs become redundant, but we observe that the number of CS-VIs that are added to the problem still increases due to a large number of CS-VIs that are generated after adding conflicts.

5.5 Odd-Cycle Valid Inequalities

Based on the Proposition 2 in Chapter 4, Odd-Cycle valid inequalities (OCIs) can be added to the constraints set of the model *SCC-IP* as valid inequalities to tighten the LP feasible region. To add these VIs to the model *SCC-IP*, we need to define a separation procedure to employ at each node of the B&B tree; subsequently we need to find those OCIs that are violated by the current LP solution, and add a subset of these violated OCI to the model *SCC-IP*.

Rebennack et al. [62] explained an exact separation procedure for OCIs in detail, and we will not repeat it here. They showed that the procedure can either find some OCIs, which are violated by an LP solution x , or prove that x satisfies all OCIs of the conflict graph G . This can be done by computing a “minimum-weight” odd-cycle in a graph, with an appropriate weighting function. If the solution x satisfies the corresponding OCI for that minimum-weight odd-cycle, then all OCIs are satisfied. Otherwise, some (maximal) violated OCIs have been obtained. For the details of the procedure, we refer the reader to [62].

We employed the separation procedure of [62], and performed a preliminary experiment to assess the impact of adding OCIs to the model *SCC-IP*. In our experiment, we solved the model *SCC-IP* with and without adding OCIs for a collection of small and medium size instances from RAN and GEO-10D libraries as described in Chapter 3. Our experiment indicated that adding OCIs to the model *SCC-IP* does not reduce the computational effort required to solve the model *SCC-IP* in most instances of our experiment. We observed that even in most of

the instances of the problem in our pilot study, the execution time of the model $SCC-IP$ after adding OCIs significantly increases. The underlying reason behind such increase in execution time is that in most of the instances in our experiment, zero or only a few OCIs are added to the B&B tree, while the CPU time required for the separation procedure at each node of the tree is relatively large. If the tree has a relatively large number of nodes, total time of the separation procedures becomes excessively large. As a result, we conclude that OCIs are not effective in solving the IP model for the instances in GEO-10D and RAN libraries.

5.6 Including All Valid Inequalities Within a B&B Framework

To include the VIs that we introduced in Chapter 4 within a branch and bound framework, we can use different orderings of adding them to the model $SCC-IP$. In order to observe the computational impact of various strategies in this context, we performed a pilot study in which we experimented on several strategies with different orderings of adding VIs to the model $SCC-IP$, and we selected the strategy with the lowest average execution time. We refer to this strategy as the *Six-Phase Method*.

In this section, we explain our proposed six-phase method in details, and perform a computational experiment to assess the effectiveness of this method for solving the model $SCC-IP$. Following is a description of each phase of the method.

1. Impose the pre-processing procedures (explained in Chapter 3) on the model $SCC-IP$.
2. Add all CD-VIs to the pre-processed model.
3. Add all CS-VIs (with RHS equal to one) to the resulting model.
4. Add a subset of MC-VIs to the resulting model through the two-phase-MCVI method (see Remark 5).
5. Let H be the collection of CC-VIs that are identified for cliques of size 2 as well as all CP-VIs; Apply an iterative cutting plane algorithm similar to the CPA explained in Section 5.1 to the resulting model and add all violated VIs that are in H to the resulting model.
6. Solve the resulting model via B&B algorithm.

Remark 5. In phase 4, we consider all conflicts corresponding to the identified CD-VIs and CS-VIs as new edges of the conflict graph G , and we then apply the MCF algorithm on updated graph G , and identify and add MC-VIs to the model. Then, as explained in Section 5.3, we remove all original conflict constraints as well as CD-VIs and CS-VIs from the model $SCC-IP$.

5.6.1 A Computational Experiment

In this subsection, we carry out a computational experiment to assess the computational impact of adding the VIs that we introduced in Chapter 4 to the model *SCC-IP* via the six-phase method. In this experiment, we turn on all pre-processing, heuristics and cut generation options of CPLEX, and we solve the model *SCC-IP* for all instances in RAN and GEO-10D data sets as described in Chapter 3. To this end, we solve the model for each instance twice. Once we solve the model for each instance via B&B method (i.e., without adding VIs) and once with the six-phase method (i.e., with VIs added to the model). In this experiment, we impose a time limit of 7200 seconds on all instances, and if the execution time of an instance exceeds this time limit, we abort the program. In Table 5.8, we report the corresponding execution times along with the total number of each family of VIs added to the model when imposing the six-phase method.

In Table 5.8, we observe that in data set GEO-10D, we cannot solve 2 instances (G10.12 and G10.15) directly with the IP solver without adding our proposed VIs within the pre-specified time limit of 2 hours, while after adding our proposed VIs to the model *SCC-IP* in the context of the six-phase method we can solve all instances. In the remaining 16 instances that the IP solver without adding any VIs could solve in 2 hours, we observe 92% reduction in execution time when we add all our VIs to the model in GEO-10D instances, on average.

For RAN instances, the B&B method and the six-phase method could solve all 11 instances in the pre-specified time limit of 2 hours. For these 11 instances in data set RAN, we observe a 25% reduction in execution time when we add all our proposed VIs to the model, on average. For OR-L instances, we observe that adding all VIs in the context of the six-phase method does not seem to be quite effective in reducing the computational effort required to solve the corresponding model *SCC-IP*. For these instances, we observe a 5% reduction in execution time when we apply the six-phase method to the model *SCC-IP*, on average.

Consequently, we observe that the impact of our proposed VIs is more significant in geometric instances than in random instances, similar to what we observed in previous experiments when we assessed the computational impact of each VI in the absence of other VIs.

Comparing Table 5.8 with Table B.1 in Appendix B, we observe that the number of added CP-VIs to the model in the six-phase method is smaller than the number of added CP-VIs to the model in the two-phase-CPVI method. We can observe a similar phenomenon in the number of added CC-VIs to the model in the six-phase method and the two-phase-CCVI method by comparing Table 5.8 with Table C.1 in Appendix C. The underlying reason behind this phenomenon is that in the six-phase method, we identify and add CP-VIs and CC-VIs that are violated at the optimal solution of the LP relaxation of the model with MC-VIs, while in the two-phase-CPVI method and the two-phase-CCVI method, we identify and add these VIs that

Table 5.8: Results of adding all VIs in the context of six-phase method to the model *SCC-IP*

	B&B Method	Six-Phase Method						
ID	Total time 1 (sec)	# of CD-VI	# of CS-VI	# of MC-VI	# of CC-VI	# of CP-VI	IP time 2 (sec)	Total time 2 (sec)
G10.1	0.7	5,695	50	2,593	18	4	0.7	4.1
G10.2	40.3	4,821	552	3,784	73	65	4.9	10.6
G10.3	448.5	3,544	4,013	4,442	71	177	24.9	34.4
G10.4	3,087.3	1,993	21,007	4,843	48	351	82.4	95.5
G10.5	2.3	3,084	0	2,757	19	1	1.3	4.0
G10.6	5.5	3,039	2	3,920	30	23	2.5	5.5
G10.7	72.3	2,442	12	4,418	62	123	2.7	6.7
G10.8	1,369.8	1,853	889	4,560	36	231	8.7	16.0
G10.9	2.0	1,136	0	3,837	36	10	1.1	6.5
G10.10	65.0	549	4	4,495	48	272	3.3	13.1
G10.11	174.0	579	29	2,220	48	13	44.0	52.4
G10.12	-	400	1,863	3,857	63	90	3,134.2	3,146.2
G10.13	36.6	236	0	2,152	17	1	22.2	30.7
G10.14	790.5	178	0	3,681	48	47	55.8	67.2
G10.15	-	138	100	4,339	59	240	788.1	810.2
G10.16	80.2	63	0	3,686	12	12	19.8	47.7
G10.17	149.5	36	0	4,350	27	89	16.4	71.3
G10.18	1,538.9	23	31	4,437	39	437	96.6	169.1
R.1	3.1	106	0	518	13	2	2.7	3.1
R.2	22.7	123	0	1,601	42	9	14.5	15.4
R.3	228.5	114	0	2,056	63	31	233.5	234.5
R.4	38.4	1	0	1,601	0	0	20.8	21.1
R.5	31.5	1	21	2,076	0	0	15.3	15.8
R.6	233.4	1	43	2,401	0	2	166.9	168.1
R.7	0.9	2,864	6	2,237	3	5	1.5	2.9
R.8	702.4	2,817	217	5,791	30	65	310.4	316.2
R.9	26.8	2,385	0	5,642	0	0	27.3	29.3
R.10	3,219.3	2,297	0	12,939	3	0	1,277.7	1,283.4
R.11	5,120.5	22	0	4,785	0	0	5,087.4	5,089.4
L.1	207.8	908	0	8,432	33	23	158.3	168.8
L.2	13.2	7	0	8,109	33	8	12.5	16.0
L.3	69.1	7	0	10,127	39	19	53.4	59.1
L.4	357.4	7	0	11,550	57	25	303.7	315.9
L.5	994.3	7	0	12,765	67	37	1,012.3	1,023.2

are violated at the optimal solution of the LP relaxation of the original model (without MC-VIs). Clearly, the feasible region of the LP relaxation of the model after adding MC-VIs becomes tighter, and the number of CP-VIs and CC-VIs that are violated by the optimal solution of this LP relaxation significantly decreases. But it is interesting to observe that even in the presence of MC-VIs, the two families of VIs, namely CP-VIs and CC-VIs, are still impactful.

Chapter 6

Bi-criteria Set Covering Problem With Conflict Constraints

In the set covering problem with conflict constraints, the goal is to find the minimum cost conflict-free subset of nodes to cover all elements. However, in some applications of this problem, such as the wireless communication location problem, the total cost of such conflict-free subset of nodes can be relatively large. In these applications, we may be interested in simultaneously maximizing the number of covered elements and minimizing the number of selected nodes. Such applications lend themselves to a bi-criteria analysis. In this chapter, we carry out an analysis of this bi-criteria optimization problem. In particular, we focus on the case where all $c_i = 1$, and we refer to it as the Bi-criteria *unicost* Set Covering problem with Conflict constraints (Bi-SCC).

We first define the Bi-criteria *unicost* Set Covering problem with Conflict constraints (Bi-SCC), and propose an integer programming model for this problem. Then we present a brief review of appropriate terminologies and solution methods of multi-objective optimization, and we introduce two strategies for analyzing the problem. We also discuss the computational requirements of the proposed strategies through a limited computational experiment.

6.1 Statement of the Problem

In the Bi-criteria *unicost* Set Covering problem with Conflict constraints (Bi-SCC), we have an m by n matrix $A = (a_{ij})$, where each entry a_{ij} is either 0 or 1, and the i -th row of this matrix (element i) is said to be covered by column j (node j) if $a_{ij} = 1$. We also have a collection F of pairs of columns (j, k) , and we say that columns j and k represent a conflict if $(j, k) \in F$. We seek to identify a conflict-free collection of columns of A (nodes) to simultaneously maximize the number of rows of A (elements) that are covered and minimize the number of columns

(nodes) selected.

We define two sets of binary variables, x_j for $j = 1, \dots, n$ and y_i for $i = 1, \dots, m$ as follows.

$$x_j = \begin{cases} 1, & \text{if column } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if row } i \text{ is covered} \\ 0, & \text{otherwise} \end{cases}$$

Following is a mathematical representation of the Bi-SCC.

$$\text{maximize} \quad \sum_{i=1}^m y_i \quad (Bi-IP)$$

$$\text{minimize} \quad \sum_{j=1}^n x_j$$

$$\text{subject to} \quad \sum_{j=1}^n a_{ij}x_j \geq y_i \quad \forall i = 1, \dots, m \quad (6.1a)$$

$$x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F \quad (6.1b)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (6.1c)$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \dots, m \quad (6.1d)$$

Constraints in group 6.1a state that if $y_i = 1$ for a row i , then this row must be covered by at least one column of A . Constraints in group 6.1b represent a collection of conflicts between various pairs of columns in A .

6.1.1 Terminology

In the model *Bi-IP*, we have two conflicting objective functions; since a decrease in the number of selected columns (nodes) results in a decrease in the number of covered rows (elements). Hence simultaneous achievement of both objectives might not be possible. In this context, let Υ denotes each feasible solution of this problem, and we define the following terms.

Definition 1. (*Domination*) For a given model *Bi-IP*, a feasible solution Υ^1 *dominates* a feasible solution Υ^2 if Υ^1 is at least as good as Υ^2 with respect to one objective function and it is strictly better than Υ^2 with respect to the other objective.

Definition 2. (*Non-dominated Solution and Efficient Point*) For a given model *Bi-IP*, a feasible solution Υ^* is said to be a *non-dominated solution*, if there is no other feasible solution

Υ^1 that dominate Υ^* . The corresponding point in the xy plane, where x and y are the values of the objective functions, is said to be an efficient point.

Definition 3. (*Weakly non-dominated solution and Weakly Efficient Point*) For a given model *Bi-IP*, a feasible solution Υ^* is said to be a *weakly efficient solution*, if there is no other feasible solution Υ^1 that has strictly better value than Υ^* for both objective functions. The corresponding point in the $x-y$ plane is said to be a weakly efficient point.

Definition 4. (*Efficient Frontier*) The set of all efficient points is known as the *efficient frontier*.

6.1.2 Efficient Frontier: An Illustrative Example

We provide an example to illustrate the above definitions and to demonstrate the trade-off between the two objective functions. To this end, we construct a small instance of the problem with $m = 50$ and $n = 100$. To construct the matrix A and the collection F , we use the same method that we used to construct RAN instances, and we set $s = 0.05$ and $p = 0.05$. The data (i.e., the matrix A and the collection F) for this instance is provided in Appendix D.

The efficient frontier for this specific instance is illustrated in Figure 6.1. In this figure, all of the given 18 points are efficient points. The values of both criteria corresponding to these points are presented in Table 6.1.

Table 6.1: Values of the evaluation criteria corresponding to the efficient points of the illustrative example

Efficient Point ID	Minimize selected nodes	Maximize covered elements	Efficient Point ID	Minimize selected nodes	Maximize covered elements
1	0	0	10	9	40
2	1	8	11	10	42
3	2	14	12	11	43
4	3	19	13	12	45
5	4	23	14	13	46
6	5	27	15	14	47
7	6	31	16	15	48
8	7	34	17	16	49
9	8	37	18	17	50

In the next section, we propose two algorithms for obtaining the efficient frontier for the bi-criteria set covering problem with conflict constraints.

Figure 6.1: Efficient frontier corresponding to the illustrative example

6.2 Strategies for Obtaining the Non-dominated Points

Ehrgott and Gandibleux [27] categorized the strategies proposed for solving multi-objective optimization problems to three groups based on the information provided by the decision makers.

- **Priori mode:** In this type of strategies the decision makers share their preferences at the beginning of the process. Goal programming could be one example of these methods.
- **Posteriori mode:** The decision makers preferences are analyzed after obtaining all the non-dominated points.
- **Interactive mode:** This type of strategies require high involvement of the decision makers during the decision making process. During the computing steps, the preferences of the decision makers are introduced.

In this chapter, we focus on the posteriori mode strategies, i.e., we are mostly interested in algorithms which obtain the entire non-dominated frontier. Most approaches proposed for solving the multi-objective optimization problem involve reformulating the multiple objective problem to a collection of single objective models. In this category, two of the efficient and popular methods in the literature for obtaining all non-dominated points are the two-phase method and the ϵ -constraint approach. The two-phase method has two phases. In the first phase, all the supported efficient solutions are obtained by solving weighted sum formulations of the problem. In the second phase, the non-supported efficient solutions are obtained by applying some bounds and problem specific methods. On the other hand, in ϵ -constraint approach, one criterion is posed as a constraint, while the other criterion is kept as an objective function. We use

the ϵ -constraint approach for obtaining all non-dominated points for the model $Bi-IP$, and we discuss it in details in the next subsection.

6.2.1 The ϵ -constraint Approach

In order to obtain all the efficient solutions of the model $Bi-IP$, we can use the ϵ -constraint method, which is a well established practice in this field [18, 29]. In this context, we can keep either objective function as the primary objective function while incorporating the other objective function into the model as a constraint. Hence, we define the following two approaches.

6.2.1.1 Approach 1

In this approach, we keep the first objective function (maximizing the number of covered elements) as the primary objective function while we incorporate the second objective function (minimizing the number of selected nodes) into the model as a constraint. We denote the right-hand-side of this constraint by ϵ_1 , and we denote the new model by $IP-\epsilon_1$. The model is shown below.

$$\text{maximize} \quad \sum_{i=1}^m y_i \quad (IP-\epsilon_1)$$

$$\text{subject to} \quad \sum_{j=1}^n a_{ij}x_j \geq y_i \quad \forall i = 1, \dots, m \quad (6.2a)$$

$$x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F \quad (6.2b)$$

$$\sum_{j=1}^n x_j \leq \epsilon_1 \quad (6.2c)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (6.2d)$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \dots, m \quad (6.2e)$$

For each value of ϵ_1 , the optimal solution (x^*, y^*) for this model is an efficient (or weakly efficient) solution for the problem Bi-SCC, and by systematically increasing the value of ϵ_1 and solving the model $IP-\epsilon_1$ in the context of the ϵ -constraint algorithm, we can obtain all efficient solutions of the problem Bi-SCC. We refer to the resulting algorithm as Algorithm 1. Following is a brief description of Algorithm 1:

1. Set $\epsilon_1 = 0$
2. Solve the model $IP-\epsilon_1$, and denote its objective function value as z_1^* .
3. If $z_1^* = m$ or $\epsilon_1 = n$, stop the algorithm (see Remark 6). Otherwise, go to next step.

4. $\epsilon_1 \leftarrow \epsilon_1 + 1$. Go to step 2.

Remark 6. In step 3 of the Algorithm 1, if $z_1^* = m$, then we stop the process, since m is the largest possible value for this objective function, and further increasing the value of parameter ϵ_1 cannot result in a better (larger) value for the objective function. Also, in this step, we stop the process if $\epsilon_1 = n$, since n is the largest possible value for this parameter at any feasible solution.

This algorithm terminates after at most $n + 1$ iterations, since we increase ϵ_1 by one at each iteration and we stop the algorithm if $\epsilon_1 = n$.

6.2.1.2 Approach 2

In this approach, we keep the second objective function (minimizing the number of selected nodes) as the primary objective function and we incorporate the first objective function (maximizing the number of covered elements) into the model as a constraint. We denote the right-hand-side of this constraint by ϵ_2 , and we denote the new model by $IP-\epsilon_2$. The model is shown below.

$$\text{minimize} \quad \sum_{j=1}^n x_j \quad (IP-\epsilon_2)$$

$$\text{subject to} \quad \sum_{j=1}^n a_{ij}x_j \geq y_i \quad \forall i = 1, \dots, m \quad (6.3a)$$

$$x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F \quad (6.3b)$$

$$\sum_{i=1}^m y_i \geq \epsilon_2 \quad (6.3c)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (6.3d)$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \dots, m \quad (6.3e)$$

For each value of ϵ_2 , the optimal solution (x^*, y^*) for this model is an efficient (or weakly efficient) solution for the problem Bi-SCC, and by systematically increasing the value of ϵ_2 and solving the model $IP-\epsilon_2$ in the context of the ϵ -constraint algorithm, we can obtain all efficient solutions of the problem Bi-SCC. We refer to the resulting algorithm as Algorithm 2. Following is a brief description of Algorithm 2:

1. Set $\epsilon_2 = 0$
2. Solve the model $IP-\epsilon_2$, and denote its objective function value as z_2^* . Also denote the set of indexes of all selected nodes as S^* , i.e., $S^* = \{j : x_j^* = 1\}$. Find the number of elements

covered by at least one node in S^* , and denote it as e .

3. If $z_2^* = n$ or $e = m$, stop the algorithm (see Remark 7). Otherwise, go to next step.
4. $\epsilon_2 \leftarrow e + 1$ (see Remark 8). Go to step 2.

Remark 7. *In step 3 of the Algorithm 2, if $z_2^* = n$, then we stop the process, since n is the largest possible value for this objective function, and further increasing the value of parameter ϵ_2 cannot result in a worse (larger) value for the objective function. Also, in this step, we stop the process if $e = m$, since m is the largest possible value for the number of covered elements, and solving the model for larger values of ϵ_2 cannot result in finding new efficient solutions.*

Remark 8. *In step 4 of Algorithm 2, we set the new value of ϵ_2 to $e + 1$, since with the previous value of ϵ_2 , e elements were covered by selected nodes, and in the next iteration we look for a new efficient solution in which the number of covered elements is larger than e .*

This algorithm clearly terminates after at most $m + 1$ iterations, since we increase ϵ_2 by at least one at each iteration and we stop the algorithm if $e = m$.

6.3 A Computational Experiment

In this section, we present the results of a limited experiment that we carried out to assess the computational requirements of the two algorithms that we described above. In this experiment, we constructed several instances of the bi-criteria problem Bi-SCC from the GEO-10 and RAN libraries (as described in Chapter 3) and solved each instance, i.e., obtained the entire efficient frontier for each instance, via both algorithms.

We observed that for relatively small instances of the problem, we were able to obtain the efficient frontier relatively quickly using either algorithm, but for larger instances of the problem, the computational requirements of both algorithms become excessive. This is mainly due to the excessive amount of time required to solve the corresponding IP models at each iteration. In subsequent chapters, we propose appropriate methodologies to overcome this difficulty for larger instances of the problem. In the remainder of this section, we describe the computational experiment that we carried out and make a few observations.

In Table 6.2, we present the results for a few instances of the problem. Each row in this table corresponds to an instance of the problem. As stated above, we solved each instance via both algorithms and obtained its corresponding efficient frontier. For each instance, we report the values of m and n (showing the size of instance), s or r_s (showing the density of the matrix A), and p or r_p (showing the size of the conflict collection F) as well as Ψ (number of non-dominated points obtained). Also for each instance, we report the total execution time as

well as the number of iterations. A dash-line entry in this table implies that the corresponding execution time exceeds our self-imposed time limit of 7200 seconds for obtaining all the non-dominated points for one instance. In this experiment, all heuristics and cut generation options of CPLEX are turned on.

Table 6.2: Execution time (seconds) of the two algorithms

ID	m	n	s or r_s	p or r_p	Ψ	Algorithm 1		Algorithm 2	
						Exec. time 1 (sec)	No. of Iter. 1	Exec. time 2 (sec)	No. of Iter. 2
G10.1	100	1000	850	700	17	29.1	17	37.6	35
G10.2			850	850	18	295.6	18	465.1	40
G10.3			850	950	20	-	-	2,833.4	43
G10.4			850	1050	-	-	-	-	-
G10.9			1050	850	6	11.7	6	17.5	13
G10.10			1050	1050	7	73.1	7	328.6	15
G10.11	300	1000	850	700	29	-	-	2,301.4	85
G10.12			850	850	-	-	-	-	-
G10.16			1050	850	10	979.7	10	279.4	20
G10.17			1050	950	10	5,008.5	10	1,702.5	24
G10.18			1050	1050	-	-	-	-	-
R.1	100	300	0.05	0.01	18	86.9	18	81.4	49
R.2			0.05	0.05	19	1,490.4	19	285.8	39
R.3			0.05	0.1	21	3,475.3	21	986.8	40
R.4			0.1	0.05	12	1,257.8	12	361.2	35
R.5			0.1	0.1	12	1,887.8	12	565.0	33
R.7	100	1000	0.01	0.01	39	35.6	39	54.4	59
R.8			0.01	0.05	-	-	-	-	-
R.9			0.05	0.01	15	-	-	1,274.0	45

We make the following observations from Table 6.2;

1. We observe that Algorithm 1 failed to solve 7 and Algorithm 2 failed to solve 4 out of 19 instances within pre-defined time limit of 2 hours.
2. The execution time for algorithm 2 appears to be significantly smaller than that of algorithm 1, especially for the instances where the corresponding execution times are relatively large. Out of the 15 instances that at least one of the algorithms could solve within 2 hours, we observe that total execution time of algorithm 1 is larger than algorithm 2

for 10 instances. We also observe that the number of iterations in algorithm 2 is larger than algorithm 1 for all instances that we could solve in this experiment. The reason behind this phenomenon is that Algorithm 2 finds weakly non-dominated solutions in many iterations.

In the above experiment, we also noticed that the CPU time required to solve IP models $IP-\epsilon_1$ and $IP-\epsilon_2$ varies significantly for different values of ϵ_1 and ϵ_2 , respectively. To illustrate this observation, we present the corresponding CPU times for instance G10.17 in Tables 6.3 and 6.4. In these tables, z_1^* and z_2^* show the optimal objective function value of the models $IP-\epsilon_1$ and $IP-\epsilon_2$ for each corresponding ϵ_1 and ϵ_2 , respectively.

From Table 6.3, we observe that the execution time grows exponentially from $\epsilon_1 = 0$ to $\epsilon_1 = 7$, and then it starts to decrease for larger values of ϵ_1 . From Table 6.4, we observe that the execution time of the model $IP-\epsilon_2$ tends to be larger for larger values of ϵ_2 . For all other instances in our experiment, we observed a similar trend.

Table 6.3: Execution times of the model $IP-\epsilon_1$ for instance G10.17

ϵ_1	z_1^*	$IP-\epsilon_1$ time (sec)
0	0	1.5
1	178	1.8
2	244	2.3
3	262	37.7
4	276	44.7
5	284	84.8
6	291	118.3
7	295	4,297.7
8	299	288.4
9	300	131.5

In order to reduce the computational requirements of both algorithms for solving relatively medium and large size instances of the problem Bi-SCC, we pursue two strategies.

Strategy 1: Design more effective methods to solve the models $IP-\epsilon_1$ and $IP-\epsilon_2$. One way to reduce the computational requirements of both algorithms is to design more effective methods for solving the integer programming models $IP-\epsilon_1$ and $IP-\epsilon_2$. These problems have structural similarities with problem SCC and it is conceivable that this relationship can be exploited to design more effective methods for solving this problem in the context of algorithms 1 and 2. To this end, in Chapter 7 we extend the pre-processing procedures and valid inequalities

Table 6.4: Execution times of the model $IP-\epsilon_2$ for instance G10.17

ϵ_2	z_2^*	$IP-\epsilon_2$ time (sec)	ϵ_2	z_2^*	$IP-\epsilon_2$ time (sec)
0	0	0.5	281	5	47.2
1	1	1.6	282	5	75.1
44	1	1.9	284	5	63.9
179	2	3.0	285	6	91.8
245	3	4.6	288	6	59.1
254	3	4.2	292	7	132.1
263	4	35.6	293	7	103.0
267	4	35.5	294	7	139.1
269	4	36.8	295	7	97.8
271	4	34.2	296	8	253.4
274	4	34.6	299	8	248.1
277	5	45.1	300	9	154.3

that we introduced in the previous chapters for solving the problem SCC, and we use these modified pre-processing procedures and valid inequalities in the context of a B&B algorithm to solve the models $IP-\epsilon_1$ and $IP-\epsilon_2$. Subsequently, we carry out a computational experiment to assess the impact of the pre-processing procedures and the VIs in solving these problems.

Strategy 2: Design effective inexact methods to solve the model $Bi-IP$. Another approach for solving this problem is to design effective inexact methods for solving the model $Bi-IP$. To this end, in Chapter 8, we design a heuristic procedure for solving the model $Bi-IP$ and carry out a computational experiment to evaluate its effectiveness. The computational requirements of this heuristic procedure are significantly smaller than those of the algorithms 1 and 2 for larger instances of the problem, hence this procedure can be used in some occasions where the execution time of the algorithms 1 and 2 to obtain the optimal solution of the model $IP-\epsilon_1$ and $IP-\epsilon_2$ is excessive.

Chapter 7

Exact Methods for Solving Models $IP-\epsilon_1$ and $IP-\epsilon_2$

In this chapter, we propose appropriate pre-processing procedures and Valid Inequalities (VIs) for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. Due to the structured similarities of these models and the IP model for the problem SCC that we discussed earlier, the pre-processing procedures and the VIs that we propose here are also similar to those that we discussed in previous chapters for the problem SCC. Subsequently, we employ these procedures and VIs in the context of a Branch-and-Bound (B&B) algorithm for solving the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

7.1 Bicriteria-Covering-Packing Valid Inequalities

In this section, we introduce the Bicriteria-Covering-Packing Valid Inequalities (BCP-VIs) for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. These VIs are similar to the Covering-Packing Valid Inequalities that we introduced in Chapter 4 in the context of the problem SCC.

Recall that $N_i^k = \{j : j \in N_i, (j, k) \notin F\}$ for each element $i \in M$ and node $k \in N \setminus N_i$. The following proposition explains the BCP-VIs.

Proposition 8. *For each element $i \in M$ and node $k \in N \setminus N_i$, the constraint*

$$\sum_{j: j \in N_i^k} x_j \geq x_k + y_i - 1 \tag{7.1}$$

is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

The BCP-VI states that for each element i and each node k that does not cover element i , if in a feasible solution $(\bar{x}, \bar{y})$, element i is covered ($\bar{y}_i = 1$) and node k is selected ($\bar{x}_k = 1$),

then at least one of the nodes that cover element i and do not have conflict with node k must be selected to be in that feasible solution.

Through the following numeric examples, we show that the proposed BCP-VIs can eliminate part of the LP feasible region of the models $IP-\epsilon_1$ and $IP-\epsilon_2$, i.e., they are not redundant.

Example 8. Consider an instance of the problem $IP-\epsilon_1$ shown by the following IP model with $m = 4$ covering inequalities, two conflict constraint, and $\epsilon_1 = 2$.

$$\begin{aligned}
& \text{maximize} && \sum_{i=1}^4 y_i && (EX8-IP) \\
& \text{subject to} && x_1 + x_3 + x_4 \geq y_1 && (\text{element 1}) \\
& && x_2 + x_3 + x_4 \geq y_2 && (\text{element 2}) \\
& && x_5 + x_6 \geq y_3 && (\text{element 3}) \\
& && x_5 + x_7 \geq y_4 && (\text{element 4}) \\
& && x_3 + x_5 \leq 1 \\
& && x_4 + x_5 \leq 1 \\
& && \sum_{j=1}^7 x_j \leq 2 \\
& && y_i \in \{0, 1\} && \forall i = 1, \dots, 4 \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, 7
\end{aligned}$$

We solve the linear programming relaxation of this model via simplex algorithm and obtain the following optimal solution:

$$\begin{aligned}
x_1^* &= x_2^* = x_6^* = 0 \\
x_3^* &= x_4^* = x_5^* = x_7^* = 0.5 \\
y_3^* &= 0.5 \\
y_1^* &= y_2^* = y_4^* = 1
\end{aligned}$$

The corresponding optimal value of the objective function is $z^* = 3.5$.

According to the definition of BCP-VIs, and setting $i = 1$ and $k = 5$, we get the following BCP-VI,

$$x_1 \geq x_5 + y_1 - 1 \tag{7.2}$$

and setting $i = 2$ and $k = 5$, we get the following BCP-VI.

$$x_2 \geq x_5 + y_2 - 1 \tag{7.3}$$

We observe that both BCP-VIs 7.2 and 7.3 are violated at (x^*, y^*) . □

Example 9. Consider an instance of the problem $IP-\epsilon_2$ shown by the following IP model with covering inequalities and conflict constraints similar to the previous example, and $\epsilon_2 = 4$.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^7 x_j && (EX9-IP) \\
& \text{subject to} && x_1 + x_3 + x_4 \geq y_1 && (\text{element 1}) \\
& && x_2 + x_3 + x_4 \geq y_2 && (\text{element 2}) \\
& && x_5 + x_6 \geq y_3 && (\text{element 3}) \\
& && x_5 + x_7 \geq y_4 && (\text{element 4}) \\
& && x_3 + x_5 \leq 1 \\
& && x_4 + x_5 \leq 1 \\
& && \sum_{i=1}^4 y_i \geq 4 \\
& && y_i \in \{0, 1\} && \forall i = 1, \dots, 4 \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, 7
\end{aligned}$$

We solve the linear programming relaxation of this model via simplex algorithm and obtain the following optimal solution:

$$\begin{aligned}
x_1^* &= x_2^* = 0 \\
x_3^* &= x_4^* = x_5^* = x_6^* = x_7^* = 0.5 \\
y_1^* &= y_2^* = y_3^* = y_4^* = 1
\end{aligned}$$

The corresponding optimal value of the objective function is $z^* = 2.5$.

BCP-VIs 7.2 and 7.3 are also valid for the model $EX9-IP$, and we observe that they are violated at (x^*, y^*) . $\square$

These examples show that the BCP-VIs can remove a segment of the feasible region of the LP relaxation for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. Hence they are not redundant and can be effective for solving these problems in the context of algorithms 1 and 2.

7.2 Bicriteria-Covering-Clique Valid Inequalities

In this section, we introduce the Bicriteria-Covering-Clique Valid Inequalities (BCC-VIs) for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. These VIs are similar to the Covering-Clique Valid Inequalities that we introduced in Chapter 4 in the context of the problem SCC.

The following proposition explains the BCC-VIs.

Proposition 9. *Given a clique q of size k , let i_j be any element of the cover-set of the j^{th} node of the clique, for $j = 1, \dots, k$. If no node in the clique q appears in every covering node-set for elements $i_1, i_2, \dots, i_k$, (i.e., in every one of the covering node-sets $N_{i_1}, N_{i_2}, \dots, N_{i_k}$) then the constraint*

$$\sum_{j \in (N_{i_1} \cup N_{i_2} \cup \dots \cup N_{i_k}) \setminus q} x_j \geq \sum_{l=1}^k y_{i_l} - (k-1), \quad (7.4)$$

is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

Proof. In a feasible solution $(\bar{x}, \bar{y})$, if at least one of the elements $i_1, i_2, \dots, i_k$ is not covered, then this inequality follows since the right hand side becomes non-positive. Similarly, in a feasible solution $(\bar{x}, \bar{y})$, if all of the elements $i_1, i_2, \dots, i_k$ are covered ($\bar{y}_{i_l} = 1, \forall l = 1, \dots, k$) and none of the nodes in the clique q is selected ($\bar{x}_d = 0, \forall d \in q$), this inequality follows, since at least one node in each of the sets $N_{i_1}, N_{i_2}, \dots, N_{i_k}$ must be selected to be in that feasible solution.

On the other hand, if all of the elements $i_1, i_2, \dots, i_k$ are covered ($\bar{y}_{i_l} = 1, \forall l = 1, \dots, k$) and any node $h \in q$ is selected to be in a feasible solution ($\bar{x}_h = 1$), it follows that there is at least one covering node-set N_{i_l} such that $h \notin N_{i_l}$. Since node h is in the clique q , no other node in q can be selected to be in that feasible solution. So the set covering constraint

$$\sum_{j: j \in N_{i_l}} x_j \geq 1$$

reduces to

$$\sum_{j \in N_{i_l} \setminus q} x_j \geq 1,$$

and therefore the inequality 7.4 follows. $\square$

Considering each conflict constraint as a clique of size 2, we derive the following corollary from Proposition 9.

Corollary 3. *For each conflicting pair of nodes $(r, s) \in F$, if we have two elements i_1 and i_2 such that each of the nodes r and s appears in only one of their corresponding covering node-sets N_{i_1} and N_{i_2} , then the constraint*

$$\sum_{j \in (N_{i_1} \cup N_{i_2}) \setminus \{r, s\}} x_j \geq y_{i_1} + y_{i_2} - 1, \quad (7.5)$$

is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

Through the following numeric examples, we show that the proposed BCC-VIs can eliminate part of the LP feasible region of the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

Example 10. Consider an instance of the problem $IP\text{-}\epsilon_1$ shown by the following IP model with $m = 4$ covering inequalities, one conflict constraint, and $\epsilon_1 = 2$.

$$\begin{aligned}
& \text{maximize} && \sum_{i=1}^4 y_i && (EX10\text{-}IP) \\
& \text{subject to} && x_1 + x_2 + x_3 \geq y_1 && (\text{element 1}) \\
& && x_1 + x_2 + x_4 \geq y_2 && (\text{element 2}) \\
& && x_3 + x_5 \geq y_3 && (\text{element 3}) \\
& && x_4 + x_6 \geq y_4 && (\text{element 4}) \\
& && x_3 + x_4 \leq 1 \\
& && \sum_{j=1}^6 x_j \leq 2 \\
& && y_i \in \{0, 1\} \quad \forall i = 1, \dots, 4 \\
& && x_j \in \{0, 1\} \quad \forall j = 1, \dots, 6
\end{aligned}$$

We solve the linear programming relaxation of this model via simplex algorithm and obtain the following optimal solution:

$$\begin{aligned}
x_2^* &= x_5^* = 0 \\
x_1^* &= x_3^* = x_4^* = x_6^* = 0.5 \\
y_3^* &= 0.5 \\
y_1^* &= y_2^* = y_4^* = 1
\end{aligned}$$

The corresponding optimal value of the objective function is $z^* = 3.5$.

According to the definition of BCC-VIs, and considering element $1 \in M_3$ and element $2 \in M_4$, and their corresponding covering-node sets $N_1 = \{1, 2, 3\}$ and $N_2 = \{1, 2, 4\}$, we get the following BCC-VI.

$$x_1 + x_2 \geq y_1 + y_2 - 1 \tag{7.6}$$

We observe that BCC-VI 7.6 is violated at (x^*, y^*) . $\square$

Example 11. Consider an instance of the problem $IP\text{-}\epsilon_2$ shown by the following IP model with

covering inequalities and conflict constraints similar to the previous example, and $\epsilon_2 = 4$.

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^6 x_j && (EX11-IP) \\
& \text{subject to} && x_1 + x_2 + x_3 \geq y_1 && (\text{element 1}) \\
& && x_1 + x_2 + x_4 \geq y_2 && (\text{element 2}) \\
& && x_3 + x_5 \geq y_3 && (\text{element 3}) \\
& && x_4 + x_6 \geq y_4 && (\text{element 4}) \\
& && x_3 + x_4 \leq 1 \\
& && \sum_{i=1}^4 y_i \geq 4 \\
& && y_i \in \{0, 1\} && \forall i = 1, \dots, 4 \\
& && x_j \in \{0, 1\} && \forall j = 1, \dots, 6
\end{aligned}$$

We solve the linear programming relaxation of this model via simplex algorithm and obtain the following optimal solution:

$$x_2^* = 0$$

$$x_1^* = x_3^* = x_4^* = x_5^* = x_6^* = 0.5$$

$$y_1^* = y_2^* = y_3^* = y_4^* = 1$$

The corresponding optimal value of the objective function is $z^* = 2.5$.

BCC-VI 7.6 is valid for the model *EX11-IP*, and we observe that it is violated at (x^*, y^*) . □

These examples show that the BCC-VIs can remove a segment of the feasible region of the LP relaxation for the models *IP- ϵ_1* and *IP- ϵ_2* . Hence they are not redundant and can be effective for solving these problems in the context of algorithms 1 and 2.

7.3 Bicriteria-Conflicting-Set Valid Inequalities

In this section, we introduce the Bicriteria-Conflicting-Set Valid Inequalities (BCS-VIs) for the models *IP- ϵ_1* and *IP- ϵ_2* . These VIs are similar to the Conflicting-Set Valid Inequalities that we introduced in Chapter 4 in the context of the problem SCC.

The following proposition explains the BCS-VIs.

Proposition 10. *For every element $i \in M$, if there exists a set $\overline{N} \subseteq N - N_i$ such that*

$N_i \subseteq \bigcup_{j \in \overline{N}} G_j$, then the constraint

$$\sum_{j \in \overline{N}} x_j \leq |\overline{N}| - y_i \quad (7.7)$$

is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

Proof. In a feasible solution $(\bar{x}, \bar{y})$, if element i is not covered ($\bar{y}_i = 0$), then the inequality 7.4 clearly follows.

On the other hand, in a feasible solution $(\bar{x}, \bar{y})$, if element i is covered ($\bar{y}_i = 1$) and we select all nodes in $\overline{N}$ to be in that feasible solution ($\bar{x}_j = 1, \forall j \in \overline{N}$), then none of the nodes in $\bigcup_{j \in \overline{N}} G_j$ can be selected to be in that solution since each node in $\overline{N}$ is in conflict with at least one node in $\bigcup_{j \in \overline{N}} G_j$. Since $N_i \subseteq \bigcup_{j \in \overline{N}} G_j$, it follows that none of the nodes in N_i is in the corresponding solution, hence element i cannot be covered. Thus in every feasible solution if element i is covered ($\bar{y}_i = 1$), then at least one of the nodes in $\overline{N}$ must be not present. Hence, the inequality 7.7 follows. $\square$

7.4 MC-VIs and CD-VIs

Clearly, Maximal Clique Valid Inequalities (MC-VIs) are valid for the models $IP-\epsilon_1$ and $IP-\epsilon_2$, and we can add them to these models without modification.

Similarly, Column Domination Valid Inequalities (CD-VIs) are clearly valid for the set of optimal solutions of the model $IP-\epsilon_2$. It is easy to see that these VIs are not necessarily satisfied by every optimal solution of the model $IP-\epsilon_1$ if the corresponding problem has multiple optimal solutions. But these inequalities do not remove those optimal solutions that are non-dominated solutions for the problem Bi-SCC. Hence, CD-VI is a valid inequality for the set of optimal solutions of the model $IP-\epsilon_1$ that are non-dominated for the problem Bi-SCC.

7.5 Other Valid Inequalities

In this section, we introduce two new families of valid inequalities for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. These valid inequalities are inspired by the pre-processing procedures that we introduced in Chapter 3 for the problem SCC. The following two propositions explain these families of valid inequalities.

Proposition 11. *For each element i ($i \in M$), if there exists another element r ($r \in M, r \neq i$) such that $N_r \subseteq N_i$, then the constraint*

$$y_i \geq y_r \quad (7.8)$$

is a valid inequality for the set of optimal solutions of the model $IP-\epsilon_1$.

Proof. This valid inequality is derived from row reduction #1 defined in Chapter 3. For every two elements i and r ($i, r \in M$), if $N_r \subseteq N_i$, i.e., if all the nodes that cover element r also cover element i , this inequality states that every feasible solution $(\bar{x}, \bar{y})$ with $\bar{y}_r = 1$ and $\bar{y}_i = 0$ cannot be an optimal solution. The reason is that, since element r is covered, then element i can also be covered and so $\bar{y}_i$ can be set to 1 in order to increase the objective function value while maintaining the feasibility of the problem. $\square$

Remark 9. *The inequality 7.8 is not necessarily satisfied by every optimal solution of the model $IP-\epsilon_2$ if the problem has multiple optimal solutions. But this inequality does not remove those optimal solutions that are non-dominated solutions for the problem Bi-SCC. Hence, the inequality 7.8 is a valid inequality for the set of optimal solutions of the model $IP-\epsilon_2$ that are non-dominated for the problem Bi-SCC.*

Proposition 12. *For each element i ($i \in M$), if it is only covered by a single node j , then the constraint*

$$x_j \geq y_i \tag{7.9}$$

is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$.

Proof. This valid inequality is derived from column reduction #1 defined in Chapter 3. In a feasible solution $(\bar{x}, \bar{y})$, if element i is not covered ($\bar{y}_i = 0$), then the inequality 7.9 clearly follows. On the other hand, in a feasible solution $(\bar{x}, \bar{y})$, if element i is covered ($\bar{y}_i = 1$), then since it is only covered by single node j , node j must be selected ($\bar{x}_j = 1$), and therefore the inequality 7.9 follows. $\square$

7.6 Computational Effectiveness of Valid Inequalities

In the previous sections, we introduced several families of valid inequalities and showed that these inequalities are valid for both models $IP-\epsilon_1$ and $IP-\epsilon_2$. We carried out a preliminary experiment to evaluate the effectiveness of these VIs in solving the corresponding models. In our experiment, we constructed several instances of the models $IP-\epsilon_1$ and $IP-\epsilon_2$ from instances in data sets A and B (as described in Chapter 5) as well as data sets GEO-10D and RAN (as described in Chapter 3), and by choosing different values for ϵ_1 and ϵ_2 . We then identified and added these VIs to models $IP-\epsilon_1$ and $IP-\epsilon_2$, and we solved each model with each family of VIs separately.

Our findings were similar to those we reported in previous chapters with respect to the single objective model $SCC-IP$, and for brevity, we do not report the details here. Following is a summary of each experiment and the corresponding observations that we made.

1. For adding BCP-VIs and BCC-VIs, we used a cutting plane algorithm (CPA) similar to what we explained in Section 5.1 of Chapter 5, and for adding CD-VIs and MC-VIs, we used a method similar to what we described in Section 5.4.3 of Chapter 5. We observed that the impact of these VIs is similar to what we observed earlier in Chapter 5. In particular, we observed that adding BCP-VIs to the models $IP-\epsilon_1$ and $IP-\epsilon_2$ can be quite effective for the instances from data set A, but for the instances in data sets GEO-10D and RAN, the impact is relatively small. Similarly, the impact of BCC-VIs was quite significant for instances from data set B, but it was relatively small for the instances in data sets GEO-10D and RAN.
2. We observed that adding MC-VIs and CD-VIs to the models $IP-\epsilon_1$ and $IP-\epsilon_2$ for GEO-10D and RAN instances results in significant improvements in reducing the execution time of the corresponding models.
3. In Section 7.5, we derived two new families of valid inequalities from pre-processing procedures that we defined for the problem SCC in Chapter 3, and we showed that these VIs are valid for both models $IP-\epsilon_1$ and $IP-\epsilon_2$. Also in Section 7.3, we introduced Bicriteria-Conflicting-Set Valid Inequalities (BCS-VIs), and we showed that BCS-VI is a valid inequality for the models $IP-\epsilon_1$ and $IP-\epsilon_2$. Our preliminary experiment indicated that adding these VIs to either model does not have a significant impact on the computational effort required to solve the corresponding model. The underlying reason behind this phenomenon is that for most of the instances in our experiment, the number of such VIs is relatively small. As a result, we conclude that these VIs are not effective in solving the models $IP-\epsilon_1$ and $IP-\epsilon_2$ for the instances in GEO-10D and RAN libraries.

7.7 A Strategy for Adding the Valid Inequalities

To include BCP-VIs, BCC-VIs, CD-VIs, and MC-VIs that we introduced in this chapter within a B&B framework, we can use different orderings of adding them to the models $IP-\epsilon_1$ and $IP-\epsilon_2$. In order to observe the computational impact of various strategies in this context, we performed a pilot study in which we experimented on several strategies with different orderings of adding VIs to these models, and we selected the strategy with the best results with respect to the execution time. We refer to this strategy as the *Four-Phase Method*.

In this section, we explain our proposed four-phase method in details, and perform a computational experiment to assess the effectiveness of this method for solving the models $IP-\epsilon_1$ and $IP-\epsilon_2$. Following is a description of each phase of the method.

1. Identify all CD-VIs and add them to the model.

2. Add a subset of MC-VIs to the resulting model (see Remark 10).
3. Let H be the collection of BCC-VIs that are identified for cliques of size 2 as well as all BCP-VIs; Apply an iterative cutting plane algorithm similar to the CPA explained in Chapter 5 to the resulting model and add all violated VIs that are in H to the resulting model.
4. Solve the resulting model via B&B algorithm.

Remark 10. *In phase 2, we consider all conflicts corresponding to the identified CD-VIs as new edges of the conflict graph G . Subsequently, we identify a subset of the maximal cliques on the updated graph G in a manner similar to the method we described in Chapter 5. We then add all the corresponding MC-VIs to the model, and remove all original conflict constraints as well as CD-VIs from the corresponding model, since they are redundant in the presence of MC-VIs.*

We carried out two computational experiments to assess the computational impact of adding BCP-VIs, BCC-VIs, CD-VIs, and MC-VIs to models $IP-\epsilon_1$ and $IP-\epsilon_2$ via the four-phase method. To this end, we consider a selected group of instances from data sets GEO-10D and RAN (as described in Chapter 3). In Table 7.1, we report the parameter values for the selected instances. In both experiments, we turn on all heuristics and cut generation options of CPLEX, and we impose a time limit of 7200 seconds on all instances.

Table 7.1: Selected instances from data sets GEO-10D and RAN

ID	m	n	s or r_s	p or r_p
G10.3	100	1000	850	950
G10.4			850	1050
G10.10			1050	1050
G10.12	300	1000	850	850
G10.16			1050	850
G10.17			1050	950
R.2	100	300	0.05	0.05
R.3			0.05	0.1
R.4			0.1	0.05
R.7	100	1000	0.01	0.01
R.8			0.01	0.05
R.9			0.05	0.01

7.7.1 First Experiment

In the first experiment, we evaluate the impact of adding VIs to the model $IP-\epsilon_1$. To this end, for each selected instance (in Table 7.1), we obtain the maximum value of the parameter ϵ_1 by solving the corresponding SCC problem, and we denote it as ϵ_1^{max} . Then, for each instance, we construct four instances of the model $IP-\epsilon_1$ by fixing the value of ϵ_1 at each of the values $\lfloor 0.3\epsilon_1^{max} \rfloor$, $\lfloor 0.5\epsilon_1^{max} \rfloor$, $\lfloor 0.7\epsilon_1^{max} \rfloor$, and $\lfloor 0.9\epsilon_1^{max} \rfloor$. We then solve the corresponding instances of the model $IP-\epsilon_1$ twice, once via B&B method (i.e., without adding VIs) and once with the four-phase method (i.e., with VIs added to the model). In Tables 7.2 and 7.3, we report the results of adding VIs to the instances of the model $IP-\epsilon_1$ constructed from GEO-10D and RAN instances, respectively.

We make the following observations from Table 7.2;

1. The B&B method failed to solve 2 instances and the four-phase method failed to solve 1 instance out of 24 instances within our self-imposed time limit.
2. The execution time of the four-phase method is significantly smaller than that of the B&B method. More specifically, for those instances that both methods terminated within our time limit, the execution time of the four-phase method is 80.5% smaller than that of the B&B method, on average.
3. For all instances, the execution time of both methods is considerably larger for larger values of ϵ_1 .
4. The impact of VIs in the context of the four-phase method is more significant for the instances with larger execution times.

We also make the following observations from Table 7.3;

1. Both B&B method and the four-phase method could solve each of the 24 instances within the time limit of 2 hours.
2. The execution time of the four-phase method is smaller than that of the B&B method, on average. More specifically, 17 out of 24 instances were relatively easy to solve as both methods could solve them in less than 20 seconds. For the remaining 7 instances, the execution time of the B&B method is smaller for 1 instance (R.2 with $\epsilon_1 = 16$), and larger for the other 6 instances, than the execution time of the four-phase method. Overall, we observe that the execution time of the four-phase method for all the instances in this table is 12.5% smaller than that of the B&B method, on average.
3. For all instances, the execution time of both methods tends to be larger for larger values of ϵ_1 .

Table 7.2: Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_1$ for GEO-10D instances

ID	ϵ_1^{max}	ϵ_1	B&B Method	Four-Phase Method				
			Total time 1 (sec)	# of MC-VIs	# of BCC-VIs	# of BCP-VIs	$IP-\epsilon_1$ time (sec)	Total time 2 (sec)
G10.3	19	5	2.7	4,259	199	172	3.4	14.5
		9	70.2		104	226	12.2	20.3
		13	223.7		68	241	18.4	27.2
		17	2,253.2		94	460	449.3	470.1
G10.4	23	6	67.5	4,331	104	292	3.6	18.4
		11	682.0		32	432	62.7	76.5
		16	4,167.5		31	503	245.9	261.1
		20	-		46	1,248	4,631.8	4,684.5
G10.10	6	1	2.0	4,505	0	0	1.0	9.3
		3	5.2		90	323	3.0	13.2
		4	37.8		53	274	5.6	14.5
		5	59.7		37	345	49.5	66.3
G10.12	31	9	71.9	3,703	199	177	30.7	39.7
		15	527.0		119	122	229.1	237.9
		21	6,449.9		87	95	1,683.4	1,693.6
		27	-		-	-	-	-
G10.16	9	2	1.0	3,684	1,090	83	6.1	7.7
		4	11.4		99	52	5.0	9.3
		6	47.3		25	28	14.7	18.2
		8	739.6		23	9	76.6	81.9
G10.17	9	2	1.2	4,350	1,686	137	4.9	9.4
		4	43.1		166	337	15.1	19.2
		6	191.4		29	125	31.0	35.1
		8	1,255.3		407	275	92.9	148.6

Table 7.3: Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_1$ for RAN instances

			B&B Method	Four-Phase Method				
ID	ϵ_1^{max}	ϵ_1	Total time 1 (sec)	# of MC-VIs	# of BCC-VIs	# of BCP-VIs	$IP-\epsilon_1$ time (sec)	Total time 2 (sec)
R.2	18	5	0.4	1,602	24	2	0.5	1.1
		9	1.1		70	8	1.1	1.8
		12	4.9		42	6	7.6	8.4
		16	75.8		171	56	288.9	292.7
R.3	20	6	0.9	2,082	77	15	0.6	1.6
		10	5.4		98	25	4.9	6.0
		14	601.0		47	31	352.5	353.8
		18	292.1		156	158	226.5	235.8
R.4	11	3	0.7	1,601	48	2	0.5	0.9
		5	1.5		14	2	1.0	1.4
		7	8.2		2	0	8.0	8.5
		9	321.1		0	0	359.4	359.8
R.7	38	11	0.6	2,306	0	0	1.4	2.3
		19	0.7		2	1	1.6	3.0
		26	1.0		3	1	1.8	3.2
		34	2.2		4	4	3.6	5.1
R.8	45	13	2.1	5,720	6	6	3.4	8.5
		22	6.4		20	24	6.2	9.9
		31	263.7		23	43	36.2	41.3
		40	1,194.7		124	222	862.0	878.7
R.9	14	4	1.3	5,639	0	0	1.4	2.3
		7	4.6		1	0	5.3	6.6
		9	11.9		1	0	13.2	14.9
		12	5,726.4		0	0	5,213.8	5,214.6

4. The impact of VIs in the context of the four-phase method is more significant for the instances with larger execution times.

Consequently, we observe that the impact of our proposed VIs in the context of the four-phase method for solving the model $IP-\epsilon_1$ is more significant for the GEO-10D instances than for the RAN instances. We also observe that the number of added BCP-VIs and BCC-VIs to the model in the context of the four-phase method for the RAN instances is significantly smaller than the corresponding numbers for the GEO-10D instances.

7.7.2 Second Experiment

In the second experiment, we evaluate the impact of adding VIs to the model $IP-\epsilon_s$. For each selected instance (in Table 7.1), the maximum value of the parameter ϵ_2 is the corresponding value of m . For each instance, we construct four instances of the model $IP-\epsilon_2$ by fixing the value of ϵ_2 at each of the values $\lfloor 0.9m \rfloor$, $\lfloor 0.95m \rfloor$, $m-2$, and $m-1$. We select the relatively large values of ϵ_2 since we observed that for smaller values of this parameter, the model becomes relatively easy to solve, and both the B&B method and the four-phase method can solve the model in a few seconds. We then solve the resulting instances of the model $IP-\epsilon_2$ twice, once via B&B method (i.e., without adding VIs) and once with the four-phase method (i.e., with VIs added to the model). In Tables 7.4 and 7.5, we report the results of adding VIs to the instances of the model $IP-\epsilon_2$ constructed from GEO-10D and RAN instances, respectively.

We make the following observations from Table 7.4;

1. The B&B method failed to solve 4 instances and the four-phase method failed to solve 2 instances out of the 24 instances within our self-imposed time limit.
2. The execution time of the four-phase method is significantly smaller than that of the B&B method. More specifically, for those instances that both methods terminated within our time limit, the execution time of the four-phase method is 85.4% smaller than that of the B&B method, on average.
3. For all instances, the execution time of both methods is considerably larger for larger values of ϵ_2 .
4. The impact of VIs in the context of the four-phase method is more significant for the instances with larger execution times.

We also make the following observations from Table 7.5;

1. Both the B&B method and the four-phase method were able to solve each of the 24 instances within the time limit of 2 hours.

Table 7.4: Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_2$ for GEO-10D instances

ID	ϵ_2	B&B Method	Four-Phase Method				
		Total time 1 (sec)	# of MC-VIs	# of BCC-VIs	# of BCP-VIs	$IP-\epsilon_2$ time (sec)	Total time 2 (sec)
G10.3	90	84.3	4,259	99	245	12.9	22.1
	95	312.6		67	214	28.8	37.7
	98	278.9		82	214	39.6	48.9
	99	683.7		78	225	38.3	47.1
G10.4	90	987.8	4,331	38	458	147.9	161.7
	95	6,472.7		33	443	308.0	321.3
	98	-		33	470	1,586.3	1,599.8
	99	-		25	459	686.2	699.5
G10.10	90	4.5	4,505	116	356	2.3	12.7
	95	7.7		103	310	3.7	15.7
	98	11.5		69	282	3.2	13.6
	99	62.6		42	248	7.3	17.5
G10.12	270	734.8	3,703	113	122	170.4	179.4
	285	5,714.9		94	118	1,276.6	1,288.0
	298	-		-	-	-	-
	299	-		-	-	-	-
G10.16	270	9.2	3,684	571	165	2.9	6.0
	285	6.4		130	51	7.2	10.4
	298	41.1		21	23	17.1	19.5
	299	25.1		27	31	10.3	13.1
G10.17	270	33.4	4,350	749	484	14.8	20.9
	285	89.0		183	365	22.4	26.8
	298	194.5		33	140	28.2	33.1
	299	328.7		36	121	43.1	47.4

Table 7.5: Results of adding all VIs in the context of four-phase method to the model $IP-\epsilon_2$ for RAN instances

		B&B Method	Four-Phase Method				
ID	ϵ_2	Total time 1 (sec)	# of MC-VIs	# of BCC-VIs	# of BCP-VIs	$IP-\epsilon_2$ time (sec)	Total time 2 (sec)
R.2	90	10.7	1,602	62	11	9.6	10.4
	95	8.8		43	6	22.5	23.6
	98	36.6		41	4	28.0	29.1
	99	48.5		35	5	43.5	44.3
R.3	90	34.9	2,082	71	24	17.8	18.9
	95	41.6		70	28	45.0	46.2
	98	142.0		59	16	104.3	105.2
	99	321.5		45	16	98.1	99.4
R.4	90	4.0	1,601	3	0	5.7	6.2
	95	26.4		2	0	27.7	28.2
	98	68.3		2	0	48.6	49.1
	99	92.1		0	0	74.0	74.3
R.7	90	0.7	2,306	4	4	2.5	5.2
	95	2.5		4	4	3.7	5.8
	98	3.5		5	2	3.9	5.4
	99	4.1		2	1	3.0	4.4
R.8	90	126.4	5,720	28	42	45.9	49.3
	95	881.4		32	52	178.3	184.9
	98	1,009.6		31	55	171.9	177.5
	99	2,334.3		33	68	192.6	197.2
R.9	90	39.0	5,639	0	0	20.6	21.5
	95	67.1		0	0	97.3	98.3
	98	182.4		0	0	191.5	192.2
	99	212.3		0	0	98.0	98.8

2. The execution time of the four-phase method is smaller than that of the B&B method, on average. More specifically, 13 out of 24 instances were relatively easy to solve as both methods could solve them in less than 50 seconds. For the remaining 11 instances, the execution time of the B&B method is smaller than that of the four-phase method for 2 instances (R.2 with $\epsilon_2 = 95$ and with $\epsilon_2 = 98$), and it is larger than that of the four-phase method for the other 9 instances. Overall, we observe that the total execution time of the four-phase method for all the instances in this table is 72.3% smaller than that of the B&B method.
3. For all instances, the execution time of both methods tends to be larger for larger values of ϵ_2 .
4. The impact of VIs in the context of the four-phase method is more significant for the instances with larger execution times.

Consequently, we observe that the impact of our proposed VIs in the context of the four-phase method for solving the model $IP-\epsilon_2$ is more significant for the GEO-10D instances than for the RAN instances. We also observe that the number of added BCP-VIs and BCC-VIs to the model in the context of the four-phase method for the RAN instances is significantly smaller than the corresponding numbers for the GEO-10D instances.

7.8 Concluding Remarks

Our computational experiments show that adding BCP-VIs, BCC-VIs, CD-VIs, and MC-VIs in the context of the four-phase method as described above can be quite effective in reducing the computational effort required to solve the corresponding models $IP-\epsilon_1$ and $IP-\epsilon_2$. The impact of adding these VIs across all instances is not the same. For some instances adding these VIs in the context of the four-phase method has little or no impact on reducing the overall computational requirements of the branch and bound method, while for other instances their impact could be quite significant.

Chapter 8

Inexact Methods for Solving the Model $IP-\epsilon_1$

As stated earlier in Chapter 6, at each iteration of the Algorithms 1 and 2, we need to construct and solve an instance of the integer programming models $IP-\epsilon_1$ and $IP-\epsilon_2$, respectively. We can solve these models via the four-phase method we discussed in the previous chapter. As reported, this method is not computationally viable when solving the models $IP-\epsilon_1$ and $IP-\epsilon_2$ with large values of ϵ_1 and ϵ_2 , respectively. For such instances, one option would be to employ heuristic methods that require less computational resources, albeit the resulting solution might not be optimal. In this chapter, we propose one such heuristic procedure for solving the model $IP-\epsilon_1$. We developed two heuristic methods in this context. One method is based on a local search procedure that we describe in Section 8.1. The other method is based on a Lagrangian relaxation of the model $IP-\epsilon_1$. This method is not effective for our collections of instances. We briefly mention this method in Section 8.2, and describe the details in Appendix E.

8.1 A Local Search Heuristic

In order to overcome the computational burden of solving hard instances of the model $IP-\epsilon_1$, we propose a heuristic algorithm that is much faster than the branch-and-bound method and the four-phase method, although the resulting solution might not be optimal. The pseudocode for this heuristic algorithm is shown in Figure 8.1. We briefly explain this algorithm here;

Solution Representation: We denote each solution by a vector (T_x, T_y) , where T_x is a conflict-free collection of nodes ($T_x \subseteq N$), and T_y is the collection of elements that are covered by the nodes in the collection T_x . Similarly, we denote (T_x^{best}, T_y^{best}) to the best solution obtained by the algorithm.

Initialization: We begin this algorithm (see line 1 of the pseudocode) with the initial

solution of $(T_x, T_y) = (\emptyset, \emptyset)$ which is always a feasible solution to any instances of the problem.

Iterative Process: At each iteration we either add or remove a node to the set T_x . To this end, we first identify the collection of nodes that can be added to T_x , i.e., the collection of nodes that do not have conflict with any node in T_x (see line 4 of the pseudocode). We denote this collection as $CAND$. We then make a decision to remove one node from T_x or add one new node to T_x . This decision depends on the collection $CAND$ and also the number of nodes in T_x .

If the collection $CAND$ is empty or the number of nodes in T_x is equal to the value of ϵ_1 , we consider removing one node from T_x . We have two policies for selecting the node to be removed from T_x . One policy is to remove the node that is added to T_x earlier than all other nodes in T_x . We use this policy to escape the local optima of the problem. Our other policy is to remove the node that brings the minimum value to the solution, i.e., the node whose removal from T_x results in a minimum decrease in the objective function value ($|T_y|$). If the number of iterations without improvement in the best solution obtained (T_x^{best}, T_y^{best}) is equal to the value of a pre-specified parameter Ω_r , we use the first policy to choose and remove a node from T_x (see lines 6-11 of the pseudocode). Otherwise, we use the second policy (see lines 13-17 of the pseudocode). After removing the selected node from T_x , we update T_y accordingly.

On the other hand, if the collection $CAND$ is not empty and the number of nodes in T_x is less than the value of ϵ_1 , we consider adding a node in $CAND$ to T_x (see lines 20-28 of the pseudocode). To this end, among all nodes in $CAND$, we select the node that adds the maximum value to the solution, that is the node which covers the maximum number of elements that are not currently in T_y . We add the selected node to T_x , and we update T_y accordingly. We also update the best solution (T_x^{best}, T_y^{best}) if the new T_y has larger number of elements than T_y^{best} .

In order to better prevent cycling, we maintain a tabu list in the algorithm. This tabu list stores a list of nodes (that are selected to be added or removed at each iteration) and these nodes reside in the tabu list for a specific number of iterations called the tabu tenure.

Termination: We terminate this procedure if both of the following conditions occur:

- The number of iterations is larger than a pre-specified parameter Ω_{iter} .
- The collection $CAND$ is empty.

8.1.1 Computational Experiments

In order to evaluate the effectiveness of the proposed heuristic procedure on an empirical basis, we use the collection of instances that we constructed in the previous chapter, and for each instance we solve the corresponding model $IP-\epsilon_1$ via the four-phase method (the exact method that we proposed in the previous chapter) as well as our proposed local search heuristic procedure. In this experiment, we turn on all heuristics and cut generation options of CPLEX, and

```

1 Initialization: Set  $q = 0$ ,  $(T_x, T_y) = (\emptyset, \emptyset)$ ,  $(T_x^{best}, T_y^{best}) = (\emptyset, \emptyset)$ , and  $CAND = \emptyset$ .

2 Iterative steps:
3 while  $q \leq \Omega_{iter}$  or  $CAND \neq \emptyset$  do
4   Identify the collection  $CAND = \{j : (j, k) \notin F, k \in T_x\}$ ;
5   if  $CAND = \emptyset$  or  $|T_x| = \epsilon_1$  then
6     if number of iterations without improvement in the objective value is  $\Omega_r$  then
7       Find the node  $e \in T_x$  which is added to the solution earlier than all other
         nodes in  $T_x$ ;
8       Remove node  $e$  from  $T_x$ , i.e.,  $T_x \leftarrow T_x \setminus \{e\}$ ;
9       Update  $T_y$ , i.e.,  $T_y = \bigcup_{j \in T_x} M_j$ ;
10      Update the tabu list;
11      Set  $q \leftarrow q + 1$ ;
12    else
13      Find node  $d \in T_x$  where  $d = \arg \min_{k \in T_x} |\bigcup_{j \in T_x \setminus \{k\}} M_j|$ ;
14      Remove node  $d$  from  $T_x$ , i.e.,  $T_x \leftarrow T_x \setminus \{d\}$ ;
15      Update  $T_y$ , i.e.,  $T_y = \bigcup_{j \in T_x} M_j$ ;
16      Update the tabu list;
17      Set  $q \leftarrow q + 1$ ;
18    end
19  else
20    Find node  $f \in N \setminus T_x$  where  $f = \arg \max_{j \in N \setminus T_x} |\bigcup_{j \in T_x \cup \{k\}} M_j|$ ;
21    Add node  $f$  to  $T_x$ , i.e.,  $T_x \leftarrow T_x \cup \{d\}$ ;
22    Update  $T_y$ , i.e.,  $T_y = \bigcup_{j \in T_x} M_j$ ;
23    if  $|T_y| > |T_y^{best}|$  then
24      Set  $T_x^{best} \leftarrow T_x$ ;
25      Set  $T_y^{best} \leftarrow T_y$ ;
26    end
27    Update the tabu list;
28    Set  $q \leftarrow q + 1$ ;
29  end
30 end

```

Figure 8.1: The pseudocode for local search heuristic procedure for solving the model $IP-\epsilon_1$

we impose a time limit of 7200 seconds on all instances when solving the corresponding model $IP-\epsilon_1$ via the four-phase method.

For the proposed heuristic procedure, we set $\Omega_{\text{iter}} = 5000$ and $\Omega_r = 50$. We also set the tabu tenure to the value $\lfloor \frac{\epsilon_1}{2} \rfloor$. We determined these values according to the results of a preliminary experiment that we performed in this context.

We evaluate the effectiveness of the heuristic approach by comparing its execution time with that of the exact method as well as by observing the corresponding value of the gap that it obtains at termination. In Tables 8.1 and 8.2, we report the results of solving the models $IP-\epsilon_1$ via the heuristic and the four-phase method for GEO-10D and RAN instances, respectively. In each table, in the third through last columns, we show the objective function value (z^*) of the model $IP-\epsilon_1$, the execution time of the four-phase method, the objective value of the best solution obtained by the heuristic method (z^{heu}), the optimality gap which is defined as $(z^* - z^{\text{heu}})/z^*$, and the execution time of the heuristic method. From Table 8.1, we observe that the solutions obtained by the heuristic method are quite close to the optimal solutions obtained by the exact method, while the execution time of the heuristic method is considerably smaller than the execution time of the exact method. More specifically, for 23 out of 24 instances that the exact method could solve in 2 hours, the average optimality gap of the solution obtained by the heuristic method is 1.6%, while the total execution time of applying the heuristic method to all these instances is 98.5% smaller than the total execution time of the exact method. We also observe that for 8 out of 24 instances in this table, the solution obtained by the heuristic method is the optimal solution of problem. The maximum optimality gap of the heuristic solution for all the instances in this table is 5.4%.

Similarly from Table 8.2, we observe that the average optimality gap of the solution obtained by the heuristic method is 0.7%, while the total execution time of applying the heuristic method to all the instances in this table is 98.7% smaller than the total execution time of the exact method. We also observe that for 16 out of 24 instances in this table, the solution obtained by the heuristic method is the optimal solution of problem. The maximum optimality gap of the heuristic solution for all the instances in this table is 3.8%.

8.2 A Lagrangian Heuristic

In addition to the local search heuristic we discussed so far, we developed a Lagrangian heuristic procedure for solving the model $IP-\epsilon_1$. However, our preliminary experiments indicated that the proposed procedure is not effective for solving the model $IP-\epsilon_1$ for instances in data sets GEO-10D and RAN. Hence, we do not discuss the procedure here. For detailed information on the proposed Lagrangian heuristic procedure, the reader is referred to Appendix E.

Table 8.1: The computational results of solving the $IP-\epsilon_1$ via the exact and inexact methods for GEO-10D instances

ID	ϵ_1	Four-Phase Method		Heuristic Method		
		z^*	Exec. time (sec)	z^{heu}	Optimality gap	Exec. time (sec)
G10.3	5	58	14.5	55	0.052	3.0
	9	78	20.3	78	0.000	2.3
	13	92	27.2	87	0.054	2.5
	17	98	470.1	96	0.020	3.2
G10.4	6	61	18.4	61	0.000	1.9
	11	80	76.5	77	0.038	1.8
	16	92	261.1	87	0.054	2.5
	20	97	4,684.5	95	0.021	3.4
G10.10	1	64	9.3	64	0.000	9.8
	3	90	13.2	88	0.022	2.4
	4	95	14.5	95	0.000	1.6
	5	98	66.3	98	0.000	1.4
G10.12	9	214	39.7	209	0.023	7.0
	15	256	237.9	249	0.027	7.3
	21	280	1,693.6	275	0.018	10.0
	27	-	-	289	-	14.7
G10.16	2	244	7.7	244	0.000	15.5
	4	280	9.3	276	0.014	6.1
	6	293	18.2	292	0.003	4.7
	8	299	81.9	299	0.000	4.6
G10.17	2	244	9.4	244	0.000	14.2
	4	276	19.2	272	0.014	5.0
	6	291	35.1	289	0.007	3.6
	8	299	148.6	298	0.003	3.6

Table 8.2: The computational results of solving the $IP-\epsilon_1$ via the exact and inexact methods for RAN instances

ID	ϵ_1	Four-Phase Method		Heuristic Method		
		z^*	Exec. time (sec)	z^{heu}	Optimality gap	Exec. time (sec)
R.2	5	49	1.1	49	0.000	1.7
	9	74	1.8	74	0.000	1.5
	12	86	8.4	86	0.000	1.6
	16	97	292.7	96	0.010	2.2
R.3	6	56	1.6	56	0.000	1.3
	10	77	6.0	77	0.000	1.2
	14	91	353.8	91	0.000	1.6
	18	99	235.8	97	0.020	2.3
R.4	3	52	0.9	50	0.038	2.0
	5	75	1.4	75	0.000	1.5
	7	87	8.5	87	0.000	1.2
	9	95	359.8	95	0.000	1.3
R.7	11	44	2.3	44	0.000	6.2
	19	67	3.0	67	0.000	6.5
	26	81	3.2	80	0.012	7.5
	34	95	5.1	92	0.032	10.1
R.8	13	49	8.5	49	0.000	4.5
	22	70	9.9	70	0.000	5.0
	31	86	41.3	84	0.023	7.1
	40	96	878.7	95	0.010	10.9
R.9	4	44	2.3	44	0.000	7.2
	7	68	6.6	68	0.000	5.1
	9	80	14.9	80	0.000	4.3
	12	94	5,214.6	93	0.011	4.0

Chapter 9

Conclusions and Future Research

In this dissertation, we study the set covering problem with conflict constraints in a single-objective context as well as in a bi-criteria context, and propose several pre-processing procedures and several families of valid inequalities for the corresponding IP models. Our specific contributions are as follows:

1. We introduce appropriate pre-processing techniques to remove redundancies, remove unnecessary variables, and tighten up the set of constraints.
2. We investigate and extend a family of valid inequalities that has been previously proposed for this model (i.e., covering-packing valid inequalities) and demonstrate their effectiveness. We show that when the constraint set of the model *SCC-IP* has a special structure, these VIs can be facet-defining for the convex hull of the feasible solutions for the IP model. We also demonstrate their effectiveness through a computational experiment.
3. We propose a new family of valid inequalities for the model *SCC-IP* that we refer to as the covering-clique valid inequalities (CC-VIs). We show that for *SCC-IP* models with a special structure, these VIs are also facet-defining, and we demonstrate their effectiveness through a computational experiment.
4. We propose an algorithm to find a subset of the maximal cliques of the conflict graph G so as to include every conflicting pair, and we present a method to add these maximal clique constraints to the model *SCC-IP*. We also propose column-domination valid inequalities and conflicting-set valid inequalities, and we show that by adding these VIs to the model *SCC-IP*, we can enhance the effectiveness of the maximal clique constraints.
5. We introduce the bi-criteria set-covering problem with conflict constraints, and propose an integer programming model to formulate this problem. We then introduce two strate-

gies for solving this problem, and present the effectiveness of these strategies through a numerical experiment.

6. We propose appropriate valid inequalities for the IP models discussed for the bi-criteria problem, and using these valid inequalities we propose a branch-and-cut algorithm for solving this problem. We demonstrate the effectiveness of the proposed branch-and-cut algorithm through a computational experiment.
7. We develop a local search heuristic approach to solve relatively large instances of the bi-criteria set-covering problem with conflict constraints, and demonstrate the efficiency of this algorithm through a numerical experiment.

9.1 Future research

We envision two immediate directions to extend the approaches outlined in this dissertation, as follows.

- In Chapter 4, we proposed Covering-Clique Valid Inequalities (CC-VIs) for the model *SCC-IP*, and in Chapter 5, we discussed a cutting plane method for identifying and adding violated CC-VIs to the model. However, our cutting plane method requires identifying all CC-VIs for a given instance, which takes a relatively long time and an excessively large amount of memory for a medium size instance. As a result, we were able to consider only those CC-VIs that are associated with cliques of size 2. One avenue for further research would be to devise an efficient separation algorithm that enables us to identify most violated CC-VIs without searching through all CC-VIs for a given instance of the problem.
- As we mentioned in Chapter 1, one application of the problem SCC could be in multi-period staff-scheduling problem. In this application, an instance of relatively medium size may have 200 tasks, 200 workers, 90 time periods (if we consider 8-hour periods in a month), which forms an instance of the problem SCC with $m = 200$ and $n = 200 \times 90 = 18000$. In Chapter 5, we observed that the B&B method and our proposed six-phase method take about two hours to solve some randomly generated instances with $m = 100$ and $n = 1000$. One can only expect that using the proposed six-phase method would require excessive time and memory for solving a large instance such as the above example. Consequently, the further step in this direction would be to devise a heuristic algorithm to solve larger instances of the problem within a reasonable execution time.

REFERENCES

- [1] J.H. Ablanedo-Rosas, Alidaee B. Gao, H., and W.Y. Teng. Allocation of emergency and recovery centres in Hidalgo, Mexico. *International Journal of Services Sciences*, 2(2):206–218, 2009.
- [2] A. Atamturk, G.L. Nemhauser, and M.W. Savelsbergh. Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121(1):40–55, 2000.
- [3] G. Avondo-Bodeno. *Economic Applications of the Theory of Graph*. Gordon and Breach, Science Publishers, New York, 1962.
- [4] A. Azadeh, G. Asadipour, H. Eivazy, and S. Nazari-Shirkouhi. A unique hybrid particle swarm optimisation algorithm for simulation and improvement of crew scheduling problem. *International Journal of Operational Research*, 13(4):406–422, 2012.
- [5] E. Balas and M.C. Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890, 1996.
- [6] Egon Balas. Disjunctive programming. *Annals of discrete mathematics*, 5:3–51, 1979.
- [7] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [8] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem 1. *Algorithmica*, 29(4):610–637, 2001.
- [9] J. Bautista and J. Pereira. Modeling the problem of locating collection areas for urban waste management. An application to the metropolitan area of Barcelona. *Omega*, 34(6):617–629, 2006.
- [10] J. Bautista and J. Pereira. A GRASP algorithm to solve the unicast set covering problem. *Computers & Operations Research*, 34(10):3162–3173, 2007.
- [11] J. E. Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93, 1987.
- [12] J.E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, pages 1069–1072, 1990.
- [13] J.E. Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European journal of operational research*, 94(2):392–404, 1996.
- [14] J.E. Beasley and K. Jonsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300, 1992.
- [15] P. Berman and A. Pelc. Distributed probabilistic fault diagnosis for multiprocessor systems. *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, pages 340–346, 1990.

- [16] J.E. Beslay. A lagrangian heuristic for the set covering problem. *Naval Research Logistics*, 37:151–164, 1990.
- [17] S. Busygin. A new trust region technique for the maximum weight clique problem. *Discrete Applied Mathematics*, 154(15):2080–2096, 2006.
- [18] J.F. Brub, M. Gendreau, and J.Y. Potvin. An exact ϵ -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, 194(1):39–50, 2009.
- [19] V. Cacchiani, V.C. Hemmelmayr, and F. Tricoire. A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, 163:53–64, 2014.
- [20] M. Campelo and R.C. Correa. A Lagrangian relaxation for the maximum stable set problem. *arXiv preprint arXiv:0903.1407*, 2009.
- [21] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. L. Guida. Algorithms for railway crew management. *Mathematical programming*, 79(1-3):125–141, 1997.
- [22] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1):353–371, 2000.
- [23] M. Caserta. Tabu search-based metaheuristic algorithm for large-scale set covering problems. *Metaheuristics*, pages 43–63, 2007.
- [24] B. Crawford, R. Soto, R. Cuesta, and F. Paredes. Application of the artificial bee colony algorithm for solving the set covering problem. *The Scientific World Journal*, 2014.
- [25] A. Darmann, U. Pferschy, and J. Schauer. Determining a Minimum Spanning Tree with Disjunctive Constraints. In: *Rossi, F., Tsoukias, A. (eds) Algorithmic Decision Theory, Lecture Notes in Computer Science, vol. 5783. Springer, Berlin Heidelberg*, pages 414–423, 2009.
- [26] I. Dukanovic and F. Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. *Mathematical Programming*, 109(2):345–365, 2007.
- [27] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.
- [28] L. Epstein and A. Levin. On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298, 2008.
- [29] X. Gandibleux. *Multiple criteria optimization: state of the art annotated bibliographic surveys (Vol. 52)*. Springer Science Business Media, 2006.
- [30] M.R. Garey and D.S. Johnson. *Computers and Intractability, A guide to the Theory of NP-Completeness. (A Series of Books in the Mathematical Sciences)*. W. H. Freeman and Company, 1979.

- [31] M. Gendreau, G. Laporte, and F. Semet. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31(3):347–358, 2004.
- [32] M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41(4):385–403, 1993.
- [33] A.M. Gerards and A. Schrijver. Matrices with the edmonds-Johnson property. *Combinatorica*, 6(4):365–379, 1986.
- [34] L.E. Gibbons, D.W. Hearn, P.M. Pardalos, and M.V. Ramana. Continuous characterizations of the maximum clique problem. *Mathematics of Operations Research*, 22(3):754–768, 1997.
- [35] J. Hasselberg, P.M. Pardalos, and G. Vairaktarakis. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization*, 3(4):463–482, 1993.
- [36] M. Hifi and M. Michrafy. A reactive local search-based algorithm for the disjunctively constrained knapsack problem. *Journal of the Operational Research Society*, 57(6):718–726, 2006.
- [37] M. Hifi and M. Michrafy. Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Computers & operations research*, 34(9):2657–2673, 2007.
- [38] H.H. Hoos and W. Pullan. Dynamic Local Search for the Maximum Clique Problem. *arXiv preprint arXiv:1109.5717*, 2011.
- [39] E. Housos and T. Elmroth. Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5):68–77, 1997.
- [40] K. Jansen and S. Ohring. Approximation algorithms for time constrained scheduling. *Parallel Algorithms for Irregularly Structured Problems*, pages 143–157, 1995.
- [41] K. Katayama, A. Hamamoto, and H. Narihisa. An effective local search for the maximum clique problem. *Information Processing Letters*, 95(5):503–511, 2005.
- [42] G.W. Kinney, J.W. Barnes, and B.W. Colletti. A reactive tabu search algorithm with variable clustering for the unicost set covering problem. *International Journal of Operational Research*, 2(2):156–172, 2007.
- [43] Y.H. Kuo and J.M. Leung. On the mixed set covering packing and partitioning polytope. *Discrete Optimization*, 22:162–182, 2016.
- [44] G. Lan, G.W. DePuy, and G.E. Whitehouse. An effective and simple heuristic for the set covering problem. *European journal of operational research*, 176(3):1387–1403, 2007.
- [45] F.J. MacWilliams and N.J.A. Sloane. *The theory of error-correcting codes*. Elsevier, 1977.
- [46] M. Mesquita and A. Paias. Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Computers & Operations Research*, 35(5):1562–1575, 2008.

- [47] W. Miller. Building multiple alignments from pairwise alignments. *Bioinformatics*, 9(2):169–176, 1993.
- [48] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.
- [49] J.L. Moore, M. Folkmann, A. Balmford, T. Brooks, N. Burgess, C. Rahbek, P.H. Williams, and J. Krarup. Heuristic and optimal solutions for set covering problems in conservation biology. *Ecography*, 26(5):595–601, 2003.
- [50] M.H. Mulati and A.A. Constantino. Ant-Line: a line-oriented ACO algorithm for the set covering problem. *Computer Science Society (SCCC), 2011 30th International Conference of the Chilean. IEEE*, 2011.
- [51] A.E.F. Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the bin packing problem with conflicts. *Inform Journal on computing*, 22(3):401–415, 2010.
- [52] Z. Naji-Azimi, P. Toth, and L. Galli. An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205(2):290–300, 2010.
- [53] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999.
- [54] B. Nepal, G. Lassan, B. Drow, and K. Chelst. A set-covering model for optimizing selection of portfolio of microcontrollers in an automotive supplier company. *European Journal of Operational Research*, 193(1):272–281, 2009.
- [55] P. Nobile and A. Sassano. A separation routine for the set covering polytope. *E. Balas, G. Cornuejols, R. Kannan, eds. Integer Programming and Combinatorial Optimization. In Proceedings of the 2nd IPCO Conference, Carnegie-Mellon University Press, Pittsburgh, PA.*, 1992.
- [56] T. Oncan, R. Zhang, and A.P. Punnen. The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research*, 40(4):920–930, 2013.
- [57] P.M. Pardalos and A.T. Phillips. A global optimization approach for solving the maximum clique problem. *International Journal of Computer Mathematics*, 33(3–4):209–216, 1990.
- [58] P.M. Pardalos and G.P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Computers & operations research*, 19(5):363–375, 1992.
- [59] U. Pferschy and J. Schauer. The Knapsack Problem with Conflict Graphs. *J. Graph Algorithms Appl.*, 13(2):233–249, 2009.
- [60] U. Pferschy and J. Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.
- [61] H.K. Rajagopalan, C. Saydam, and J. Xiao. A multiperiod set covering location model for dynamic redeployment of ambulances. *Computers & Operations Research*, 35(3):814–826, 2008.

- [62] S. Rebennack, G. Reinelt, and P.M. Pardalos. A tutorial on branch and cut algorithms for the maximum stable set problem. *International Transactions in Operational Research*, 19(1-2):161–199, 2012.
- [63] Z.G. Ren, Z.R. Feng, L.J. Ke, and Z.J. Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784, 2010.
- [64] D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.
- [65] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2012.
- [66] P. Samer and S. Urrutia. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, 9(1):41–55, 2015.
- [67] A. Singh and A.K. Gupta. A hybrid heuristic for the maximum clique problem. *Journal of Heuristics*, 12(1):5–22, 2006.
- [68] N.J.A. Sloane. Unsolved problems in graph theory arising from the study of codes. *Graph Theory Notes of New York*, 18:11–20, 1989.
- [69] S. Sundar and A. Singh. A hybrid heuristic for the set covering problem. *Operational Research*, 12(3):345–365, 2012.
- [70] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.
- [71] G.K. Tutunchi and Y. Fathi. Effective methods for solving the bi-criteria p-center and p-dispersion problem. *Computers & Operations Research*, 101:43–54, 2019.
- [72] S. Umetani and M. Yagiura. Relaxation heuristics for the set covering problem. 2007.
- [73] F.J. Vasko, D.D. Newhart, K.L. Stott Jr, and F.E. Wolf. A large-scale application of the partial coverage uncapacitated facility location problem. *Journal of the Operational Research Society*, 54(1):11–20, 2003.
- [74] F.J. Vasko and G.R. Wilson. Using a facility location algorithm to solve large set covering problems. *Operations Research Letters*, 3(2):85–90, 1984.
- [75] Y. Wang, D. Ouyang, L. Zhang, and M. Yin. A novel local search for unicost set covering problem using hyperedge configuration checking and weight diversity. *Science China Information Sciences*, 60(6):062103, 2017.
- [76] D. Warrier, W.E. Wilhelm, J.S. Warren, and I.V. Hicks. A branch-and-price approach for the maximum weight independent set problem. *Networks*, 46(4):198–209, 2005.

- [77] Q. Wu and J.K. Hao. A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709, 2015.
- [78] M. Yagiura, M. Kishida, and T. Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172(2):472–499, 2006.
- [79] T. Yamada, S. Kataoka, and K. Watanabe. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43(9), 2002.
- [80] R. Zhang, S.N. Kabadi, and A.P. Punnen. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191–205, 2011.

APPENDICES

Appendix A

Clarifying The Maximal Clique Finder Algorithm

To illustrate how the Maximal Clique Finder algorithm (MCF) works, we here give a simple numeric example in the context of a specific instance the SCC.

Example 12. Consider the conflict graph G shown in Figure A.1a. Now, we employ our iterative algorithm to find maximal cliques of the conflict graph G , as follows.

Figure A.1: The conflict graph G of Example 12

Initialization Step: $Q = \emptyset$.

Iterative Step: Iteration 1;

- We define the set q^1 as $q^1 = \emptyset$.

- We then search for the node with maximum updated degree. Here, nodes v_2 and v_3 with $d'(v_2) = d'(v_3) = 3$ are both the nodes with maximum updated degree. Since there is a tie, we choose the vertex with smaller index, v_2 . So the node v_2 is selected to be in the current clique, i.e., $q^1 = \{v_2\}$.
- We find the set of the neighbors of each node in q^1 . The node v_2 is the only element of q^1 , and the set of neighbors of v_2 is $N(v_2) = \{v_1, v_3, v_4\}$.
- We search for the node with maximum updated degree among neighbors of v_2 . The node v_3 has the maximum updated degree in the set $N(v_2)$ with $d'(v_3) = 3$. Thus, v_3 is selected to be in the current clique, i.e., $q^1 = \{v_2, v_3\}$.
- We search for the set of nodes that are neighbors with both v_2 and v_3 , i.e., $N(v_2) \cap N(v_3) = \{v_1, v_3, v_4\} \cap \{v_1, v_2, v_4\} = \{v_1, v_4\}$.
- We search for the node with maximum updated degree in the set $N(v_2) \cap N(v_3)$. Nodes v_1 and v_4 have the same updated degrees, $d'(v_1) = d'(v_4) = 2$. So, we choose the vertex with smaller index, v_1 , to be in the current clique, i.e., $q^1 = \{v_2, v_3, v_1\}$.
- We search for the set of nodes that are neighbors with v_2 , v_3 , and v_1 , i.e., $N(v_2) \cap N(v_3) \cap N(v_1) = \emptyset$. Hence, no other node can be added to the current clique, and we go to the step 2.1 of the algorithm.
- $Q = \{q^1\}$, $E' = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$, and the graph is updated as shown in Figure A.1b, in which all edge that are in E' are shown by dashed lines, and all other edges are shown by solid lines. Now the updated degree for each node is $d'(v_1) = 0$, $d'(v_2) = d'(v_3) = 1$, $d'(v_4) = 2$.

Iterative Step: Iteration 2;

- We define the second clique as $q^2 = \emptyset$.
- We then search for the node with maximum updated degree. Hence, we select node v_4 to be in the current clique, i.e., $q(2) = \{v_4\}$.
- We find the set of the neighbors of each node in $q(2)$. The node v_4 is the only element of q^2 , and the set of neighbors of v_4 is $N(v_4) = \{v_2, v_3\}$.
- We search for the node with maximum updated degree among neighbors of v_4 , i.e., $N(v_4) = \{v_2, v_3\}$. Nodes v_2 and v_3 have the same updated degrees, $d'(v_2) = d'(v_3) = 1$. So, we choose the vertex with smaller index, v_2 , to be in the current clique, i.e., $q^2 = \{v_4, v_2\}$.

- We search for the set of nodes that are neighbors with both v_4 and v_2 , i.e., $N(v_4) \cap N(v_2) = \{v_3\}$.
- The node v_3 is selected to be in the current clique, i.e., $q^2 = \{v_4, v_2, v_3\}$.
- We search for the set of nodes that are neighbors with v_4 , v_2 , and v_3 , i.e., $N(v_4) \cap N(v_2) \cap N(v_3) = \emptyset$. Hence, no other node can be added to the current clique, and we go to the step 5 of the algorithm.
- $Q = \{q^1, q^2\}$, $E' = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_3, v_4)\}$, and the graph is updated as shown in Figure A.1c. Since $E' = E$, we terminate the algorithm and return a collection of two maximal cliques as $Q = \{q^1, q^2\}$ where $q^1 = \{v_2, v_3, v_1\}$ and $q^2 = \{v_4, v_2, v_3\}$.

Appendix B

Impact of CP-VIs in Presence of CPLEX Cuts

As mentioned in Section 5.1.1.3 of Chapter 5, in order to observe the impact of CP-VIs in the presence of CPLEX cuts, we perform a computational experiment in which we solve the model *SCC-IP* for a selected group of instances in GEO-10D, RAN, and OR-L data sets as described in Chapter 3 as well as all instances in data set A that we constructed in Section 5.1.1.1. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the model *SCC-IP* for these instances, considering the following four different scenarios:

- **Scenario 1:** We solve the model via B&B method (i.e., without adding CP-VIs) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 2:** We solve the model via B&B method (i.e., without adding CP-VIs) where we turn on all cut generation options of CPLEX.
- **Scenario 3:** We solve the model with the two-phase-CPVI method (i.e., with CP-VIs added to the model). We turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 4:** We solve the model with the two-phase-CPVI method (i.e., with CP-VIs added to the model). We turn on all cut generation options of CPLEX.

We solve each of the instances with each of the above mentioned scenarios and in Tables B.1 and B.2 we report the results of this experiment, including the execution times along with total number of CPLEX cuts added to the B&B tree in Scenarios 2 and 4 as well as total number of CP-VIs added to the model in Scenarios 3 and 4.

Table B.1: Results of adding CP-VIs to the model *SCC-IP* for GEO-10D, RAN, and OR-L instances in the presence of CPLEX cuts

	Scenario 1	Scenario 2		Scenario 3			Scenario 4			
ID	Total time (sec)	# of CPLEX cuts	Total time (sec)	# of CP-VI	IP time (sec)	Total time (sec)	# of CP-VI	# of CPLEX cuts	IP time (sec)	Total time (sec)
G10.4	2,230.0	157	3,087.3	977	2,178.7	2,184.8	977	85	1,780.1	1,786.2
G10.5	1.7	0	2.3	1	1.2	1.9	1	0	1.8	2.5
G10.6	5.6	3	5.5	39	7.4	8.6	39	0	10.0	11.2
G10.7	85.6	11	72.3	156	49.4	51.9	156	1	144.2	147.1
G10.8	714.6	43	1,369.8	510	196.4	199.8	510	6	1,656.8	1,660.3
G10.10	19.7	9	65.0	283	11.3	14.6	283	3	67.3	70.3
G10.13	31.6	1	36.6	3	31.8	37.8	3	1	37.0	41.2
G10.14	606.1	6	790.5	92	569.2	574.9	92	0	644.6	649.8
G10.15	-	-	-	430	-	-	430	-	-	-
R.3	362.4	6	228.5	40	211.7	212.1	40	3	275.5	276.0
R.7	0.7	8	0.9	4	0.4	0.8	4	5	1.2	1.5
R.8	500.3	60	702.4	74	460.9	462.3	74	16	378.4	379.9
R.9	424.6	0	26.8	0	424.6	425.1	0	0	26.8	27.3
L.3	81.0	4	69.1	18	67.0	69.5	18	0	71.8	74.9
L.4	376.3	6	357.4	28	329.8	333.3	28	1	305.5	309.1
L.5	1,035.7	10	994.3	48	1,077.1	1,081.6	48	5	942.2	947.4

Table B.2: Results of adding CP-VIs to the model *SCC-IP* for instances in data set A in the presence of CPLEX cuts

	Scenario 1	Scenario 2		Scenario 3			Scenario 4			
ID	Total time (sec)	# of CPLEX cuts	Total time (sec)	# of CP-VI	IP time (sec)	Total time (sec)	# of CP-VI	# of CPLEX cuts	IP time (sec)	Total time (sec)
CP.1	86.4	157	10.0	102	2.2	11.6	102	28	2.4	11.6
CP.2	242.8	170	27.4	111	9.0	17.6	111	49	14.7	23.8
CP.3	660.5	143	62.8	116	13.2	23.0	116	62	22.7	31.7
CP.4	470.2	160	43.4	117	16.7	31.6	117	70	21.6	36.5
CP.5	1,226.8	189	121.5	127	56.1	71.1	127	72	85.8	100.7
CP.6	3,236.2	191	885.0	128	247.5	263.0	128	83	237.0	253.0
CP.7	66.8	85	12.9	94	7.5	13.0	94	12	7.5	13.3
CP.8	362.8	93	45.7	103	18.1	27.2	103	31	19.5	28.8
CP.9	1,416.8	127	139.8	109	34.7	41.4	109	44	47.1	54.0
CP.10	3,873.3	135	92.1	113	59.3	67.8	113	26	25.2	33.6
CP.11	-	103	2,721.9	109	606.2	615.4	109	32	546.0	555.0
CP.12	-	-	-	120	5,362.3	5,373.9	120	47	4,908.9	4,920.0
CP.13	75.2	64	45.3	83	35.0	42.4	83	14	33.8	41.0
CP.14	494.0	75	285.1	85	55.6	62.1	85	9	118.3	125.1
CP.15	649.0	56	149.4	82	151.9	158.6	82	27	165.1	171.9
CP.16	-	71	2,491.3	78	644.5	651.5	78	28	1,105.8	1,113.6
CP.17	-	-	-	81	2,460.8	2,468.6	81	29	3,138.3	3,145.8
CP.18	-	-	-	88	2,023.2	2,031.3	88	23	1,902.8	1,910.6

Appendix C

Impact of CC-VIs in Presence of CPLEX Cuts

As mentioned in Section 5.2.1.3 of Chapter 5, in order to observe the impact of CC-VIs in the presence of CPLEX cuts, we perform a computational experiment in which we solve the model *SCC-IP* for a selected group of instances in GEO-10D, RAN, and OR-L data sets as described in Chapter 3 as well as all instances in data set B that we constructed in Section 5.2.1.1. To this end, we first impose pre-processing procedures (explained in Chapter 3) on each instance, and then we solve the model *SCC-IP* for these instances, considering the following four different scenarios:

- **Scenario 1:** We solve the model via B&B method (i.e., without adding CC-VIs) where we turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 2:** We solve the model via B&B method (i.e., without adding CC-VIs) where we turn on all cut generation options of CPLEX.
- **Scenario 3:** We solve the model with the two-phase-CCVI method (i.e., with CC-VIs added to the model). We turn off all pre-processing, heuristics and cut generation options of CPLEX.
- **Scenario 4:** We solve the model with the two-phase-CCVI method (i.e., with CC-VIs added to the model). We turn on all cut generation options of CPLEX.

We solve each of the instances with each of the above mentioned scenarios and in Tables C.1 and C.2 we report the results of this experiment, including the execution times along with total number of CPLEX cuts added to the B&B tree in Scenarios 2 and 4 as well as total number of CC-VIs added to the model in Scenarios 3 and 4.

Table C.1: Results of adding CC-VIs to the model *SCC-IP* for GEO-10D, RAN, and OR-L instances in the presence of CPLEX cuts

	Scenario 1	Scenario 2		Scenario 3			Scenario 4			
ID	Total time (sec)	# of CPLEX cuts	Total time (sec)	# of CC-VI	IP time (sec)	Total time (sec)	# of CC-VI	# of CPLEX cuts	IP time (sec)	Total time (sec)
G10.1	1.3	3	0.7	17	1.4	1.8	17	4	0.7	1.2
G10.2	17.7	19	40.3	123	11.7	13.2	123	14	19.0	20.5
G10.3	277.4	75	448.5	199	201.9	205.8	199	68	496.3	500.3
G10.4	2,230.0	157	3,087.3	228	1,897.9	1,902.8	228	153	2,866.1	2,871.2
G10.6	5.6	3	5.5	55	11.7	13.2	55	1	11.9	13.4
G10.9	3.8	4	2.0	37	4.8	8.3	37	0	1.6	4.9
G10.14	606.1	6	790.5	107	421.3	445.3	107	5	109.4	131.8
G10.16	81.5	0	80.2	17	49.3	52.8	17	1	84.8	88.0
G10.17	205.7	3	149.5	37	159.9	169.9	37	3	170.4	180.4
R.1	2.3	1	3.1	12	2.5	2.6	12	3	3.8	3.9
R.2	32.5	3	22.7	45	23.7	23.8	45	2	18.0	18.2
R.3	362.4	6	228.5	70	233.9	234.6	70	2	279.0	279.8
R.10	3,449.8	0	3,219.3	3	3,431.6	3,432.1	3	0	3,363.3	3,363.9
L.1	147.3	10	207.8	27	181.0	181.8	27	12	227.3	228.1
L.2	16.0	2	13.2	21	12.8	13.4	21	2	14.4	15.0
L.4	376.3	6	357.4	51	304.9	306.3	51	4	358.2	359.6
L.5	1,035.7	10	994.3	59	1,078.9	1,080.6	59	4	1,050.6	1,052.3

Table C.2: Results of adding CC-VIs to the model *SCC-IP* for instances in data set B in the presence of CPLEX cuts

	Scenario 1	Scenario 2		Scenario 3			Scenario 4			
ID	Total time (sec)	# of CPLEX cuts	Total time (sec)	# of CC-VI	IP time (sec)	Total time (sec)	# of CC-VI	# of CPLEX cuts	IP time (sec)	Total time (sec)
CC.1	4,764.7	174	1.0	39	8.7	9.0	39	117	0.9	1.2
CC.2	-	170	1.0	40	21.4	21.7	40	131	0.8	1.0
CC.3	-	181	1.5	45	124.0	124.2	45	131	1.0	1.3
CC.4	-	246	2.5	36	355.0	355.4	36	167	2.5	2.9
CC.5	-	246	3.7	53	975.4	975.9	53	220	3.0	3.5
CC.6	-	206	8.4	-	-	-	57	181	5.9	6.4
CC.7	12.5	99	1.2	38	1.2	1.5	38	115	1.1	1.5
CC.8	229.5	161	11.5	50	9.2	9.5	50	108	8.5	8.7
CC.9	585.0	168	11.2	60	23.6	24.1	60	122	8.8	9.3
CC.10	3,108.6	213	21.5	72	402.3	403.2	72	182	16.6	17.5
CC.11	3,270.9	243	89.5	93	1,471.0	1,472.3	93	241	47.7	48.9
CC.12	-	237	94.2	89	6,736.2	6,737.3	89	277	97.8	98.9
CC.13	-	285	120.4	-	-	-	89	226	127.8	128.9
CC.14	47.9	99	8.6	66	12.2	12.9	66	117	10.2	10.9
CC.15	169.2	116	18.9	63	25.7	26.2	63	131	21.8	22.3
CC.16	329.4	129	57.9	55	78.5	79.3	55	108	32.7	33.5
CC.17	756.5	162	34.0	60	124.5	125.1	60	119	24.8	25.4
CC.18	-	159	878.7	79	3,260.0	3,261.1	79	146	819.0	820.2
CC.19	-	174	1,037.7	63	6,103.4	6,104.0	63	152	1,013.4	1,013.9
CC.20	-	162	2,718.1	-	-	-	70	194	514.5	515.5

Appendix D

Data for Instance Explained in Section 6.1.2

In Table D.1, we present the data for the set covering matrix A . Since such a large matrix cannot be fit into the page, we present only the indexes of the columns that cover each row. Similarly in Table D.2, we present the conflict collection F by providing the indexes of all columns that are in conflict with each column.

Table D.1: Set covering matrix data for instance explained in Section 6.1.2

Row index	Indexes of columns covering this row	Row index	Indexes of columns covering this row
1	11, 34, 62, 88, 94	26	21, 22, 38, 59, 61, 87
2	25, 26, 49, 83	27	10, 18, 51, 84, 91
3	14, 40, 61, 65, 76, 95	28	7
4	28, 42, 52, 53	29	7, 12, 13, 14, 16, 19, 27, 76, 98
5	13, 53	30	42, 44, 62, 85, 94
6	24, 46, 57, 70, 85, 86, 100	31	9, 13, 26, 49, 67, 96
7	21, 49, 55, 86	32	2, 43, 52, 61
8	77, 96	33	23, 39, 42, 52, 55, 56, 59, 60, 88, 94
9	5, 20, 22, 25	34	6, 31, 35, 50, 53, 55, 68, 91
10	49, 64, 82, 88, 91, 94, 100	35	18, 30, 33, 45, 51, 69, 76, 80, 85, 92, 94
11	3, 21, 23, 37, 47, 48, 55, 97	36	11, 42, 44, 59, 73
12	5, 17, 57, 99	37	10, 21, 32, 41, 42, 51, 70, 76
13	96	38	57
14	55, 83, 95, 97	39	27, 34, 47, 59, 67, 72, 75, 99
15	6, 24, 33, 37, 64, 76	40	38, 40, 71, 81, 94
16	57, 63, 69	41	6, 15, 33, 49, 71, 88, 99
17	2, 14, 25, 50, 62, 69, 77	42	1, 44, 45, 74
18	6, 38, 98	43	3, 18, 25, 48, 64, 73, 89
19	1, 14, 26, 27, 58, 83, 84, 98, 99	44	30, 45, 52, 96
20	12, 21, 45, 94	45	29, 42, 50, 53, 55, 74, 84
21	13, 18, 31, 35, 41	46	6, 8, 15, 33, 53, 71, 90
22	9, 22, 47, 66, 70, 83, 94	47	11, 33, 37, 38, 63, 64
23	16, 20, 26, 54, 78, 80	48	9, 75, 85
24	18, 43, 46, 47, 76	49	5, 10, 18, 50, 55, 91, 97
25	18, 22, 61, 91	50	4, 19, 28, 67

Table D.2: Conflict relations data for instance explained in Section 6.1.2

Col. index	Indexes of columns in conflict with this column	Col. index	Indexes of columns in conflict with this column
1	12, 35, 63, 89, 95	26	12, 34, 45, 67, 71, 77
2	28, 29, 52, 86	27	32, 56, 58, 96
3	20, 46, 67, 71, 82	28	2, 5, 7, 18, 49, 52, 53, 82
4	5, 38, 52, 62, 63	29	2, 17, 21, 53, 57, 96
5	4, 28, 68	30	11, 56, 86, 87
6	45, 67, 78, 91	31	16, 24, 34, 55, 57, 83
7	13, 14, 28, 49, 77, 83	32	21, 27, 38, 46, 79
8	22	33	12, 16, 45, 52, 68
9	22, 41, 50, 65, 67, 70	34	18, 25, 26, 31
10		35	1, 24, 37, 42, 43, 44, 46, 49, 57
11	15, 30, 48, 54, 57, 60, 66, 69, 87, 89	36	22, 42, 64
12	1, 15, 25, 26, 33, 75, 83, 95	37	22, 35, 45, 47, 65, 88, 97
13	7, 48, 90	38	4, 32, 50, 54, 67, 90
14	7	39	47, 76, 82
15	11, 12, 16, 75	40	18, 19, 63, 72, 81
16	15, 19, 31, 33, 42, 60, 69, 73, 100	41	9, 24, 84, 100
17	29	42	16, 35, 36, 45, 55, 58, 59, 62, 63, 91, 97
18	28, 34, 40, 73, 85, 96	43	35, 52, 77, 81, 96, 99
19	16, 40, 52, 59, 67, 96	44	35, 45, 58, 81
20	3, 48	45	6, 21, 26, 33, 37, 42, 44, 53, 65, 68, 80, 86
21	29, 32, 45, 57, 58, 89	46	3, 32, 35, 50, 57, 61, 66, 73, 75, 92
22	8, 9, 36, 37, 51, 52, 65, 74, 98	47	25, 37, 39, 70, 72, 87
23	70, 89, 94	48	11, 13, 20, 49, 86, 97
24	31, 35, 41	49	7, 28, 35, 48, 57, 66, 67, 76, 95
25	12, 34, 47, 72, 91, 95	50	9, 38, 46, 51

Table D.2: Continued

Col. index	Indexes of columns in conflict with this column	Col. index	Indexes of columns in conflict with this column
51	22, 50, 83	76	39, 49, 89, 90
52	2, 4, 19, 22, 28, 33, 43	77	7, 26, 43, 56
53	28, 29, 45, 58, 65, 78, 90, 98	78	6, 53, 55, 62, 90
54	11, 38, 57, 60, 84	79	32
55	31, 42, 78, 80	80	45, 55, 63
56	27, 30, 67, 77, 90	81	40, 43, 44, 72, 96
57	11, 21, 29, 31, 35, 46, 49, 54, 59, 68, 86	82	3, 28, 39, 58, 88
58	21, 27, 42, 44, 53, 60, 82, 99	83	7, 12, 31, 51, 75, 91, 96
59	19, 42, 57, 69, 71	84	41, 54, 88
60	11, 16, 54, 58, 74, 75	85	18
61	46, 65, 94	86	2, 30, 45, 48, 57, 74, 91, 96
62	4, 42, 71, 78	87	11, 30, 47, 75
63	1, 4, 40, 42, 64, 80, 89	88	37, 68, 75, 82, 84
64	36, 63, 69	89	1, 11, 21, 23, 63, 69, 71, 76, 96
65	9, 22, 37, 45, 53, 61, 75, 90, 97	90	13, 38, 53, 56, 65, 76, 78, 92, 99
66	11, 46, 49	91	6, 25, 42, 70, 73, 83, 86
67	3, 6, 9, 19, 26, 38, 49, 56, 74	92	46, 90, 97
68	5, 33, 45, 57, 75, 88, 96, 99	93	70, 99
69	11, 16, 59, 64, 70, 89, 99	94	23, 61
70	9, 23, 47, 69, 91, 93, 100	95	1, 12, 25, 49
71	3, 26, 59, 62, 89	96	18, 19, 27, 29, 43, 68, 81, 83, 86, 89
72	25, 40, 47, 81, 99	97	37, 42, 48, 65, 92
73	16, 18, 46, 91	98	22, 53
74	22, 60, 67, 86	99	43, 58, 68, 69, 72, 90, 93
75	12, 15, 46, 60, 65, 68, 83, 87, 88	100	16, 41, 70

Appendix E

A Lagrangian Heuristic Procedure for Solving the Model $IP-\epsilon_1$

Here, we discuss the details of a Lagrangian heuristic procedure we proposed for solving the model $IP-\epsilon_1$. We refer to the proposed procedure as Lagrangian heuristic. Our proposed procedure has two major components:

1. **Lagrangian relaxation:** Obtaining an upper bound on the objective value of the model $IP-\epsilon_1$ by Lagrangian relaxation procedure.
2. **Heuristic procedure:** Applying a heuristic procedure to obtain an improved feasible solution and lower bound on the objective of the model $IP-\epsilon_1$.

We terminate this approach when the relative difference between the best upper bound and the best lower bound is below a certain threshold. Following is a description of each component of this heuristic approach.

1. Lagrangian Relaxation

We relax the first group of constraints from the model $IP-\epsilon_1$ and introduce a non-negative Lagrange multiplier λ_i associated with the i^{th} constraint in this group and write the corresponding

Lagrangian relaxation as follows.

$$L(\boldsymbol{\lambda}) = \text{maximize} \quad \sum_{i=1}^m y_i + \sum_{i=1}^m \lambda_i \left(\sum_{j=1}^n a_{ij} x_j - y_i \right) \quad (LR-IP-\epsilon_1)$$

$$\text{subject to} \quad x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F \quad (\text{E.1a})$$

$$\sum_{j=1}^n x_j \leq \epsilon_1 \quad (\text{E.1b})$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n \quad (\text{E.1c})$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \dots, m \quad (\text{E.1d})$$

Rearranging the terms in the objective function, we obtain

$$L(\boldsymbol{\lambda}) = \text{maximize} \quad \sum_{j=1}^n \left(\sum_{i=1}^m \lambda_i a_{ij} \right) x_j + \sum_{i=1}^m (1 - \lambda_i) y_i$$

$$\text{subject to} \quad x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F$$

$$\sum_{j=1}^n x_j \leq \epsilon_1$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \dots, m$$

Since the first group of constraints from the model $IP-\epsilon_1$ is no longer present in the above (relaxed) model, it follows that for each i we can determine the optimal value of y_i for each given value of λ_i , i.e., $y_i = 1$ if $\lambda_i < 1$, and $y_i = 0$ otherwise. It follows that we can substitute the term containing y_i in the objective function with a corresponding term containing λ_i .

$$L(\boldsymbol{\lambda}) = \text{maximize} \quad \sum_{j=1}^n \left(\sum_{i=1}^m \lambda_i a_{ij} \right) x_j + \max(1 - \lambda_i, 0)$$

$$\text{subject to} \quad x_j + x_k \leq 1 \quad \forall j, k = 1, \dots, n : (j, k) \in F$$

$$\sum_{j=1}^n x_j \leq \epsilon_1$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n$$

For any value of the coefficients λ_i , the corresponding value of $L(\boldsymbol{\lambda})$ provides an upper bound for the optimal value of the model $IP-\epsilon_1$. To obtain the lowest possible upper bound in this context, we employ a subgradient search procedure, i.e., at each iteration this procedure, we

modify the value of λ and solve the model $LR-IP-\epsilon_1$ (see [53]). Note that the model $LR-IP-\epsilon_1$ is equivalent to the problem of finding an independent (pairwise non-adjacent) set of nodes with size smaller than ϵ_1 on a graph $G = (V, E)$ where the set of vertices is $V = \{v_j : j = 1, \dots, n\}$, i.e., each vertex v_j corresponds uniquely to the column j of matrix A (for all $j = 1, \dots, n$), and E corresponds to the set of edges of G so that for each two columns j and k ($j, k = 1, \dots, n$), $(v_j, v_k) \in E$ when there is a conflict between columns j and k , i.e., $(j, k) \in F$. Hence, we can use the maximal clique valid inequalities in the context of a two-phase method similar to the two-phase-MCVI method (introduced in Chapter 5) to reduce the execution time of solving the model $LR-IP-\epsilon_1$.

2. Heuristic Procedure

In order to construct a collection of potentially good feasible solutions for the model $IP-\epsilon_1$, at the end of each iteration of the subgradient search, for each $i \in M$, we let $y_i = 1$ if there exists a column j with $x_j^* = 1$ and $a_{ij} = 1$, where $x^* = [x_1^*, x_2^*, \dots, x_n^*]^T$ represents the optimal values of the variables x_j in the model $LR-IP-\epsilon_1$. Then, the corresponding value of the objective function, i.e., $\sum_{i=1}^m y_i$, is a lower bound for the optimal value of the model $IP-\epsilon_1$.

Results of a Preliminary Experiment

We performed a preliminary experiment in which we applied the proposed Lagrangian heuristic method to the model $IP-\epsilon_1$ for the instances that we constructed in Chapter 7. In our experiment, we ran the algorithm for a reasonably large number of iterations for all instances. The results of our experiment indicated that the optimality gap of the solution obtained by the proposed Lagrangian heuristic is large for most of the instances. As a result, we conclude that the proposed method is not effective for solving the model $IP-\epsilon_1$ for instances in data sets GEO-10D and RAN.