2-1-2013

A Computational study of branching rules for multi-commodity fixed-charged network flow problems

Sheetal Murkute

Thesis

A Computational Study of Branching Rules for Multi-Commodity Fixed-Charged Network Flow Problems

Sheetal Murkute

February, 2013

Department of Industrial and Systems Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology

Thesis Advisor

Dr. Michael Hewitt

Thesis Committee

Dr. Michael E. Kuhl

Dr. Scott Grasman

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering

Thesis Advisor:

Dr. Michael Hewitt
Assistant Professor
Department of Industrial and Systems Engineering, RIT

Committee Members:

Dr. Michael E. Kuhl
Professor
Department of Industrial and Systems Engineering, RIT

Dr. Scott Grasman
Department Head and Professor
Department of Industrial and Systems Engineering, RIT

Approved By:

Dr. Michael Hewitt _____ Date _____

Dr. Michael E. Kuhl _____ Date _____

Dr. Scott Grasman _____ Date _____

Abstract

Branch and bound based algorithms are used by many commercial mixed integer programming solvers for solving complex optimization problems. In a branch and bound based method, a feasible region is divided into smaller sub-problems. This is called branching and various branching strategies have been developed to improve the performance of branch and bound based algorithms. However, their performance has primarily been studied on general mixed integer programs. Thus, in the first phase of this thesis, we study the performance of these branching strategies on a specific, structured mixed integer program, the capacitated multi-commodity fixed charge network flow (MCFCNF) problem. We also develop new branching strategies using the pool of available feasible solutions for solving the mixed integer program for MCFCNF. We present the computational results for testing various branching rules with four different variants of the network design problem studied with SCIP and GLPK mathematical solvers.

Keywords: Mixed-integer-programming; Branch-and-bound; Variable selection methods; Entropic branching; Pseudo-cost branching; Strong branching; Reliability branching; Solution based approach

Table of Contents

Abstract 3
List of Tables 7
List of Figures 8
1. Introduction 9
1.1 Background and Motivation 9
1.2 Problem Statement 11
2. Branch and Bound Based Algorithm 12
2.1 Branch and Bound Search Tree 12
2.2 Branching 13
2.3 Algorithm for Branch and Bound Search Method 15
2.4 Methods to Improve the Performance of Branch and Bound Based Algorithms 17
3. Overview of the Variable Selection Methods 23
3.1 Literature Review 23
3.2 Classification of Variable Selection Methods 25
3.2.1 Methods Based on the Current LP Relaxation 26
3.2.2 Methods Based on the Current Statistics Calculations 29
3.2.3 Methods Considering the LP Relaxation of Child Nodes 31
3.2.4 Hybrid Methods 33
4. Multi-Commodity Fixed Charged Network Flow (MCFCNF) Problem 36
4.1 Problem Description 36
4.2 A Service Network Design Application – Freight Transportation 37
4.3 Literature Review 40
4.4 Mathematical Formulation 41
4.5 Variants of MCFCNF Problem 43
4.5.1 Binary Arc Variables, Binary Flow Variables: 43
4.5.2 Integer Arc Variables, Fractional Flow Variables: 44
4.5.3 Integer Arc Variables, Binary Flow Variables: 45
4.5.4 Naming Conventions Used for Representing the Variants 45
5. Computational Results for the Existing Variable Selection Methods 46
5.1 Test Sets 46

5.2 Experimental Setup 48
5.3 Computational Results 50
5.3.1 Description of Tables 50
6. Solution Based Approach for Variable Selection 56
6.1 Motivation 56
6.2 Solution Based Branching Rule 57
6.3 Algorithm for Selecting a Branching Variable in the Solution Based Branching Rule 60
6.4 Hybrid Method for the Variants with Binary Flow Variables 61
6.5 Algorithm for the Solution Based Approach 61
7. Computational Results for the Solution Based Approach 62
7.1 Experimental Setup for Collecting the Pool of Feasible Solutions with SCIP 63
7.2 Results for the Solution Based Approach for Variable Selection 64
7.2.1 Observations: 66
7.3 Results for Studying the Effect of the Quality and Quantity of Feasible Solutions 68
8. Conclusions and Future Research 71
Acknowledgements 72
References 73
APPENDIX I 76
RESULTS WITH SCIP: Branching Rules 76
Part A: Variant_B_C 76
Part B: Variant_B_B 77
Part C: Variant_I_C 79
Part C: Variant_I_B 81
APPENDIX II 83
RESULTS WITH GLPK 83
Experimental Setup 83
Results for the Existing Branching Rules 84
Part A: With Disaggregation Constraints and with the pseudo-cost based node selection strategy: 87
Part B: With Disaggregation Constraints and with the default (best bound) node selection strategy: 88

APPENDIX III 89
Results for analyzing the effect of quality and quantity of the feasible solutions: 89
2000 feasible solutions 89
500 feasible solutions 89
100 feasible solutions 90
APPENDIX IV 91
File Formats 91
Implementation details for SCIP 91

List of Tables

Table 1 : List of the existing variables selection methods studied from the literature 25
Table 2: Naming convention for the variants of the MCFCNF problem..... 45
Table 3 : Naming convention used for the tested datasets..... 47
Table 4 : Parameter settings in SCIP 50
Table 5 : Time in seconds (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)..... 51
Table 6 : MIP Gap (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)..... 52
Table 7 : Total Solving Nodes (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)..... 52
Table 8: Time in seconds (SCIP formulation with disaggregation constraints and with cutting planes)..... 53
Table 9: MIP Gap (SCIP formulation with disaggregation constraints and with cutting planes) 54
Table 10: Average number of branch and bound nodes (SCIP formulation with disaggregation constraints and with cutting planes)..... 54
Table 11 : Formulae for calculating average utilization for the variants of the network design problem 59
Table 12: Quality of feasible solutions collected with SCIP (%MIP Gap) 62
Table 13: Time in seconds and MIP Gap: Leading existing branching rules and the solution based approach..... 65
Table 14: Average of the number of branch and bound nodes: Leading existing branching rules and the solution based approach 65
Table 15: Rank of the Solution based Branching Rule..... 66
Table 16: Datasets for which the solution based branching rule shows significantly poor performance 67
Table 17: Geometric mean of Time in seconds 67
Table 18: Geometric mean of MIP Gap..... 68
Table 19: Geometric mean of Time in seconds for Variant_B_C 69
Table 20: Geometric mean of MIP Gap for Variant_B_C..... 69
Table 21 : Parameter settings with GLPK 84
Table 22: GLPK formulation with disaggregation, with the default (best-bound) node selection strategy..... 85
Table 23: GLPK formulation with disaggregation, with the pseudo-cost based node selection .. 85

List of Figures

Figure 1: Branch and bound search tree 12
Figure 2: Branching on a variable (Figures taken from [30]) 13
Figure 3: conventions used for the description of algorithms 26
Figure 4 : Most Fractional Variable Selection Method 27
Figure 5 : Least Fractional Variable Selection Method 27
Figure 6 : Pseudo-Cost Branching Method (Figure taken from [30]) 30
Figure 7 : Strong Branching Method 32
Figure 8: MCFCNF problem – a directed network graph 37

1. Introduction

Branch and bound based algorithms are used by many commercial mixed integer programming solvers for solving complex optimization problems [3]. Real world optimization problems often involve a very large number of variables and constraints. For NP-hard problems, the time required to solve the problem increases exponentially in the size of the problem [7]. This degrades the performance of branch and bound based algorithms with respect to solution time and solution quality. Various techniques such as cut generation, heuristic solutions, and preprocessing have been developed to reduce the solution time of branch and bound based algorithms. Because branch-and-bound method repeatedly solves restrictions of the original problem (called sub-problems), an important decision that can affect the performance of branch and bound algorithms significantly is 'what sub-problems to create next' and 'which of the sub-problems to explore next' [3]. These techniques are generally referred as 'branching rules' for the branch and bound based search algorithm.

1.1 Background and Motivation

Branch and bound based methods typically explore a sub-problem that is an integer program by solving its linear programming (LP) relaxation. The most common branching rule, known as variable dichotomy or simply branching on a variable, creates sub-problems by bounding the value of a variable whose value is fractional in that LP relaxation [3]. The research that has been conducted for developing efficient variable selection strategies has been focused on general mixed integer programming problems and has shown that these strategies can significantly affect solution times of branch and bound based algorithms [3]. However, these methods have not been computationally analyzed in particular to the multi-commodity fixed charged network flow (MCFCNF) problems.

MCFCNF problems are an academically and practically-relevant class of mixed integer problems. Practical applications of MCFCNF problems include the airline schedule design problem [4], package delivery problem [23], and telecommunication problems [11]. In the MCFCNF problems, multiple commodities are to be routed through a directed network consisting of nodes and arcs with limited capacities. Each commodity is associated with a certain demand which is to be routed from a particular node in the network called a source-node to another node in the network called a destination-node. The objective is to select arcs to be installed in the network in order to reduce total cost for satisfying all the demand with arc capacities as a constraint. The total cost consists of two components: fixed cost for installing a network and variable cost that depends on the amount of commodity to be routed. Realistically sized instances of MCFCNF problems are NP-hard problems [7] and have a large number of nodes and arcs in the network. Furthermore, a large number of commodities need to be routed through the network. While solving such complex optimization problems, the performance of branch and bound based algorithms may get degraded with respect to solution times and solution quality.

Branching strategies can have significant impact on solution times. Various existing branching strategies have been analyzed in literature for general MIPs. However, these strategies have not been computationally analyzed in particular to the network design problems. Network design problems are an academically and practically significant class of mixed integer programming problems. This suggests that understanding the best branching strategy for that particular class of problems is critical. This motivates the idea of my thesis to evaluate which of the existing strategies has the best performance for MCFCNF problems and to design new strategies for solving MCFCNF problems. We have done so by analyzing the performance of various branching strategies with SCIP and GLPK mathematical solvers. For our analysis, we used different statistics such as solution time, MIP gap and total number of branch and bound nodes.

Many popular optimization solvers like CLPEX [8] provide the option of selecting a branching rule by the users [20]. The results of this thesis will help users choose the best option when solving a MCFCNF problem. In addition to analyzing the existing techniques, this thesis is also aimed to propose a new variable selection method. Because MCFCNF-type substructures appear within various real-world network design optimization problems, this analysis of branching rules

may help users achieve better solutions within shorter time for real-world network design optimization problems.

1.2 Problem Statement

The first objective of this thesis is to determine the existing branching strategy that performs well for MCFCNF problems. We have done this by implementing and evaluating existing branching strategies from the academic literature on various instances with different characteristics. Thereafter, the second goal of this thesis is to design new branching strategies for MCFCNF problems.

In this document, we present the computational analysis of some of the existing branching rules for MCFCNF. These experiments have been implemented using SCIP (Solved Constraint Integer Programming) [30] and GLPK (GNU Linear Programming Kit) [21] software libraries, which provide the underlying MIP solver framework. Various control parameters are used to set the behavior of the MIP solver. The call back routines allow the user to modify the behavior of branch and bound based solving process at the time of branching.

This document is organized as follows. After a brief introduction of the branch and bound algorithm in section 2, section 3 presents various existing variable selection strategies that we have implemented. Section 4 presents the formulation and the different variants of MCFCNF problem that we have studied. The computational experiments, experimental setup with SCIP and results for the existing branching rules, are presented in section 5. We then introduce the solution based approach used for developing a new branching rule in section 6. Section 7 presents the computational results for this new solution based approach. The report concludes with a discussion of the results and conclusions in section 8.

2. Branch and Bound Based Algorithm

Branch and bound based algorithms are the most commonly used algorithms by commercial mixed integer programming solvers [1]. In these algorithms, the feasible region is divided into smaller regions, in hopes that solving multiple optimization problems over smaller regions (referred to as solving a sub-problem) is easier than optimizing over the entire feasible region at once. This process of dividing the feasible region and solving resulting sub-problems is often represented with a binary tree, called a branch-and-bound search tree, with a node of the tree representing a sub-problem [32]. The nodes thus created are explored subsequently in the similar way, by dividing the feasible region further into the smaller sub-problems that are hopefully easier. All the information is then put together again to solve the original problem.

2.1 Branch and Bound Search Tree

Figure 1 shows the general structure of the branch and bound search tree. The top node in the tree is referred to as the root node, and represents the original problem to optimize over the entire feasible region. The root node is said to be at level 0 in a branch and bound search tree.

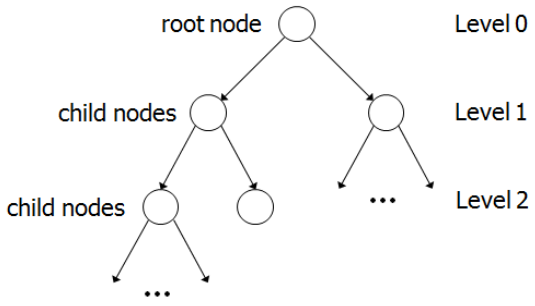

Figure 1: Branch and bound search tree

The feasible region is then divided into two smaller sub-problems. Each sub-problem is represented as a child node of the node representing the problem from which the sub-problem was derived. Thus, each child node is associated with certain parent node and a node level. Typically, the feasible regions are divided into smaller sub-problems by adding bounds on a

variable which is integral, but has fractional value in the LP relaxation of that feasible region. This is called branching.

2.2 Branching

Part (a) of figure 2 [30] represents entire feasible region of the optimization problem to be solved that corresponds to a particular node. Each division of feasible region as indicated in part (b) of figure 2 [30] represents a child node.

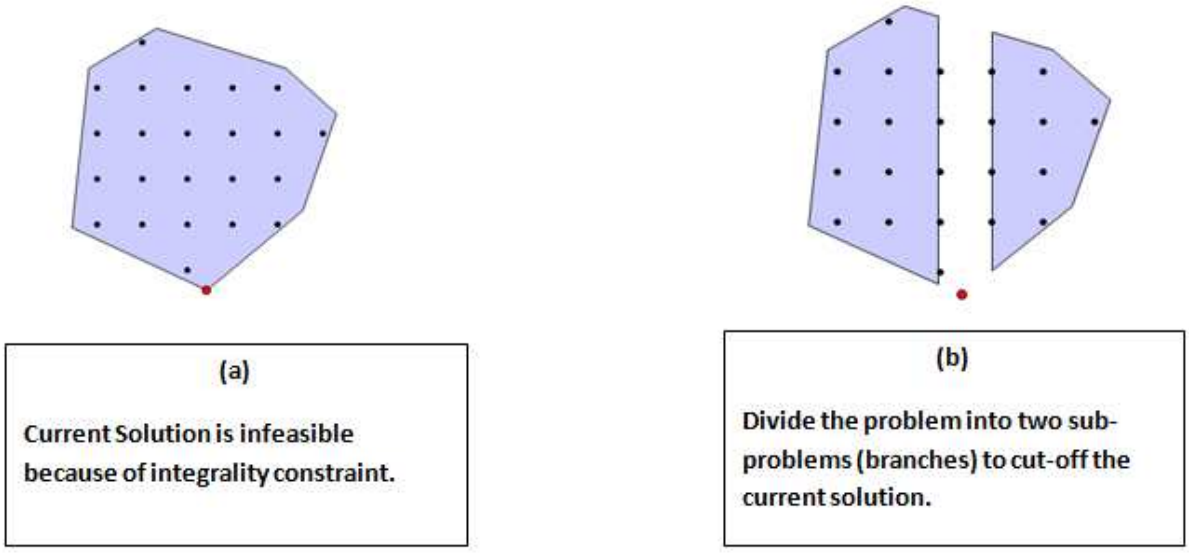

Figure 2: Branching on a variable (Figures taken from [30])

For a minimization problem, the primal (upper) bounds on the optimal value of a sub-problem are provided by feasible solutions and the dual (lower) bounds are obtained by relaxation of the feasible region [6]. Using this information, many nodes in a branch and bound search tree are implicitly enumerated.

Assumptions:

We use the following notation.

Consider a minimization mixed integer programming problem corresponding to S.

Let $S = S_1 \cup .... \cup S_k$ be the decomposition of S into smaller sets, and

c = objective function coefficients for the decision variables x.

Let $z_k = \min \{cx : x \in S_k\}$ for $k = 1, \ldots, K$,

z = optimal solution of the main problem S.

Let z_{k_up} be an upper bound on z_k and z_{k_low} be a lower bound on z_k.

$z_{up} = \min_k z_{k\_up}$ is an upper bound on z and $z_{low} = \min_k z_{k\_low}$ is a lower bound on z.

Pruning of nodes:

There are three basic reasons to prune the nodes in a branch and bound search tree, and thus enumerate a large number of solutions implicitly [32].

I. Pruning by optimality:

A node in a branch and bound search tree can be pruned by optimality if a sub-problem associated with that node has been solved and the integer optimal solution for that sub-problem is obtained. That is, if $zt = \{ \min cx : x \in S_t \}$ has been solved, prune the node corresponding to sub-problem S_t.

II. Pruning by bound:

A node can be pruned if lower bound on the optimal solution of the sub-problem associated with that node is greater than or equal to the global upper bound.

That is, a node corresponding to set S_k can be pruned by bound if $z_{k\_low} \geq z_{up}$ (for a minimization problem).

III. Pruning by infeasibility:

A node corresponding to the sub-problem S_t can be pruned by infeasibility if $S_t = \phi$ (void or infeasible).

2.3 Algorithm for Branch and Bound Search Method

A typical algorithm for branch and bound search method [32] for solving a mixed integer programming problem is presented as follows. Because there is a one-to-one mapping between nodes and sub-problems, we use the terms interchangeably.

Input: A mixed-integer program.

Initialization:

List of unexplored nodes = Initial problem S with formulation P on list.

Global upper bound = $z_{\_up}$= Infinity.

Global lower bound = $z_{\_low}$ = 0.

Incumbent solution x* = void (φ).

Algorithm 1: Branch and bound based algorithm

---

1. A. Compare global upper and lower bounds against optimality tolerance.

a) If $z_{\_up} = z_{\_low}$, STOP.

b) If the relative gap between $z_{\_up}$ and $z_{\_low}$ $[\,(z_{\_up} - z_{\_low})/(z_{\_up})\,] \leq$ specified MIP optimality tolerance level, STOP. (If $z_{\_up} = 0$, then denominator is set to a small value).

B. If the list of unexplored nodes is empty,

a) Return the incumbent x* as optimal solution, and STOP.

b) If there is no incumbent, return that the problem is infeasible and STOP.

2. Select a set S^i with the formulation P^i to solve next from the list of unexplored nodes.

a) Solve the LP relaxation over P^i of the selected problem.

b) Dual bound $z^i_{\_low}$= LP solution value.

c) x^i(LP) = LP solution.

d) Update global lower bound: $z_{\_low} = \min_i z^i_{\_low}$

3. Pruning:

a) If P^i is empty, LP relaxation is infeasible. This makes IP infeasible. Prune by infeasibility.

Go to step 1.

b) If for LP relaxation, dual bound of current node $z^i_{\_low} \geq$ global upper bound $z_{\_up}$, prune by bound.

Go to step 1.

c) If LP relaxation is integer-feasible, that is, if x(LP) is integer, update the primal bound $z_{\_up} = z^i_{\_up}$, and incumbent $x^* = x^i$ (LP). Prune by optimality.

Go to step 1.

4. Select a variable for branching from the set of candidate variables (Candidate variables are the integer bounded variables that have fractional values in the current LP relaxation solution).

5. Create two sub-problems (child nodes) from the current node by branching on the selected variable and add these new nodes to the list of unexplored nodes. This will return two sub-sets $S_1^i \; and \; S_2^i$ with formulations $P_1^i \; and \; P_2^i$.

6. Go to step 1.

Instead of solving the problem to optimality, we can also specify an optimality tolerance level for branch and bound algorithm. The relative gap between global upper bound ($z_{\_up}$) and global lower bound ($z_{\_low}$) is calculated as [$(z_{\_up} - z_{\_low})/(z_{\_up})$], except for when $z_{\_up}$ is 0, in which case the denominator is set to a small value. This relative gap is compared with the specified optimality tolerance level [8, 21]. In this case, the execution of branch and bound algorithm stops just after reaching the optimality tolerance level even though the list of unexplored sub-problems is not empty. This guarantees that a solution lies within a certain percentage of the optimal solution. The solvers such as CPLEX [8] also allow for specifying absolute tolerance level.

Furthermore, we can also introduce cuts after solving the LP relaxation as mentioned in step 2. After obtaining an optimal solution for LP relaxation, a cutting plane algorithm can be used to find further linear constraints that are satisfied by all feasible integer solutions of the current sub-problem, but violated by the current fractional solution. If such an inequality is found, it is added to the linear constraints. This new formulation is solved again and a new solution is obtained. This process is repeated until either an integer solution is found (which is then known to be optimal solution for that sub-problem) or until no more cutting planes are found [3].

2.4 Methods to Improve the Performance of Branch and Bound Based Algorithms

Consider a minimization mixed integer programming problem. The optimal solution value for this problem is always less than any other feasible solution for this problem [6]. Hence, any primal feasible solution provides an upper bound on the optimal solution value whereas any dual feasible solution obtained by LP relaxation provides a lower bound on the optimal solution of that problem. The smallest among all LP relaxation values associated with the active sub-problems provides a global lower bound on the optimal value. When the global lower bound and global upper bound are equal, in which case the list L of active sub-problems vanishes. This terminates the branch and bound algorithm according to step 1 of the branch and bound algorithm (presented above in section 2.3). Decreasing the global upper bounds is accomplished by finding improved feasible solutions during search process for minimization problems [3]. Some of the techniques used to improve the bounds quickly are presented next.

Preprocessing [3]:

Before solving an integer program, many mixed integer programming solvers check if the formulation is as strong as possible given the information available. The basic idea behind this is to detect and eliminate the redundant constraints and variables, and tighten the formulation whenever possible. Feasible regions for the resulting formulations are smaller than the original problem and hence require less time to solve using branch and bound based algorithms.

Following example illustrates the preprocessing method for the integer programming problem [29, 32].

Maximize $5x1 + 3x2 + 2x3 + x4$

Subject to:

$7x1 + 3x2 - 4x3 - 2x4 \leq 1$

$-2x1 + 7x2 + 3x3 + 4x4 \leq 6$

$-2x2 - 3x3 - 6x4 \leq -5$

$3x1 - 2x3 \geq 1$

$x1, x2, x3, x4 = \{0, 1\}$

For preprocessing, we can add some derived logical constraints in the model.

1. From constraint 1 ,
   a) if x1 = 1 then x3 = 1, thus $x1 \leq x3$
   b) if x1 = 1 then x4 = 1, thus $x1 \leq x4$
   c) if x1 = 1 and x2 = 1 then the problem becomes infeasible, thus $x1 + x2 \leq 1$

2. From constraint 2,
   a) if x2 = 1 then x1 = 1, thus $x2 \leq x1$

From steps 1 and 2, we have ($x1 + x2 \leq 1$) and ($x2 \leq x1$)
Thus, we can derive that x2 = 0.

Hence, we can add the constraint x2 = 0 in the model and repeat.

We can similarly pass through all constraints and all decision variables and repeat the same process. As we can notice from the above example, we can add a derived constraint (x2 = 0) during preprocessing. This makes the feasible region smaller and hence improves solution time as compared to the original problem. Some other techniques for preprocessing are tightening of

variable bounds, identification of infeasibility, identification of redundant constraints and variables [29].

Valid Inequalities:

Mathematical definition of valid inequalities is presented as follows [28].

An inequality $\pi x \leq \pi_0$ is a valid inequality for $X \subseteq R^n$ if $\pi x \leq \pi_0$ for all $x \in X$.

Valid inequalities are the constraints added to the original sub-problem that are violated by the optimal LP solution of the original problem, but at least one optimal integer solution for this problem satisfies these inequalities. Valid inequalities make the formulation stronger by reducing the size of feasible region. Branch and bound based algorithms take less time to solve the problem because of smaller feasible region. Examples of valid inequalities are Gomory mixed integer cuts, cover inequalities and mixed integer rounding inequalities [32].

The cover inequalities technique is an example of the general cutting-plane method.

Consider a binary knapsack problem with decision variables x_j:

Maximize $\sum_{j\in N} c_j x_j$, Subject to $\sum_{j\in N} a_j x_j \leq b$,

Let N denote $\{1, \dots, n\}$. A set $C \in N$ is called a cover if it satisfies $\sum_{j\in C} a_j > b$.

Since it is impossible to set all the variables in C to 1 simultaneously while maintaining feasibility, the linear inequality $\sum_{j\in C} x_j \leq |C| - 1$ is valid inequality for that problem. This is called cover inequality [32].

Following example illustrates a cover inequality for a binary knapsack problem.

Consider a binary knapsack problem,

Maximize $x1 + x2 + x3 + x4$

Subject to:

$$11x1 + 6x2 + 6x3 + 5x4 \leq 19$$

$$x1,\ x2,\ x3,\ x4 = \{0,1\}$$

For the relaxed knapsack problem (without binary constraints), bounds on the decision variables will be

$$0 \leq \{x1, x2, x3, x4\} \leq 1$$

LP relaxation solution for this relaxed knapsack problem is$\{2/11, 1, 1, 1\}$.

We then construct a linear inequality that is valid (we can guarantee that all binary vectors feasible to the given knapsack problem satisfy the inequality) but which is not satisfied by the fractional vector $\{2/11, 1, 1, 1\}$ that we obtained after solving the relaxed problem.

Thus, we can add a cover inequality $x1 + x2 + x3 \leq 2$. This is a valid inequality as it is satisfied by all the integer feasible solution of a given binary knapsack problem, and it cuts off the LP relaxation solution$\{2/11, 1, 1, 1\}$.

Heuristics:

Heuristics are used to improve the upper bounds by providing a good upper bound early in the branch and bound search process [3]. This improves the performance of branch and bound method with respect to the solution time. Examples of the heuristics that are implemented in many mixed integer programming solvers are local branching method [16] and RINS (Relaxation Induced Neighborhood Search) [13].

Branching Rules

Branching is an important step for any branch and bound based algorithm. The research on the branching strategies has shown that these strategies can significantly affect solution times of branch and bound based algorithms [3]. We present existing branching strategies that we have analyzed for MCFCNF problem in order to determine which of these strategies performs best for MCFCNF.

Branching rules provide two main choices.

1: Variable Selection Methods:

One of the branching rules is based on the variable selection methods that determine how the current problem is divided into smaller sub-problems that are easier to solve than the original problem. One way to divide a feasible region associated with the original problem is to choose an integer variable with a fractional value in the current LP relaxation [3]. Typically, the integer variables that are fractional in the LP relaxation solution of a present problem are considered as candidate variables for branching. The variable selection methods determine which of these candidate variables to branch on next. The two sub-problems are created by changing the bounds on the selected variable for branching.

For example, consider a mixed integer programming problem with 5 decision variables {x1, x2, y1, y2, y3}, where the variables x1 and x2 are continuous, and variables y1, y2, and y3 are binary. Suppose that a LP relaxation solution of this binary mixed integer programming problem is {12, 15, 1, 0.2, 0.5}. The integer variables that have fractional values in the current LP relaxation form a set of candidate variables for branching. In this problem, variables y2 and y3 (with LP relaxation values 0.2 and 0.5) will be listed as candidate variables for branching. If we apply the most fractional variable selection method, variable y3 will be selected as a branching variable as it has the most fractional value in the current LP relaxation solution.

2: Node Selection Methods:

Another branching rule is based on the node selection methods. After dividing the original problem into two sub-problems, the node selection methods are used to select one of the sub-problems from the list of unexplored sub-problems to process next once the current node is solved. One of the methods used to select the nodes is a depth first search method. This method always selects a node that is a child of the node currently being explored and thus attempts to find the first integer feasible solution quickly. The depth first search method mainly focuses on global upper bound found (for a minimization problem) from the available feasible solutions as

researchers have found that feasible solutions are often found deep in the branch and bound search tree [3].

The best bound search method selects a node that gives the best dual (lower) bound. This method favors exploring nodes at the top of branch and bound search tree by finding the nodes with the lowest value of LP relaxations (for a minimization problem). If z^{opt} is the optimum solution to a minimization problem, then node k is said to be superfluous is $z^k > z^{opt}$, where z^k is the lower bound for node k [3]. Best bound search ensures that no superfluous nodes will be explored.

The linear programming based branch-and-bound method and its various challenges have been discussed in Atumturk et al. (2005) [3]. These challenges include branching rules, search techniques for exploring nodes in a branch-and-bound decision tree, cut generation and preprocessing techniques. Further references for the related work can be found in this paper. This paper also reviews major algorithmic components of the commercial MIP solvers, and discuss the various options that are available to the user to control the behavior of MIP solvers. Even though there has been significant development in integer programming solvers, there still are many challenges for these solvers. These challenges have been addressed and a possible future scope for improving the performance of branch and bound based algorithms has been discussed in this paper.

3. Overview of the Variable Selection Methods

In the branch and bound based algorithms, the feasible region is divided into smaller sub-problems. This is called branching. For branching, a variable is selected from the set of integer variables that have fractional values in the current LP relaxation. This set is referred as the set of candidate variables for branching.

The generalized algorithm for the implementation of variable selection methods [1] can be stated as follows.

Algorithm 2: variable selection

Input: LP relaxation solution of the current sub-problem to be solved next
Output: index of the variable to be branched on next.

1. Determine the set of the candidate variables for branching.
2. For each candidate variable, calculate the score based on a particular branching rule.
3. Sort the variables based on score values.
4. Return the index of variable with maximum score / (or minimum score based on a particular branching rule).

We now present a brief review of the literature on various branching rules. The classification of these branching rules and a brief algorithmic description is presented further in section 3.2.

3.1 Literature Review

The research on the branching strategies has shown that these strategies can significantly affect solution times of branch and bound based algorithms [3]. It has been experimentally shown in Achterberg et al. (2005) [1] that the branching technique that selects the most fractional variable as discussed in Atumturk et al. (2005) [3] is not much efficient than randomly selecting a branching variable.

Traditional approaches for selecting branching variables rely on estimating the impact of the candidate variables on the objective functions. Various branching rules based on the estimate of

the impact of a variable on the objective function have been discussed in Atumturk et al. (2005) [3]. Similar methods that use the idea of selecting a branching variable that maximizes the degradation in the objective function value at the child node LP relaxation are presented in Dakin (1965) [12], and Eckstein (1994) [14].

Instead of measuring the impact on the objective function, a branching rule based on measuring the impact of variables on the active-constraints in current LP relaxation has been proposed in Patel et al. (2007) [26]. These branching rules select a branching variable using three main steps 1. Normalization of the active constraints (By dividing through by the number of candidate variables that appear in the constraint) 2. Assigning scores to the candidate variables by summing the normalization effects over all active constraints. 3. Selecting the branching variable with maximum score.

A new approach for selecting the branching variable by considering uncertainty at a particular node has been developed in Gilpin et al. (2011) [20]. This uncertainty is measured by calculating entropy using an information theoretic approach. Some definitions from information theory, such as the notion of entropy [31], which measures the amount of uncertainty in a random event, have been used while developing this information theoretic approach. The strong branching method and hybrid approaches involving strong branching and entropy based branching are also explained in Gilpin et al. (2011) [20].

Various branching strategies such as most-infeasible branching, pseudo-cost branching and strong branching have been explained in Achterberg et al. (2005) [1]. Pseudo-cost branching rule presented here is a sophisticated rule and it makes use of the statistics from a current branch and bound tree. This method has drawback that the statistics available in the beginning of branch and bound tree are not reliable. To overcome the drawback of pseudo-cost branching, a new branching technique called ‘reliability branching’, which is a hybrid method that uses pseudo-cost branching and strong branching has also been introduced in Achterberg el al. (2005) [1].

The following table lists various branching strategies that we have studied from the above literature for the network design problem.

Table 1 : List of the existing variables selection methods studied from the literature

No.	Variable Selection Method	Reference	Author	Year
1	Pseudo-Cost Branching Considering the History of Branching Variables	A computational Study of Search Strategies for Mixed Integer Programming	J. Linderoth, M. Savelsbergh	1999
2	Most Infeasible Branching	Integer-Programming Software Systems	A. Atumturk, M. Savelsbergh	2005
3	Best Estimate Methods	Integer-Programming Software Systems	A. Atumturk, M. Savelsbergh	2005
4	Strong Branching	Branching Rules Revisited	T. Achterberg	2005
5	Reliability Branching (Hybrid Method Consisting the Strong and Pseudo-Cost Branching)	Branching Rules Revisited	T. Achterberg	2005
6	Entropic Branching	Information Theoretic Aproaches to Branching in Search	A. Gilpin, T. Sandholm	2011
7	Entropic Lookahead Branching	Information Theoretic Aproaches to Branching in Search	A. Gilpin, T. Sandholm	2011

In the literature presented above, various branching rules have been developed and studied for general MIPs. However, the performance of these branching rules remains to be studied particularly for the MCFCNF problem.

3.2 Classification of Variable Selection Methods

In this section, we present the classification of variable selection methods. We also briefly present a literature review and the algorithms for each of these techniques.

Conventions used in the description of the algorithms:

- x^* = The optimal solution to the current node's LP
- candidates $\leftarrow \{i \mid x_i \text{ fractional}\}$
- x_i = variable to be branched on (binary variable which is fractional in the current LP relaxation)
- $x_f = x_i - \text{floor}(x_i)$

- z = Objective value of LP relaxation at the current node
- z_l = Lower bound on the objective value of the down branch
- z_u = Lower bound on the objective value of the up branch

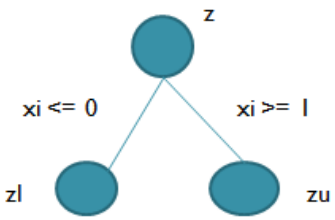

Figure 3: conventions used for the description of algorithms

3.2.1 Methods Based on the Current LP Relaxation

In these methods, the information used for variable selection is obtained from the LP relaxation solution at the current node in branch and bound tree. For example, the method of branching on the most fractional variable selects a variable whose value in the LP relaxation solution is furthest from being integral [3]. Selecting the variable that has the most fractional value in the current LP relaxation requires negligible computation time. Recent research shows, however, that this method is not much better than randomly selecting a branching variable [1]. In this method, the score for the branching variables (x_i) is calculated as follows:

Branching on the Most Fractional Variable

For each i ∈ candidates:

1. score (x_i) ← x_i – floor (x_i) for $x_i \leq 0.5$
   ← ceil (x_i) – x_i for $x_i > 0.5$
2. i* = arg {max i∈candidates{score(x_i)}}
3. Return i∗

The branching variables with the maximum score is then selected for branching.

For example, suppose that we have a set of candidate variables y1, y2, and y3 that are binary variables having fractional values in the current LP relaxation solution. Let the values of these variables in the current LP relaxation be 0.5, 0.3 and 0.9. The most fractional variable selection method selects a variable y1 for branching as it has the most fractional value.

yi = {0,1}	y1	y2	y3
Values in the current LP solution	0.5	0.3	0.9

Figure 4 : Most Fractional Variable Selection Method

Similarly, the least fractional variable selection method selects a candidate variable with the least fractional value in the current LP relaxation solution.

Branching on the Least Fractional Variable

For each i ∈ candidates:

1. score (x_i) $\leftarrow x_i - \text{floor}(x_i)$ for $x_i \leq 0.5$

   $\leftarrow \text{ceil}(x_i) - x_i$ for $x_i > 0.5$

2. $i^* = \arg\{\min i \in \text{candidates}\{\text{score}(x_i)\}\}$

3. Return i^*

The branching variables with the minimum score (least fractional) is then selected for branching.

For the same example presented above, the least fractional variable selection method selects variable y3 as a branching variable.

yi = {0,1}	y1	y2	y3
Values in the current LP solution	0.5	0.3	0.9

Figure 5 : Least Fractional Variable Selection Method

Information theoretic branching (without look-ahead) as presented in Gilpin el al. (2011) [20] ranks the variables based on the entropy calculations by considering values of the variables in

current LP relaxation. A new approach for selecting a branching variable has been developed by considering uncertainty in the variable values involved at a particular node in this paper. This uncertainty in the variable values is measured by calculating the entropy using information theoretic approach. Entropy is considered as a probability of each outcome (0 or 1) occurring. In this approach, the fractional portion of the integer-constrained variable in the current LP solution is treated as probability with which we expect the variable to be greater than its current value in the optimal solution.

The entropy value for each candidate branching variables is calculated as follows:

Entropic Branching

For each $i \in$ candidates:

1. entropy(x_i) =

$$e(x_i) = -x_i \log_2 x_i - (1 - x_i)\log_2(1 - x_i) \qquad 0 < x_i < 1$$

$$= 0 \qquad x_i \in \{0,1\}$$

2. $i^* = \arg\{\min i \in \text{candidates}\{\text{entropy}(x_i)\}\}$

3. Return $i*$

The branching variable with the minimum entropy is then selected for branching.

At the beginning of a search, the nodes on the frontier of a branch and bound search tree have large amount of uncertainty about variable values. A path in the branch and bound search tree ends when there is no uncertainty left in variable values. Selecting a branching variable, that involves less uncertainty, from a set of candidate variables, will result in better performance of the search algorithm. Thus, the idea is to drive the search to reduce uncertainty (entropy) in the current sub-problem.

Computation time required for selecting a variable at each node is very low for these methods as they only require information related to the current LP relaxation solution. However, these

methods are not very sophisticated, and often perform poorly, particularly on the larger instances.

3.2.2 Methods Based on the Current Statistics Calculations

In these methods, statistics calculated during branch-and-bound for each candidate variable are used for ranking the variables. For example, the pseudo-cost branching [1] assigns the scores to the candidate variables by calculating pseudo-costs associated with each variable. These pseudo-cost calculations consider the history of branching success of the variables in the branch and bound tree. Branching success of a variable considers change in bound per unit change in the variable value when that variable is selected for branching.

Pseudo-cost branching explained in Achterberg el al. (2005) [1] keeps a history of the success of the variables, which already have been branched on. Whenever a variable is selected for branching, the information such as change in bound per unit change in the variable value, number of times that variable is branched on is stored as a branching history of that variable. Pseudo-costs for each candidate variable are calculated from this history that is stored whenever a particular variable is selected for branching.

The following algorithm briefly describes how the pseudo-costs are calculated for a branching variable:

Pseudo-Cost Branching

For each i ∈ candidates:

1. $\Delta_i^+ = z_u - z$, $\Delta_i^- = z_l - z$
2. f_i^+ = ceil (x_i) – x_i and f_i^- = x_i – floor (x_i)
3. $\varsigma_i^+ = \Delta_i^+ / f_i^+$

4. Calculate pseudo-cost considering the branching history of variable for upward branching.

$$\Psi_i^+ = \sigma_i^+ / \eta_i^+$$

Where , σ is the sum of gain per unit change (ς) over all problems where variable ' i ' was selected as branching variable, and η is number of such problems.

5. score(xi) = score($f_i^- \Psi_i^-, f_i^+ \Psi_i^+$)

Where, $score(q^-, q^+) = (1 - \mu) * \min\{q^-, q^+\} + \mu * \max\{q^-, q^+\}$

For the variables which are not branched on before, that is, η = 0, the average of the initialized pseudo-costs over all the variables is used.

The variable with maximum score is then selected for branching. The branching history of the selected variable is updated before continuing further branching.

As this method uses the statistics available from the current branch and bound tree, no additional computations are involved while assigning scores to the candidate variables. Furthermore, this method has no computational burden while processing each node as the LP relaxations at the child nodes are not required to be calculated for each candidate variable. However, the weakness of pseudo-cost branching is that reliable and enough information is not available in the beginning of branch and bound search tree, and the pseudo-costs are almost identical for all candidate variables.

The following example (taken from [30]) illustrates the pseudo-cost branching rule for variable selection.

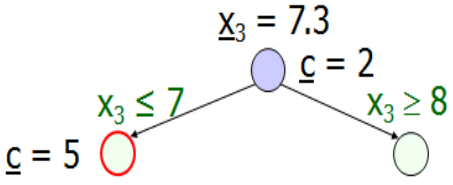

Figure 6 : Pseudo-Cost Branching Method (Figure taken from [30])

LP relaxation at the current node (indicated by a filled circle in the above diagram) yields a lower bound on the objective value. Let $\underline{c}$ denote the lower bound on objective value of the problem associated with each node. Integer variable x_3 has fractional value 7.3 (denoted by $\underline{x_3}$) in the current LP relaxation solution. Branching on this variable decomposes the current problem into smaller sub-problems with the inequalities $x_3 \leq 7$ and $x_3 \geq 8$. LP relaxation for the sub-problems yields a lower bound on their objective values.

Objective gain per unit change on the lower branch for variable x3 is then calculated as:

$$\text{Objective gain per unit change in variable } x_3 = \frac{5-2}{7.3-7} = \frac{3}{0.3} = 10$$

Pseudo-cost of variable x_3 for the down branch is then calculated as average of all objective gains per unit over all the occurrences for which the variable x_3 has been branched on. Pseudo-cost for the up branch is calculated similarly. These pseudo-cost values are further used to assign scores to the candidate variable as mentioned in step 5.

3.2.3 Methods Considering the LP Relaxation of Child Nodes

The strong branching method [20], best estimate methods [3], and the entropic look-ahead method [20] use statistics related to the LP relaxation at the child nodes to evaluate candidate variables.

The variable selection strategies addressed in Atumturk et al. (2005) [3] are the best estimate methods and strong branching methods. The best estimate methods assign the scores to the candidate branching variables by computing the estimates on the degradation in LP objective. One way to obtain the values of these estimates is to simply observe the increase in the LP bound due to branching using dual simplex pivots for the child nodes. The degradation estimations of the lower and upper branches are then combined to assign score to the candidate variables.

Instead of estimating the change in bounds on the objective function values of the child nodes, strong branching method partially solves the LP at child-nodes using smaller number of simplex

iterations and observes the change in objective function value. The computational requirements while solving LPs just to identify the best candidate variable for branching can become prohibitive. Typically, only few LP iterations over limited set of candidate variables are executed [3]. In the Strong Branching method, the score for the candidate variables is assigned as follows:

Strong Branching

For each i ∈ candidates:

1. $\text{score}(x_i) = 10 * \min\{z_l, z_u\} + \max\{z_l, z_u\}$
2. $i^* = \arg\{\max i \in \text{candidates}\{\text{score}(x_i)\}\}$
3. Return $i*$

The candidate variable with maximum score is then selected for branching.

Consider the following example that illustrates the strong branching method. Assume that we have LP relaxation solution at the current node and variable x_3 has value 7.3 (denoted by $\underline{x_3}$) in this LP solution. Current problem is divided into smaller sub-problems (that correspond to the child nodes) by adding constraints $x_3 \leq 7$ and $x_3 \geq 8$. In strong branching method, we solve LP relaxation problems at the child-nodes using smaller number of simplex iterations and observe change in the objective function values. Let $\underline{c}$ denote the LP relaxation solution value for each node.

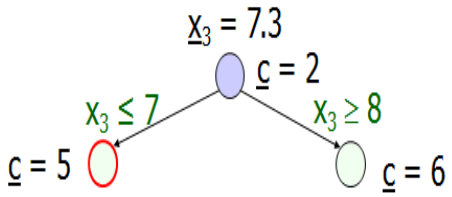

Figure 7 : Strong Branching Method

For this example, score for the candidate variable x_3 using strong branching method will be calculated as follows:

$$\text{score}(x_3) = 10 * \min\{5,6\} + \max\{5,6\} = 56$$

As discussed in section 3.2.1, the entropic branching considers the entropy or uncertainty left in the variable values for assigning the scores. Entropic look-ahead branching considers one step look-ahead (by calculating LP relaxations at child nodes) as in strong branching [20]. The difference is that, instead of examining the objective values of the potential child nodes as in strong branching, the remaining uncertainty in the potential child-nodes is calculated in entropic look-ahead branching. Variable with the least uncertainty is chosen for branching. This algorithm can further be modified for more than one step look-ahead, which may further reduce the size of branch and bound search tree.

With slight modification to the entropic branching method, the entropy for the entropic look-ahead method is calculated as follows:

Entropic Look-ahead Branching

For each $i \in$ candidates:

1. $\text{entropy}(x_i) = \sum_{j=1}^{n}[(1 - x_f) * e(x_j^l) + x_f * e(x_j^u)]$
2. $i^* = \arg\{\min i \in \text{candidates}\{\text{entropy}(x_i)\}\}$
3. Return $i*$

The candidate with minimum entropy value is then selected for branching.

The drawback of these methods is that the computation time required at each node is very high. However, for the larger problems, time required for this computational requirement may get compensated by the performance of this method as well as by limiting the number of simplex iterations for calculating LP relaxation of child nodes as in strong branching.

3.2.4 Hybrid Methods

The hybrid methods use more than one existing algorithms for selecting the branching variable. Various hybrid methods are discussed in Gilpin et al. (2011) [20] that use both entropic branching and strong branching for assigning scores to the candidate variables. Example of this

is the tie-breaking approach in which the scores are first assigned using entropy calculations, and then the ties while assigning the scores are broken using strong branching.

To overcome the drawback of pseudo-cost branching for the uninitialized variables as mentioned before, Achterberg et al. (2005) [1] introduces reliability branching, a hybrid of pseudo-cost and strong branching. In this branching technique, the scores for variables that have not been branched on a sufficient number of times (specified by the algorithm parameter η rel) are assigned based on the strong branching method. The pseudo-cost branching method is then used to assign the scores for the candidate variables for which branching history is available.

The following algorithm describes how the scores are assigned to the candidate variables in the reliability branching method:

Reliability Branching

---

1. For all candidates i, Calculate score s_i = score based on pseudo-cost calculation.
2. Sort the variables in non-increasing order of pseudo-cost scores.
3. For all candidates, with min $\{\eta^-, \eta^+\} < \eta$ rel

   (where η is number of times the variable is branched on before.

   η rel = parameter for strong branching initialization.)

   a. Update scores based on strong branching

   b. If maximum score is not changed for consecutive updates,

   $i^* = \arg\{\max i \in \text{candidates}\{\text{score}(x_i)\}\}$
4. Return i*.

Thus, the variable i* is selected next for the branching.

The calculation of LP relaxation values is required for strong branching; however, the strong branching is used only for the uninitialized candidate variables. Also, the overhead of calculating LP relaxation values gets compensated due to high performance of this method and by limiting

the number of simplex iterations for strong branching. Extensive computational analysis using various parameter settings has been presented in Achterberg et al. (2005) [1] and it shows the effectiveness of this method. This paper presents computational results for reliability and pseudo-cost branching for certain set of parameter values. We tested the same set of parameters for MCFCNF problem, and determined the best performing parameters over 20 datasets. We used these best performing parameters for our GLPK experiments. For SCIP experiments, we have used the default parameter settings provided with the SCIP solver.

4. Multi-Commodity Fixed Charged Network Flow (MCFCNF) Problem

MCFCNF problems are an academically and practically-relevant class of mixed integer problems. Practical applications of this particular class of network design problems include telecommunication problems [11], package delivery problems [23], airline schedule design problems [4], and the freight transportation problem [9]. For some of these applications, network design problems form a substructure of the formulation of the original problem.

4.1 Problem Description

In the MCFCNF problems, multiple commodities are to be routed through a directed network consisting nodes and arcs with the limited capacities. Each commodity is associated with a certain demand which is to be routed from a particular node in the network called a source-node to another node in the network called a destination-node. We select the arcs to be installed in the network. The objective is to install arcs in the network in order to reduce the total cost for satisfying all the demand with arc capacities as a constraint. The total cost involved in routing these commodities has two components: fixed cost for installing a network, and variable cost which depends on the amount of commodity to be routed [22].

Figure 8 presented below represents a directed network graph with arc capacities indicated as u_i. There are multiple source nodes and sink (destination) nodes for different commodities. All the demands of these commodities are to be satisfied with arc capacities as constraints.

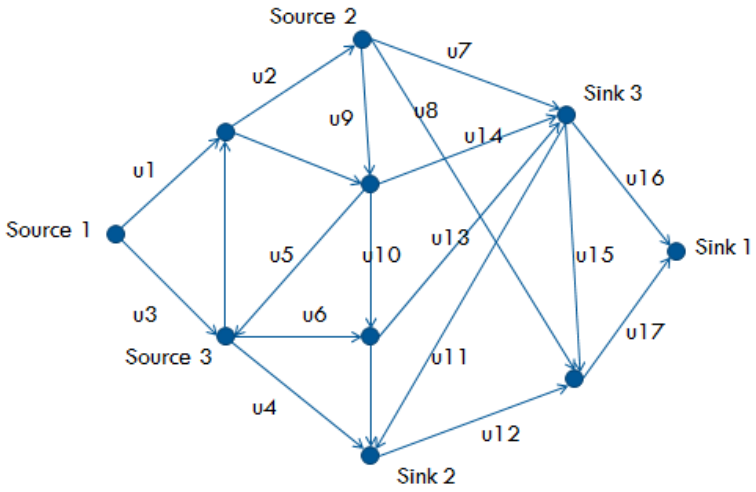

Figure 8: MCFCNF problem – a directed network graph

Realistically sized instances of network design problems have a large number of nodes and arcs in the network and a large number of commodities needed to be routed through a network. Hence, such NP-hard optimization problems have a very large number of constraints and variables. As mentioned in Hewitt et al. (2010) [22], many realistically sized instances of MCFCNF (or instances of models that contain MCFCNF as a substructure) are so large (for example [27]) that sometimes even the LP relaxation of the arc-based integer programming (IP) formulation can be intractable. Furthermore, exact methods are only capable of handling small instances, far smaller than many realistic-sized instances. The performance of branch and bound based algorithms may get degraded with respect to solution times and solution quality while solving such complex optimization problems.

In the next subsection, we present an application of MCFCNF problem (as presented in Crainic et al. (2000) [9]) along with mathematical formulation.

4.2 A Service Network Design Application – Freight Transportation

Service network design problems arise in various applications such as rail-roads, airlines, trucking firms, and Less-Than-Truckload (LTL) motor carriers [9]. These require determination of cost-minimizing or profit-maximizing set of services and schedules, given limited resources.

Freight transportation is an example of the MCFCNF problem as most freight transportation planning issues exhibit a multi-commodity nature.

Freight transportation problems deal with long-haul, intercity transportation. Transportation operations are mainly concerned with the movements of goods over relatively long distances, between terminals or cities. Goods may be moved by rail, truck, ship, or any combination of modes. Tactical (medium term) planning is particularly vital for intercity freight carriers that make intensive use of consolidation operations. The main tactical issues for this type of carriers are selection and scheduling of services, specification of terminal operations, and routing of freight.

We now present a general mathematical formulation for the freight transportation planning problem (as presented in Crainic et al. (2000) [9]).

Consider a graph $G = (N, A)$. Where, N is a vertex set (set of nodes) and A is a link set (set of arcs) in a directed network. The terminals in a transportation companies operation are modeled with nodes and potential services are modeled with arcs. P represents the set of commodities (freights) to be transported through the network.

Decision Variables:

y_{ij} $(y \in Y)$:

These are integer variables modeling discrete choice design decisions. When $Y \in \{0,1\}$, these are the binary decision variables that indicate whether an arc (i,j) is open, that is selected for inclusion in the final network. When $Y \in \{\text{Integers} \geq 0\}$, the y_{ij} variables are not restricted to $\{0,1\}$ values and usually represent the number of facilities or units of capacity installed, or the level of service offered.

x_{ij}^{p} :

These are the continuous flow decision variables that indicate amount of flow of commodity p using link (i,j).

Parameters:

f_{ij} : Fixed cost of opening link (i,j) (selecting for inclusion in the final network), and when y belong to N, hypothesis is that a f_{ij} cost is incurred for each unit of facility installed or service offered

c_{ij}^{p} : The transportation cost per unit of flow of product p on link (i,j).

u_{ij} : The capacity of link (i,j)

d_{i}^{p} : The demand of product p at node i.

Objective Function:

Minimize $\sum_{(ij)\in A} f_{ij} y_{ij} + \sum_{(ij)\in A} \sum_{p\in P} c_{ij}^{p} x_{ij}^{p}$

Subject To:

$$\sum_{j\in N} x_{ij}^{p} - \sum_{j\in N} x_{ji}^{p} = d_{i}^{p} \qquad i \in N, \quad p \in P \qquad (1)$$

$$\sum_{p\in P} x_{ij}^{p} \leq u_{ij} y_{ij} \qquad (i,j) \in A \qquad (2)$$

$$(y, x) \in \tau, \qquad (i,j) \in A, \ p \in P \qquad (3)$$

$$y \in Y, \qquad (i,j) \in A \qquad (4)$$

$$x_{ij}^{p} \geq 0, \qquad (i,j) \in A, \ p \in P \qquad (5)$$

The objective function measures the total cost of the system.

Constraints (1) express the flow conservation and demand satisfaction requirements.

Constraints (2) are the capacity constraints that ensure flow does not exceed capacity.

Constraints (3) are some additional constraints τ such as budget constraints that indicate restrictions imposed upon resources shared by several (or all) links or the disaggregation constraints that are used for tighter formulation.

Constraints (4) and (5) indicate bounds on the decision variables.

4.3 Literature Review

This section presents a brief overview of the literature that we studied in particular to the MCFCNF problems, and various heuristic and exact methods that are suggested for solving these problems more effectively.

Network design problems are theoretically and practically significant class of problems. Practical applications of this problem have been presented in many papers such as [4, 9, 22, 23, and 28]. The service network design problems faced by less-than-truckload (LTL) freight transportation carriers and a solution approach for the huge instances that result when they are applied to large-scale LTL networks are presented in Crainic et al. (2000) [9]. The results presented in Ghamlouche et al. (2003) [19] indicate that the instances even with the experimental size (around 100 to 300 arcs), take a long time for finding the optimal solution. Realistically sized instances are larger than this experimental size of instances. Even though much research has been devoted to MCFCNF, exact methods are only capable of handling the small instances that are far smaller than many realistic-sized instances [15, 22]. Therefore, various heuristic methods such as tabu search method and the cut generation methods such as flow cover inequalities that were introduced in Padberg et al. (1985) [25] have been developed for improving the solution time for solving the MCFCNF problems.

As described in Ghamlouche et al. (2003) [19], meta-heuristics are used to find good primal solutions for the network design problems. An efficient procedure to find good feasible solutions using a tabu search metaheuristic for a difficult network optimization problem is proposed in Crainic et al. (2000) [11]. It also studies the relationships among the tabu search framework, simplex pivoting, and column generation. Some heuristic approaches such as cycle-based neighborhood structures, and tabu search algorithms using the neighborhood cycles that allow the rerouting of multiple commodities are presented in Ghamlouche et al. (2003) [19]. A new solution approach for the fixed-charge network flow (FCNF) problem that produces provably high-quality solutions quickly has been developed in Hewitt et al. (2010) [22]. This approach

combines mathematical programming algorithms with heuristic search techniques by using an IP technology for neighborhood search.

In the literature presented above, the exact methods for solving MCFCNF problems, such as cut generation and heuristic methods such as tabu search have been studied. However the branching rules that can play an important role in improving the performance have not been explored.

4.4 Mathematical Formulation

In this section, we present the arc-based formulation (as presented in Hewitt et al. (2010) [22]) for the MCFCNF problem. We use the decision variables x_{ij}^k to indicate the amount of commodity k routed from node i to node j in the network, and the binary variables y_{ij} to represent whether an arc (i,j) is installed in a network or not. Let C_{ij}^k denote the variable cost of routing commodity k on arc (i,j), let f_{ij} denote the fixed cost for installing an arc (i,j) in the network, and let u_{ij} denote the capacity of arc (i,j). Furthermore, we use A and N to represent the set of directed arcs and nodes in the network, respectively. K denotes the set of commodities that have certain demand r^k that must be routed from the source s(k) to the sink t(k). The commodities are distinguished according to the source and destination pairs. The objective function is to minimize the sum of fixed and variable costs.

Decision variables:

x_{ij}^k : Amount of commodity k routed from node i to node j in the network.

y_{ij} : This is a binary variable that represents whether an arc (i,j) is installed in a network.

$y_{ij} = 1$ if arc (i,j) is installed in a network,

$= 0$ otherwise.

Parameters:

C_{ij}^k : Cost of routing commodity k on arc (i,j)

f_{ij} : Fixed cost for installing an arc (i,j)

u_{ij} : Capacity of arc (i,j)

Objective Function:

Minimize $\sum_{k\in K}\sum_{j:(i,j)\in A} C_{ij}^{k}\,(r^{k}x_{ij}^{k}) + \sum_{j:(i,j)\in A} f_{ij}\,y_{ij}$

Subject to

$$\sum_{j:(i,j)\in A} x_{ij}^{k} - \sum_{j:(i,j)\in A} x_{ji}^{k} = b_{i}^{k} \qquad \forall\, i \in N, k \in K \qquad (1)$$

$$\sum_{k\in K} r^{k}x_{ij}^{k} \le u_{ij}y_{ij} \qquad \forall\, (i,j) \in A \qquad (2)$$

$$x_{ij}^{k} \le y_{ij} \qquad \forall\, (i,j) \in A,\ k \in K \qquad (3)$$

$$0 \le x_{ij}^{k} \le 1 \qquad \forall\, (i,j) \in A,\ k \in K \qquad (4)$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \qquad (5)$$

b_{i}^{k} = 1 if i is a source node

= -1 if i is a destination node

= 0 otherwise

Constraints (1) ensure flow balance, where b_{i}^{k} denotes if the node i in the network is a source node ($b_{i}^{k} = 1$) or a destination node ($b_{i}^{k} = 1$) for commodity k, or an intermediate node ($b_{i}^{k} = 0$).

Constraints (2) are the capacity constraints that ensure that the total flow on the arc does not exceed its capacity. These constraints are also used as the coupling constraints that ensure that an arc is used if and only if its fixed charge is paid.

Constraints (3) are the disaggregation constraints. These are the valid inequalities which are used to make the mixed integer programming formulation stronger by reducing the size of the feasible

region. The resulting formulation has a tighter LP relaxation at the expense of adding many constraints.

Constraints (4) indicate the variables are positive and continuous. This represents that the flow of any commodity over arc (i,j) is always positive.

Constraints (5) indicate that the y_{ij}variables are binary variables. This states that arc (i,j) will be either installed in a network if y_{ij} is 1, or it will not be installed in a network if y_{ij} is 0.

4.5 Variants of MCFCNF Problem

The mathematical formulation presented above in the section 4.4 contains binary y_{ij} variables that represent whether an arc (i,j) is installed in a network, and the continuous x_{ij}^{k} variables that indicate the fraction of the flow of a commodity k on a particular arc (i,j). As the flow variables x_{ij}^{k} are continuous, the flow of a commodity can be distributed over several paths.

Practical applications of this variant include the road transportation problems, where a commodity is identified by its type. In this case, a commodity can be distributed over several paths. Both arc based and path based mathematical formulations of this variant of MCFCNF are presented in Gendron et al. (1994) [17].

Along with this formulation, we have also studied 3 other variants of the MCFCNF problem as presented below. The following formulations vary from each other in the way the bounds are applied on the y_{ij} and x_{ij}^{k} variables.

4.5.1 Binary Arc Variables, Binary Flow Variables:

In this formulation, the binary y_{ij} variables indicate whether an arc (i,j) is installed in a network, and the binary x_{ij}^{k} variables indicate whether the commodity k is routed on an arc (i,j). Thus, a commodity can utilize only a single arc, and its flow will not be distributed over several arcs.

The formulation will be similar to that mentioned above in section 4.4, with the only difference in the bounds on the x_{ij}^{k} variables (Thus, the constraint (4) will be replaced by constraint (6) presented below).

$$x_{ij}^{k} \in \{0, 1\} \qquad \forall (i, j) \in A \qquad (6)$$

The practical applications of this variant include the railroad transportation [5] and truck transportation problems [15] where commodities do not split path. The arc-based and path-based formulations for this variant of MCFCNF are presented in Hewitt et al. (2010) [22]. The splittable (continuous flow variables) and unsplittable (binary flow variables) network design problems have been studied in detail in Atumturk et al. (2002) [2]. As noted here, flow of a commodity is restricted to run through a single path along the network in many applications such as production–distribution with single sourcing, and express package delivery. Further references for these applications can be found in Atumturk et al. (2002) [2].

4.5.2 Integer Arc Variables, Fractional Flow Variables:

This formulation allows multiple arcs between nodes i and j. The formulation is similar to that presented in section 4.4, with the only difference in the bounds of the y_{ij} variables. The y_{ij} variables are bounded to take integral values that represent number of arcs between nodes i and j. The flow of a commodity can be distributed over multiple paths.

Network design problems where multiple capacities of a single arc are installed in the network are discussed in Atumturk et al. (2002) [2]. Practical applications of the variants with integer arc variables include truck transportation applications, railroad applications for determining the number of engines to power a set of trains on a railroad network as well as telecommunication applications for installing or leasing fiber optic cables on a telecommunication network Atumturk et al. (2002) [2].

4.5.3 Integer Arc Variables, Binary Flow Variables:

This formulation also allows multiple arcs between nodes i and j. The y_{ij} variables are bounded to take integral values instead of binary. There can be multiple arcs installed between two nodes and the flow of a commodity is not distributed over several paths. Practical applications for this variant include freight transportation problems [9, 15].

An efficient heuristic approach that can easily handle different variants of the problem as presented above has been proposed in Hewitt el al. (2010) [22]. We now present the naming conventions that we have used for these variants of network design problem.

4.5.4 Naming Conventions Used for Representing the Variants

We use the following naming convention in this document for representing the variants listed above in this section. The variants are named as 'Variant_X_Y', where X represents whether the y_{ij} variables are binary or integral, and Y represents whether the x_{ij}^k variables are continuous or binary.

Table 2: Naming convention for the variants of the MCFCNF problem

Variant Name	y_{ij} variables	x_{ij}^k variables
Variant_B_C	Binary	Continuous
Variant_B_B	Binary	Binary
Variant_I_C	Integer	Continuous
Variant_I_B	Integer	Binary

5. Computational Results for the Existing Variable Selection Methods

This section presents the computational results for the existing branching rules on several instances for multiple variants of the MCFCNF problem. All calculations were performed on a Linux server. Each data set is tested for all the branching rules studied. The time limit for solving MIP is set to 4 hours, and the MIP gap tolerance is set to 1%. Different branching rules find high quality primal solutions at different rates. Hence, we used a heuristic primal solution to provide a better primal bound initially, and the performance on improving the dual bound is tested.

5.1 Test Sets

We have used a group of 20 randomly generated instances of the Multi-Commodity Fixed Charge Network Flow problems (MCFCNF) that are developed by Bernard Gendron [17, 18] and used in several publications [10, 11, 22] to test relaxations and exact or heuristic approaches to the problem.

Instances with different characteristics are tested with the time limit of 4 hours. Primal heuristic solution is provided while solving the problem using branch and bound based algorithm. As we branch on the arc variables that are associated with fixed costs in the objective function, we have used the instances that have dominant fixed costs. For testing branching rules on the datasets with different characteristics, we used a group of 20 instances, divided in two classes, one with loose capacity constraints and the other with tight capacity constraints. Size of the experimental instances that we selected varies from 10 nodes and 60 arcs to 20 nodes and 315 arcs as indicated below in Table 3.

The first column in Table 3 indicates the name of the instance. The second column (N, A, K) represents number of nodes, arcs, and commodities respectively in a particular instance. The instances have a naming scheme Rx_y, where x and y have the following meaning: The third column indicates ‘Fixed cost rank’ represented by ‘x’, and the ‘Capacity rank’ that is represented by ‘y’. The fixed cost rank can be selected as 1, 5, or 10. Higher fixed cost rank indicates that the

fixed costs are more dominant as compared to the variable costs. We have selected the instances with fixed cost rank 10, with the dominant fixed costs.

Furthermore, the capacity rank indicates whether the capacity constraints are tight or loose. Higher capacity rank indicates the tight capacity constraints.

Table 3 : Naming convention used for the tested datasets

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)
r06_6	10,60,50	F,L (10,2)
r07_6	10,82,10	F,L (10,2)
r08_6	10,83,25	F,L (10,2)
r09_6	10,83,50	F,L (10,2)
r10_6	20,120,40	F,L (10,2)
r14_6	20,220,100	F,L (10,2)
r15_6	20,220,200	F,L (10,2)
r16_6	20,314,40	F,L (10,2)
r17_6	20,318,100	F,L (10,2)
r18_6	20,315,200	F,L (10,2)
r06_9	10,60,50	F,T (10,8)
r07_9	10,82,10	F,T (10,8)
r08_9	10,83,25	F,T (10,8)
r09_9	10,83,50	F,T (10,8)
r10_9	20,120,40	F,T (10,8)
r14_9	20,220,100	F,T (10,8)
r15_9	20,220,200	F,T (10,8)
r16_9	20,314,40	F,T (10,8)
r17_9	20,318,100	F,T (10,8)
r18_9	20,315,200	F,T (10,8)

We have used the mathematical solvers SCIP [30] and GLPK [21] for these experiments. The experimental set-up and computational results with SCIP solver are presented in section 5.3. SCIP is one of the fastest non-commercial mixed integer programming solvers [30]. Furthermore, being open-source software, we get access to the source code for studying the implementation of existing branching rules. Hence, we have used SCIP for our analysis of the existing branching rules and the solution based approach. In order to evaluate the influence of SCIP solver on the computational results, we have also tested the existing branching rules for Variant_B_C with GLPK solver. We have presented these results with GLPK in Appendix II.

5.2 Experimental Setup

SCIP is currently one of the fastest non-commercial open source mixed integer programming (MIP) solvers [30]. SCIP provides a framework for implementing mixed integer programming problems, and for studying various branching strategies through the built-in methods. Hence, we chose SCIP as the primary solver for all the experimentations for this analysis.

As noted in Achterberg et al. (2002) [1], "The complex interrelations between cutting plane generation, primal heuristics, node selection, and branching variable selection makes benchmarking branching strategies difficult". The behavior of the branch-and-bound algorithm can be altered significantly by changing the parameter settings that control these components. Through extensive experimentation, integer-programming software vendors have determined the default settings that work well for most instances encountered in practice [3]. Hence, we have used the default parameter settings for our comparison of the branching strategies. We calculated the optimal solution values for all the instances beforehand and provided those as primal heuristics, since leading towards the feasible solutions fast is a desirable property of branching rules.

All the instances are solved with SCIP with the time limit of 4 hours and MIP gap limit of 1%. The primal heuristic solution is initially calculated for each dataset and for all the variants by

executing these codes with 30 hours of time limit. Out of total 20 instances, 14 instances with Variant_B_C, 11 instances with Variant_B_B, and 12 instances with Variant_I_C and 12 with Variant_I_B are solved to optimality within this time limit. All the parameters except the time and MIP gap limits are set to their default values. Any additional functionality such as branching rules, variable data, separators (cutting planes) that are added while solving MIP are added as plug-ins for the SCIP solver.

In SCIP, the separators are used to generate general purpose cutting planes [30]. Separators generate valid inequalities from a specific constraint class or a subset of the constraints of a single constraint class. In contrast, general purpose cuts do not require or exploit any knowledge about the underlying problem structure but use only the current LP relaxation and the integrality conditions [30]. SCIP provides the built-in separator for the MCFCNF problems and we can add these as plug-in to the main formulation. We tested different formulations of the network design problem such as formulations with and without disaggregation constraints, and with and without separators. The performance of the existing and new branching rules with respect to solution time and solution quality has been analyzed for different variants of the network design problem as described in the section 4.5. We further analyzed the effect of quality and quantity of feasible solutions for the new branching rule.

Parameter Settings:

The branching rules such as reliability branching and strong branching require parameter values. For the SCIP experiments, we have used all the default settings of these parameters. The following table lists these parameters and their values set in the SCIP solver for these branching rules.

Table 4 : Parameter settings in SCIP

Branching Rule	Parameters	Values	Description
Reliability Branching	lambda	8	maximal number of further variables evaluated without better score
	DEFAULT_MINRELIABLE	1	minimal value for minimum pseudo cost size to regard pseudo cost value as reliable
	DEFAULT_MAXRELIABLE	8	maximum value for minimum pseudo cost size to regard pseudo cost value as reliable
	etarel - Reliability parameter	Calculated dynamically at the run-time	Reliability parameter is calculated dynamically with the weighted sum of the DEFAULT_MINRELIABLE And DEFAULT_MINRELIABLE values.
Strong Branching	DEFAULT_INITCAND	100	maximal number of candidates initialized with strong branching per node

5.3 Computational Results

In this section, we present the computational results for the existing branching rules. The results presented below are organized as follows. We initially present the results with the SCIP solver for analyzing the existing branching rules, and determine the best branching strategy for all the 4 variants of the network design problem. In order to understand how significant the influence of the SCIP solver is, we further analyzed Variant_B_C with GLPK. The results of the existing branching rules tested with GLPK are presented in Appendix II.

5.3.1 Description of Tables

Tables 5 to 7 present the computational results with SCIP for the existing branching rules for Variant_B_C in detail. We present SCIP results for all the variants in appendix I. First column in these tables is the name of the data-set. Second column represents number of nodes, arcs, and

commodities (N, A, K). Third column indicates the type of the instance in the form (fixed cost rank, and capacity rank). Higher the fixed cost rank, higher is the dominance factor of the fixed costs over the variable costs. Higher capacity ranks indicate that the capacity constraints are tight. All the data-sets that we tested have dominant fixed cost. We tested the instances with both loose and tight capacity constraints.

For the results with SCIP presented in Tables 5 to 7 and appendix I, we report the geometric mean and variance of the values in each column of the tables. Since all the instances are not solved to 1% MIP Gap limit, we also report the number of instances that are solved to the specified 1% MIP gap limit (NumOpt), and the average time and MIP gap for the instances that are not solved to 1% MIP gap limit (NonOpt_Avg) in Appendix I.

Table 5 : Time in seconds (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time in seconds	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching
r06_6	10,60,50	F,L (10,2)		487.54	253.62	2,524.59	3,891.32	233.74	1,254.42	138.36
r07_6	10,82,10	F,L (10,2)		0.39	0.47	0.65	0.69	0.65	0.65	0.65
r08_6	10,83,25	F,L (10,2)		5.62	6.66	10.64	9.22	8.85	7.10	8.60
r09_6	10,83,50	F,L (10,2)		2,274.02	1,521.54	5,455.43	2,645.11	465.14	2,547.86	265.94
r10_6	20,120,40	F,L (10,2)		3,878.25	1,343.92	229.62	169.74	86.30	141.50	146.23
r14_6	20,220,100	F,L (10,2)		14,402.30	14,402.60	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_6	20,220,200	F,L (10,2)		14,412.60	14,411.10	14,400.00	14,400.00	14,400.00	14,401.20	14,400.00
r16_6	20,314,40	F,L (10,2)		14,400.10	14,400.20	14,400.00	14,400.00	6,172.02	14,400.00	2,603.21
r17_6	20,318,100	F,L (10,2)		14,403.80	14,402.50	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_6	20,315,200	F,L (10,2)		14,415.70	14,406.40	14,400.00	14,400.00	14,400.00	14,400.00	14,401.40
r06_9	10,60,50	F,T (10,8)		15.95	16.39	22.41	20.75	16.78	17.23	13.13
r07_9	10,82,10	F,T (10,8)		22.09	12.08	97.95	87.78	22.17	33.39	8.57
r08_9	10,83,25	F,T (10,8)		200.30	88.81	1,472.28	2,429.41	45.95	608.37	40.93
r09_9	10,83,50	F,T (10,8)		22.98	11.34	29.22	138.33	13.35	87.35	12.01
r10_9	20,120,40	F,T (10,8)		1,004.26	241.26	14,400.00	7,251.00	154.61	4,031.43	74.22
r14_9	20,220,100	F,T (10,8)		14,400.20	14,401.10	14,400.00	14,400.00	14,400.00	14,400.00	12,153.90
r15_9	20,220,200	F,T (10,8)		78.09	80.86	90.20	94.44	121.67	96.84	118.27
r16_9	20,314,40	F,T (10,8)		14,400.00	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r17_9	20,318,100	F,T (10,8)		14,402.40	14,402.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_9	20,315,200	F,T (10,8)		14,405.30	14,407.80	14,400.00	14,400.30	14,401.70	14,401.20	14,401.60
			Time in sec: Geometric Mean	841.90	647.64	1,237.92	1,272.56	526.16	979.68	435.18
			Variance	7,040.64	7,197.30	6,985.72	6,790.82	7,058.05	7,007.19	6,992.24
			NonOpt_Avg	10,153.53	9,787.19	10,114.90	9,939.40	8,016.83	10,074.55	8,638.24

Table 6 : MIP Gap (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	MIP Gap	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching
r06_6	10,60,50	F,L (10,2)		0.88	0.99	1.00	1.00	1.00	1.00	1.00
r07_6	10,82,10	F,L (10,2)		0.09	0.09	0.09	0.09	0.09	0.09	0.09
r08_6	10,83,25	F,L (10,2)		1.00	0.80	0.90	0.82	0.94	0.63	0.98
r09_6	10,83,50	F,L (10,2)		0.83	0.92	1.00	1.00	1.00	1.00	1.00
r10_6	20,120,40	F,L (10,2)		0.76	0.76	0.99	0.99	1.00	0.97	0.97
r14_6	20,220,100	F,L (10,2)		7.03	6.80	5.39	4.51	5.35	4.58	4.36
r15_6	20,220,200	F,L (10,2)		7.68	7.35	6.84	7.56	7.48	7.68	7.37
r16_6	20,314,40	F,L (10,2)		5.26	3.65	2.69	1.63	1.00	1.52	1.00
r17_6	20,318,100	F,L (10,2)		8.72	8.62	5.72	5.28	5.09	5.10	5.06
r18_6	20,315,200	F,L (10,2)		5.35	5.09	4.52	4.75	5.01	4.82	5.35
r06_9	10,60,50	F,T (10,8)		0.80	0.97	0.96	0.98	0.98	1.00	0.97
r07_9	10,82,10	F,T (10,8)		0.86	0.76	1.00	1.00	0.99	0.99	0.83
r08_9	10,83,25	F,T (10,8)		0.91	0.87	1.00	1.00	0.99	1.00	0.95
r09_9	10,83,50	F,T (10,8)		0.79	0.96	0.97	1.00	0.91	1.00	0.98
r10_9	20,120,40	F,T (10,8)		0.97	0.97	1.84	1.00	0.99	1.00	0.93
r14_9	20,220,100	F,T (10,8)		2.72	1.83	3.67	3.34	1.61	3.28	0.98
r15_9	20,220,200	F,T (10,8)		0.56	0.56	0.56	0.56	0.56	0.56	0.56
r16_9	20,314,40	F,T (10,8)		7.10	6.63	5.62	5.08	4.58	4.81	3.87
r17_9	20,318,100	F,T (10,8)		5.65	5.42	5.69	4.68	4.76	5.01	4.05
r18_9	20,315,200	F,T (10,8)		4.88	4.38	4.60	4.50	5.64	5.60	3.84
			MIP Gap: Geometric Mean	1.77	1.70	1.78	1.64	1.57	1.63	1.45
			Variance	2.94	2.78	2.23	2.17	2.28	2.26	2.08
			NumOpt	11.00	11.00	10.00	11.00	12.00	11.00	13.00
			NonOpt_Avg	6.04	5.53	4.66	4.96	4.94	4.71	4.84

Table 7 : Total Solving Nodes (SCIP formulation for Variant_B_C with disaggregation constraints and with cutting planes)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Total Solving Nodes	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching
r06_6	10,60,50	F,L (10,2)		543.00	691.00	152,741.00	189,101.00	9,741.00	89,631.00	4,851.00
r07_6	10,82,10	F,L (10,2)		1.00	1.00	1.00	1.00	1.00	1.00	1.00
r08_6	10,83,25	F,L (10,2)		1.00	2.00	681.00	197.00	94.00	171.00	2.00
r09_6	10,83,50	F,L (10,2)		870.00	1,249.00	645,941.00	79,621.00	13,621.00	149,111.00	6,541.00
r10_6	20,120,40	F,L (10,2)		217.00	194.00	4,171.00	631.00	411.00	1,181.00	691.00
r14_6	20,220,100	F,L (10,2)		17.00	45.00	27,243.00	8,016.00	9,235.00	13,553.00	6,714.00
r15_6	20,220,200	F,L (10,2)		1.00	2.00	1,103.00	57.00	180.00	1.00	100.00
r16_6	20,314,40	F,L (10,2)		56.00	459.00	220,994.00	87,412.00	23,751.00	124,033.00	6,261.00
r17_6	20,318,100	F,L (10,2)		2.00	10.00	21,165.00	5,863.00	7,868.00	10,942.00	5,438.00
r18_6	20,315,200	F,L (10,2)		1.00	2.00	435.00	236.00	228.00	214.00	1.00
r06_9	10,60,50	F,T (10,8)		6.00	29.00	1,211.00	461.00	201.00	504.00	24.00
r07_9	10,82,10	F,T (10,8)		67.00	181.00	61,211.00	27,181.00	5,961.00	9,871.00	765.00
r08_9	10,83,25	F,T (10,8)		281.00	411.00	301,151.00	226,241.00	2,671.00	81,091.00	2,281.00
r09_9	10,83,50	F,T (10,8)		16.00	15.00	552.00	5,332.00	122.00	2,092.00	19.00
r10_9	20,120,40	F,T (10,8)		423.00	372.00	602,833.00	171,881.00	1,951.00	115,931.00	275.00
r14_9	20,220,100	F,T (10,8)		94.00	234.00	22,749.00	9,660.00	9,553.00	8,630.00	19,939.00
r15_9	20,220,200	F,T (10,8)		1.00	1.00	1.00	1.00	1.00	1.00	1.00
r16_9	20,314,40	F,T (10,8)		87.00	243.00	119,241.00	35,656.00	38,534.00	51,748.00	50,791.00
r17_9	20,318,100	F,T (10,8)		7.00	52.00	1,635.00	3,781.00	756.00	758.00	3,030.00
r18_9	20,315,200	F,T (10,8)		1.00	2.00	459.00	401.00	1.00	1.00	53.00
			Solving Nodes: Geometric Mean	17.64	36.70	5,256.22	2,554.81	622.75	1,340.66	300.34
			Variance	231.59	313.36	195,256.89	71,244.99	9,828.66	49,779.98	11,680.36

A summary of the computational results with SCIP for the existing branching rules for all the 4 variants of the network design problem is presented in Tables 8 to 10. Labels of the columns represent names of the branching rules tested with SCIP and rows indicate the results for different variants of the problem. Values in Tables 8, 9, and 10 represent the geometric mean of the time, MIP gap and total number of solving nodes over all the tested instances respectively.

Tables 8, 9 and 10 present the results with SCIP for the formulation with disaggregation constraints and with cutting planes. The geometric mean of the solution time and MIP gap are calculated for comparing the performance of the branching rules. We conclude from these results that the reliability branching method outperforms all the existing branching rules. The most infeasible, least infeasible and the random branching rules do not perform well for any variant of the network design problem. This observation is in accordance with our conclusions drawn from the GLPK experiments. The 'Full strong branching' method (that limits the number of candidate variables in strong branching), the 'Full strong^{+} branching' method that is a 'Full strong branching' with all variables (referred to as 'all full strong branching' in SCIP documentation and source code [30]) and the pseudo-cost branching method also perform remarkably well next to the reliability branching method.

Table 8: Time in seconds (SCIP formulation with disaggregation constraints and with cutting planes)

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-Cost Branching	Random Branching	Reliability Branching
Variant_B_C	841.90	647.64	1,237.92	1,272.56	526.16	979.68	**435.18**
Variant_B_B	560.29	434.51	835.61	758.47	315.33	806.50	**261.66**
Variant_I_C	1,040.43	1,094.93	1,633.09	1,484.48	801.77	1,387.55	**636.28**
Variant_I_B	536.05	492.09	749.05	646.04	331.40	678.56	**286.03**

Table 9: MIP Gap (SCIP formulation with disaggregation constraints and with cutting planes)

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-Cost Branching	Random Branching	Reliability Branching
Variant_B_C	1.77	1.70	1.78	1.64	1.57	1.63	**1.45**
Variant_B_B	2.39	1.89	2.65	2.18	1.79	2.27	**1.72**
Variant_I_C	2.06	1.69	2.01	1.84	1.69	1.85	**1.49**
Variant_I_B	2.38	2.12	2.62	2.39	2.16	2.32	**2.12**

Table 10: Average number of branch and bound nodes (SCIP formulation with disaggregation constraints and with cutting planes)

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-Cost Branching	Random Branching	Reliability Branching
Variant_B_C	**17.64**	36.70	5,256.22	2,554.81	622.75	1,340.66	300.34
Variant_B_B	**3.22**	8.20	2,561.64	1,883.58	730.10	2,481.22	331.18
Variant_I_C	**39.61**	141.73	16,364.82	4,549.57	2,989.70	5,833.92	1,399.99
Variant_I_B	**2.97**	15.30	2,632.77	1,552.06	717.04	1,901.38	378.15

When we consider the geometric mean of the total number of branch and bound nodes needed to solve the problem instances, strong branching that considers all the candidate variables (Full strong^{+} branching) requires the least number of branch and bound nodes followed by the strong branching method that limits number of candidate variables (Full strong branching). Reliability branching is in the third position followed by the pseudo-cost branching. The least infeasible, most infeasible and the random branching perform the worst with respect to the number of branch and bound nodes needed to solve the problem instances. These observations are in accordance with the analysis presented in Achterberg et al. (2002) [1].

We now introduce the new solution based approach for branching rule in the next section. While developing the new branching rule, we performed series of experiments for selecting a branching variable that corresponds to the arc with minimum, maximum, most fractional, and least fractional average utilization. Further experimentations are performed for testing the effect of quality and quantity of feasible solutions on the performance of the solution based approach.

6. Solution Based Approach for Variable Selection

6.1 Motivation

From the computational results presented above, we can observe that there is a tradeoff between the time spent per node and the number of nodes needed to solve the problem. Strong branching rule requires the least number of nodes. However, this method shows average performance among the leading branching rules because the processing time per node is very high. Reliability branching rule is the winning branching rule. It requires less number of branch and bound nodes, and processing time per node is also less as compared to strong branching.

Hence, as noted in Atumturk et al. (2005) [3], the ultimate goal is to find a fast branching strategy that minimizes the number of branch and bound nodes that need to be evaluated, and requires less processing time for evaluating each node. This motivates our idea of using the information contained in a pool of known feasible solutions for designing a new branching rule. Unlike most of the traditional branching rules that require high processing time for calculating LP relaxation for the child nodes, this solution based approach does not have computational burden at each node.

During the development phase of our new branching rule, we initially analyzed some statistics collected for the existing branching strategies. In branch and bound based methods, nodes on the frontier of a branch and bound search tree have large amount of uncertainty about variable values [20]. Hence, the branching variables selected in the initial levels of a branch and bound tree can significantly affect the performance of branch and bound algorithm. Based on this observation, we collected first five branching variables for all the data-sets for the existing branching strategies. We further studied if these initial branching variables are related to the structure of the network. For example, we analyzed if the arcs corresponding to these branching variables originate from any source node or terminate on a destination node. We also noted the variable that is branched on most of the times in a branch and bound tree, and studied if there is any particular characteristic of the arc associated with this variable. We later collected utilizations of the first 5 branching variables at the time of branching, and studied if there is any pattern in those

values. For example, we analyzed if the initial branching variables correspond to the least utilized or most utilized arcs.

Next phase of our analysis was directed towards finding the difference between variables selection for leading existing branching rules (such as reliability branching) and the branching rules that do not perform well (such as most fractional branching). However, the results from this study did not suggest a pattern or lead to any definitive conclusions.

Utilization of an arc is defined as flow per unit capacity of that arc (Utilization = Total flow / Capacity). One key observation that directed us towards our new technique is that the values of the arc variables in LP relaxation at the current branch and bound node are current utilization values of those arcs. Furthermore, utilization of an arc affects the remaining flow because of the capacity constraints. And hence, utilizations play major role in deciding whether a new arc needs to be installed in the network or not. Utilization values may play significant role in branching, however, branching based on LP solutions may not be as good as IP solutions. Hence we considered using the average utilization for the arcs calculated from the set of known feasible solutions for that instance. We further tested various methods of assigning the scores to candidate variables. For example, we analyzed the solution based branching rule by selecting a branching variable with the least average utilization, most average utilization, least fractional average utilization, and most fractional average utilization. However, we observed that the methods that selecting branching variable with most average utilization, most fractional average and least average utilization do not perform well. Hence, our solution based branching rule selects a branching variable with least average utilization.

We now present the detailed description of the new solution based branching rule below.

6.2 Solution Based Branching Rule

The branching rule that we developed considers the use of the information contained in a set of known feasible solutions. Suppose that a set of feasible solutions is available for each of the instances. The utilization and average utilization of each arc can be calculated from the available pool of feasible solutions as described below.

For the variants of the network design problem that have binary arc variables (Variant_B_C and Variant_B_B),

$$\textbf{Utilization of arc } (\mathbf{i},\mathbf{j}) = \frac{\left(\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{D_k x_{ij}^k}\right)}{\boldsymbol{U_{ij}}}$$

$$\textbf{Average utilization for each arc} = \frac{\sum_{\mathbf{f \in F}} \frac{\left(\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{D_k x_{ij}^k}\right)}{\boldsymbol{U_{ij}}}}{\mathbf{n}}$$

Where, U_{ij} indicates the capacity of arc (i,j) and D_k is the demand of commodity k. The x_{ij}^k values indicate the fraction of a commodity k on arc (i,j) and these values are collected from the set of known feasible solutions. [Please note that the x_{ij}^k values do not represent the values from the solution at the current branch and bound node, and are collected for each feasible solution from the known solution pool]. F is a set of feasible solutions such that F = [f1, f2, f3, …. , fn] , n represents the total number of feasible solutions. The numerator value in the calculation of average utilization represents the sum of the utilizations of an arc over all the feasible solutions [f1, f2, f3, …. , fn] in set F.

For Variant_I_C and Variant_I_B, multiple arcs can be installed between the nodes i and j. In this case, the y_{ij} variables can have integral values in the available feasible solutions. Hence, the utilization and the average utilization of an arc for these variants will be calculated as:

$$\textbf{Utilization of the arc (i,j)} = \frac{\left(\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{D_k\, x_{ij}^k}\right)}{\mathbf{yij} * \boldsymbol{U_{ij}}}$$

$$\textbf{Average utilization for each arc} = \frac{\sum_{\mathbf{f \in F}} \frac{\left(\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{D_k x_{ij}^k}\right)}{\mathbf{yij} * \boldsymbol{U_{ij}}}}{\mathbf{n}}$$

For the variants of network design problem with integer arc variables (Variant_I_C and Variant_I_B), we have also tested the approach (mentioned below as ‘Variant_I_C_Approach2’ and ‘Variant_I_B_Approach2’) of calculating average utilization of an arc considering the flow on the last copy of that arc. For these variants, the y_{ij} variables can have integral values in the

available feasible solutions. Furthermore, the last copy of the arc (i,j) is incompletely utilized. Hence, the utilization of an arc for these variants is calculated for the last copy of the arc. The numerator value in the following formula indicates the flow on the last copy of the arc (i,j).

$$\textbf{Utilization of the arc (i,j)} = \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right) \% U_{ij}}{U_{ij}}$$

$$\textbf{Average utilization for each arc} = \frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right) \% U_{ij}}{U_{ij}}}{n}$$

The following table summarizes the calculation of the average utilizations for all the 4 variants of the MCFCNF problem:

Table 11 : Formulae for calculating average utilization for the variants of the network design problem

Variant	Average Utilization
Variant_B_C	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right)}{U_{ij}}}{n}$
Variant_B_B	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right)}{U_{ij}}}{n}$
Variant_I_C	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right)}{y_{ij} * U_{ij}}}{n}$
Variant_I_B	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right)}{y_{ij} * U_{ij}}}{n}$
Variant_I_C_Approach2	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right) \% U_{ij}}{U_{ij}}}{n}$
Variant_I_B_Approach2	$\frac{\sum_{f \in F} \frac{\left(\sum_{k=1}^{K} D_k x_{ij}^k\right) \% U_{ij}}{U_{ij}}}{n}$

6.3 Algorithm for Selecting a Branching Variable in the Solution Based Branching Rule

We tested the solution based approach by selecting a branching variable that corresponds to the arc with the least average utilization, as well as with the least fractional average utilization. The algorithms for selecting a branching variable for these methods are presented below.

While presenting these algorithms, we follow the same conventions that are used in section 3 for describing the existing variable selection methods.

Selecting a branching variable corresponding to the arc with least average utilization:

For each i ∈ candidates:

1. score(xi) = Average utilization of each arc
2. i* = arg {min i∈candidates{score(x_i)}}
3. Return i∗

Selecting a branching variable corresponding to the arc with least fractional average utilization:

For each i ∈ candidates:

1. avguti = Average utilization of each arc
2. score (x_i) ← avguti – floor (avguti) for avguti ≤ 0.5

   ← ceil (avguti) – avguti for avguti > 0.5
3. i* = arg {min i∈candidates{score(x_i)}}
4. Return i∗

6.4 Hybrid Method for the Variants with Binary Flow Variables

In the variants where the flow variables x_{ij}^k are binary (Variant_B_B and Variant_I_B), the candidate branching variables may include both x_{ij}^k and y_{ij} variables. In this case, only y_{ij} variables are considered as candidate variables, and the solution based approach is used for selecting the branching variable from these candidate variables. If the candidate variables include only the flow variables x_{ij}^k, the reliability branching method is applied for branching.

A generalized algorithm for variable selection using the solution based approach can be presented as follows:

6.5 Algorithm for the Solution Based Approach

Algorithm 3: Solution based approach for the new branching rule

Input: A set of feasible solutions.
Output: Index of the variable to be branched on next.

1. Determine the set of candidate variables for branching.
2. For each candidate variable, calculate the average utilization using a set of available feasible solutions that are available as input.
3. Assign the scores for each candidate variable as described in section 6.2
4. Return the index of the candidate variable with least score.

7. Computational Results for the Solution Based Approach

In this section, we present the computational results with SCIP for the new solution based approach. While presenting these results, we also compare the results of the leading existing branching rules with the new solution based approach for our analysis.

We collected the feasible solutions with SCIP with the default parameter settings without specifying any MIP gap limit. We used these feasible solutions in the experiments for comparing the performance of the solution based approach with the leading existing branching rules. The following table presents the quality of feasible solutions that we used for all the 4 variants of the network design problem. For each of the variants, we present average MIP gap for the collected feasible solutions.

Table 12: Quality of feasible solutions collected with SCIP (%MIP Gap)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity Rank)	Variant_B_C	Variant_B_B	Variant_I_C	Variant_I_B
r06_6	**10,60,50**	**F,L (10,2)**	21.225	12.125	**> 500**	1.055
r07_6	**10,82,10**	**F,L (10,2)**	21.19	106.16	**> 500**	111.32
r08_6	**10,83,25**	**F,L (10,2)**	20.16	88.59	**> 500**	107.04
r09_6	**10,83,50**	**F,L (10,2)**	4.685	33.65	**> 500**	38.565
r10_6	**20,120,40**	**F,L (10,2)**	26.35	14.36	**> 500**	30.195
r14_6	**20,220,100**	**F,L (10,2)**	70.77	63.08	**> 500**	70.675
r15_6	**20,220,200**	**F,L (10,2)**	39.16	123.69	**> 500**	118.53
r16_6	**20,314,40**	**F,L (10,2)**	83.69	128.49	**> 500**	132.735
r17_6	**20,318,100**	**F,L (10,2)**	100.105	119.42	**> 500**	98.815
r18_6	**20,315,200**	**F,L (10,2)**	79.93	93.505	**> 500**	84.25
r06_9	**10,60,50**	**F,T (10,8)**	5.225	17.675	**> 500**	23.905
r07_9	**10,82,10**	**F,T (10,8)**	27.92	-	**> 500**	-
r08_9	**10,83,25**	**F,T (10,8)**	16.93	33.93	**> 500**	46.05
r09_9	**10,83,50**	**F,T (10,8)**	17.035	37.055	**> 500**	13.92
r10_9	**20,120,40**	**F,T (10,8)**	6.235	47.645	**> 500**	4.835
r14_9	**20,220,100**	**F,T (10,8)**	26.865	17.86	**> 500**	49.47
r15_9	**20,220,200**	**F,T (10,8)**	18.25	13.17	**> 500**	43.44
r16_9	**20,314,40**	**F,T (10,8)**	89.31	111.955	**> 500**	150.88
r17_9	**20,318,100**	**F,T (10,8)**	84.45	50.93	**> 500**	80.98
r18_9	**20,315,200**	**F,L (10,8)**	43.58	20.08	**> 500**	39.23
		Average solution quality	40.80	59.65	>500	65.57

For analyzing the effect of quality and quantity of feasible solutions on the solution based approach, we collected the feasible solutions by specifying the MIP gap limit for the solutions using the experimental setup presented below in section 7.1.

7.1 Experimental Setup for Collecting the Pool of Feasible Solutions with SCIP

The pool of feasible solutions that is required for analyzing the solution based branching method has been collected using the methods available in the SCIP libraries.

We have also studied the effect of the quality and quantity of these feasible solutions on the performance of the branching rule. For testing the effect of quality and quantity of available feasible solutions the feasible solutions with the specified MIP gap window are collected by specifying the corresponding lower and upper bounds on the objective function. We first solved all the instances to optimality, and then calculated corresponding lower and upper bounds. Quality and quantity for feasible solutions are adjusted using parameter values as shown below:

$$\text{Bound}_{\text{Low}} = \alpha * z_{\text{opt}}$$
$$\text{Bound}_{\text{Hi}} = \beta * z_{\text{opt}}$$

Here, α and β are the parameters that specify the lower and upper MIP gap limits that indicate the quality of feasible solutions. z_{opt} represents the optimal solution value calculated for a particular instance.
Hence, the required constraint that is added to our main formulation is as mentioned below:

$$\text{Bound}_{\text{Low}} \leq \text{Objective Function} \leq \text{Bound}_{\text{Hi}}$$

The feasible solutions are then collected for this formulation for studying the effect of the quality of the solution-pool by selecting the required parameter values for quality of feasible solutions (MIP gap limit), and quantity of feasible solutions (number of feasible solutions).

Average utilization of the arcs is calculated using a set of known feasible solutions as described in section 6. This average utilization is required to assign the score to the candidate variables while implementing the solution based approach. Hence, we associate this average utilization to the variable corresponding to the particular arc at the time of variable creation [Appendix IV] and later use this data while implementing solution based branching rule.

7.2 Results for the Solution Based Approach for Variable Selection

Tables 13 and 14 present the leading existing branching rules with SCIP as analyzed in section 5 along with the new solution based approach with least average utilization. We consider the geometric mean of both the time and MIP gap over all the tested instances for determining the winning branching rule. As we were unable to collect the feasible solutions for the instance R07_9 and R15_6 for the Variant_B_B and Variant_I_B, we calculated the geometric mean over 18 instances for these two variants. For the remaining two variants, the geometric mean over all the 20 instances is presented for all the branching rules. Reliability branching method is a clear winner followed by the pseudo-cost branching rule for all the variants of the network design problem.

The performance of the solution based approach that we developed is in the third position for the variants Variant_B_C, Variant_B_B, and Variant_I_B. For Variant_I_C, the 'Full strong branching' method that limits the number of candidate variables outperforms the solution based approach. The 'Full strong^{+} branching' method comes in the last position (among the 5 leading branching rules) for all the variants.

Table 13: Time in seconds and MIP Gap: Leading existing branching rules and the solution based approach

Branching Rules		Full Strong^{+} Branching	Full Strong Branching	Pseudo-Cost Branching	Reliability Branching	Solution based approach
Variant_B_C	Time	841.90	647.64	526.16	**435.18**	614.12
	MIP gap	1.77	1.70	1.57	**1.45**	1.55
Variant_B_B	Time	560.29	434.51	315.33	**261.66**	388.25
	MIP gap	2.39	1.89	1.79	**1.72**	2.02
Variant_I_C	Time	1,040.43	1,094.93	801.77	**636.28**	1,282.80
	MIP gap	2.06	1.69	1.69	**1.49**	1.88
Variant_I_B	Time	536.05	492.09	331.40	**286.03**	372.84
	MIP gap	2.38	2.12	2.16	**2.12**	2.27

Table 14: Average of the number of branch and bound nodes: Leading existing branching rules and the solution based approach

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Pseudo-Cost Branching	Reliability Branching	Solution based approach
Variant_B_C	**17.64**	36.70	622.75	300.34	1,601.16
Variant_B_B	**3.22**	8.20	730.10	331.18	1,111.02
Variant_I_C	**39.61**	141.73	2,989.70	1,399.99	4,835.00
Variant_I_B	**2.97**	15.30	717.04	378.15	798.65

Results for Variant_I_C and Variant_I_B for the solutions based approach that calculates average utilization of an arc considering the last copy of the arc:

	Time (sec)	MIP Gap	Total Solving Nodes
Variant_I_C_Approach2	1,310.19	1.87	6,931.38
Variant_I_B_Approach2	370.53	2.29	823.9

The solution based approach shows average performance with respect to the number of branch and bound nodes. It does not outperform the leading existing branching rules, but requires less number of branching nodes than the branching rules such as least infeasible and most infeasible branching. Detailed results (obtained with SCIP solver) of solution based branching rule for all variants of the MCFCNF problem are presented in appendix I.

7.2.1 Observations:

As observed from the computational results presented above, reliability branching rule is the winning branching rule followed by the pseudo-cost branching rule. The solution based approach that we developed ranks 3rd for the Variant_B_C, Variant_B_B, and Variant_I_B. It ranks 4th for Variant_I_C, probably because the set of feasible solution that we used for this variant has very poor quality (as presented above in Table 12).

We have presented some additional observation in the following table. The second column of Table 15 presents the ranks for the solution based rule when compared with the other existing branching rules that we studied. The third column presents the number of instances (out of 20) for which the solution based rule outperforms the winning reliability branching rule. For this, we compared the solution time of 20 instances for reliability branching and the solution based branching rules. For the instances that are terminated because of the specified time limit, we compared the MIP Gap values.

Table 15: Rank of the Solution based Branching Rule

Branching Rules	Rank of the Solution based branching Rule	# of instances for which the solution based branching rule outperforms reliability branching
Variant_B_C	3	5
Variant_B_B	3	10
Variant_I_C	4	4
Variant_I_B	3	9

It is interesting to note that there are 6, 10, 4 and 9 instances for which the solution based branching rule outperforms the reliability branching after comparing them individually.

Furthermore, there are 3 to 4 data-sets for each variant for which the solution based rule shows very poor performance compared to the reliability branching. These 3 to 4 data sets are common for all the variants (r06_6, r09_6, r10_6, r16_6). Following table lists the data-sets that show significantly poor performance for the solution based branching rule.

Table 16: Datasets for which the solution based branching rule shows significantly poor performance

Variant_B_C	Variant_B_B	Variant_I_C	Variant_I_B
r06_6	r06_6	r06_6	r06_6
r09_6	-	r09_6	r09_6
-	r10_6	r10_6	r10_6
r16_6	r16_6	r16_6	-
r10_9	r16_9	r10_9	-

If we exclude these 3 to 4 datasets (from total 20) for each variant, and then analyze the results of the leading branching rules for remaining datasets, the results are as follows:

Table 17: Geometric mean of Time in seconds

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Pseudo-Cost Branching	Reliability Branching	Solution based approach
Variant_B_C	678.05	570.43	516.29	**481.49**	533.73
Variant_B_B	283.98	237.23	225.85	207.74	**182.75**
Variant_I_C	823.25	871.47	703.13	**610.17**	869.42
Variant_I_B	349.37	319.73	248.81	254.96	**221.11**

Table 18: Geometric mean of MIP Gap

Branching Rules	Full Strong^{+} Branching	Full Strong Branching	Pseudo-Cost Branching	Reliability Branching	Solution based approach
Variant_B_C	1.8	1.8	1.7	1.6	1.8
Variant_B_B	2.5	2.3	2.1	2.0	2.3
Variant_I_C	2.5	2.0	1.9	1.6	1.9
Variant_I_B	2.2	2.1	2.1	2.1	2.1

Thus, for Variant_B_C and Variant_I_C, the difference in the performance of solution based and reliability branching rules significantly reduces after excluding the datasets presented in Table 16 for each variant. For the variants Variant_B_B and Variant_I_B, the solution based branching rule outperforms reliability branching. If we notice the nature of these datasets, most of these are the instances with 'Loose Capacity Constraints' (datasets with '_6' in name, Refer to Table 2).

7.3 Results for Studying the Effect of the Quality and Quantity of Feasible Solutions

In the Tables 19 and 20 below, we present the computational results for the solution based approach with different quality and quantity of the feasible solutions. These results are presented for the variant of the network design problem with binary arc variables and continuous flow variables (Variant_B_C). The values in Table 19 represent the solution time in seconds and those in Table 20 represent the MIP gap. Each column in these tables represents the results for the feasible solutions in a specified MIP gap range. We tested all the 20 data-sets over 5 sets of the MIP gap ranges from 0% to 50%, with the interval of 10%. We were unable to collect the feasible solutions for all the datasets in each of these MIP gap ranges with SCIP. Hence, in Tables 19 and 20, we present the geometric mean of the time and MIP gap over the intersection of the instances for which we have the feasible solutions in all the MIP gap ranges. We present all the available results for all the MIP gap ranges in detail in Appendix III.

The rows represent the number of feasible solutions. We tested the solution based approach for 2000, 500 and 100 feasible solutions for each MIP gap range. These solution sets with different number of feasible solutions are not mutually exclusive (The pool of feasible solutions with 500 feasible solutions is a subset of a set with 2000 feasible solutions). Values in the last row represent the average over each quantity of the feasible solutions for each MIP gap range presented in a particular column. And the values in the last column represent the average over all the MIP gap ranges for each quantity of the feasible solutions.

Table 19: Geometric mean of Time in seconds for Variant_B_C

TIME (sec)		MIP Gap Ranges					
		0 to 10	10 to 20	20 to 30	30 to 40	40 to 50	
Quantity of solutions	2000	78.00	120.69	134.75	119.52	165.16	**120.15**
	500	94.64	124.75	111.91	128.95	124.95	**116.31**
	100	95.70	134.71	114.07	119.58	119.75	**116.06**
		89.06	**126.58**	**119.82**	**122.60**	**135.20**	

Table 20: Geometric mean of MIP Gap for Variant_B_C

MIP Gap		MIP Gap Ranges					
		0 to 10	10 to 20	20 to 30	30 to 40	40 to 50	
Quantity of solutions	2000	0.98	0.96	0.99	0.98	1.00	**0.98**
	500	0.96	1.02	0.99	0.97	0.99	**0.98**
	100	0.98	1.06	0.99	1.01	0.99	**1.00**
		0.97	**1.01**	**0.99**	**0.98**	**0.99**	

We can observe from Table 19 that the solution time with the new branching rule improves significantly with the improved quality of the feasible solutions. All the data-sets that we have used for this analysis are solved to the specified MIP gap limit of 1%. Furthermore, change in the number of feasible solutions (tested for 2000, 500 and 100 feasible solutions) does not affect the solution based branching rule significantly.

We also compared the results of the winning existing branching rule (reliability branching) with the results of the solution based approach using different quality and quantity of feasible

solutions. This comparison has been done only for the instances that were used for the results in Tables 19 and 20. For reliability branching, the geometric mean of the solution time for these instances is 23.66, and the geometric mean of MIP Gap is 0.71. Thus, the reliability branching rule wins over the solution based approach for the tested instances.

8. Conclusions and Future Research

We have proposed a solution based approach for branching rule that selects a branching variable corresponding to the arc with the least average utilization. Unlike most of the traditional branching rules that rely on the LP relaxation of the current node or child nodes for selecting the branching variable, we use the information available in the pool of known feasible solutions.

Computational experiments with SCIP and GLPK on the different data-sets demonstrate that the reliability branching rule wins among the existing branching rules followed by the pseudo-cost branching rule. Next to these two rules, the new rule that we developed using the solution based approach performs remarkably well for all the variants of the network design problem. Furthermore, the study of the quality and quantity of the feasible solutions clearly indicates that the performance of the new solution based approach improves as the quality of the pool of available feasible solutions improves. We have assumed that we have the pool of feasible solutions available. Hence, the time required for collecting the pool of feasible solutions has not been considered while measuring the performance of the new rule. However, various heuristics and meta-heuristics can be used for collecting the pool of feasible solutions with better quality.

For future research, we can consider collecting the feasible solutions more effectively. We can find the scope of further improvement in the solution based approach by using the hybrid approaches. For example, we can use the strong branching method in the beginning of the search until we obtain the feasible solutions from the current branch and bound tree. The solution based approach can then be used for the remaining part using these feasible solutions collected from the current branch and bound tree.

Acknowledgements

First and foremost, I offer my sincerest gratitude to Dr. Michael Hewitt for being my advisor and guide. I am grateful to him for his continuous encouragement and the invaluable inputs that he has provided through the development of this thesis. This work would not have been possible without his guidance.

I would also like to thank Dr. Michael E. Kuhl and Dr. Scott Grasman for being in my thesis committee and for their advice and insight throughout my work.

I am also indebted to the many contributors to the GLPK and SCIP open source programming community. I thank them for the quick responses with any doubts during the development of my application programs. I also received a lot of support from our systems administrators Paul Mezzanini, and Ralph Bean for installing and maintaining SCIP and GLPK on our server.

References

[1] Achterberg, T., Koch, T., & Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, *33*(1), 42-54.

[2] Atamtürk, A., & Rajan, D. (2002). On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming*, *92*(2), 315-333.

[3] Atamtürk, A., & Savelsbergh, M. W. (2005). Integer-programming software systems. *Annals of Operations Research*, *140*(1), 67-124.

[4] Barnhart, C., & Cohn, A. (2004). Airline schedule planning: Accomplishments and opportunities. *Manufacturing & service operations management*, *6*(1), 3-22.

[5] Barnhart, C., Jin, H., & Vance, P. H. (2000). Railroad blocking: A network design application. *Operations Research*, *48*(4), 603-614.

[6] Bertsimas, D., & Tsitsiklis, J. N. (1997). Introduction to linear optimization, *Athena Scientific, Belmont, Massachusetts.*

[7] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001*)*. Introduction to algorithms. *MIT Press, Cambridge, Massachusetts.*

[8] CPLEX,I.L.O.G.(2003).9.0 user's manual. *ILOG, SA.*(*http://www.ilog.com/products/cplex*)

[9] Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research, 122(2),* 272-288.

[10] Crainic, T. G., Frangioni, A., & Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, *112*(1), 73-99.

[11] Crainic, T. G., Gendreau, M., & Farvolden, J. M. (2000). A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, *12*(3), 223-236.

[12] Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, *8*(3), 250-255.

[13] Danna, E., Rothberg, E., & Pape, C. L. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, *102*(1), 71-90.

[14] Eckstein, J. (1994). Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5. *SIAM Journal on Optimization*, *4*(4), 794-814.

[15] Erera, A., Hewitt, M., Savelsbergh, M., & Zhang, Y. (2012). Improved load plan design through integer programming based local search. *Transportation Science, Advance online publication. doi:10.1287/trsc.1120.0441*

[16] Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, *98*(1), 23-47.

[17] Gendron, B., & Crainic, T. G. (1994). Relaxations for multicommodity capacitated network design problems. *Publication CRT-965, Université de Montréal, Centre de recherche sur les transports, Montre´al, QC, Canada.*

[18] Gendron B., & Crainic T.G.. (1996). Bounding procedures for multicommodity capacitated network design problems. *Publication CRT-96-06, Universite´ de Montre´al, Centre de Recherche sur les Transports, Montre´al, QC, Canada.*

[19] Ghamlouche, I., Crainic, T. G., & Gendreau, M. (2003). Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research*, *51*(4), 655-667.

[20] Gilpin, A., & Sandholm, T. (2011). Information-theoretic approaches to branching in search. *Discrete Optimization*, *8*(2), 147-159.

[21] GLPK (GNU linear programming kit). Accessed January 15, 2012, from *http://www.gnu.org/software/glpk/*

[22] Hewitt, M., Nemhauser, G. L., & Savelsbergh, M. W. (2010). Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, *22*(2), 314-325.

[23] Kim, D., Barnhart, C., Ware, K., & Reinhardt, G. (1999). Multimodal express package delivery: A service network design application. *Transportation Science*, *33*(4), 391-407.

[24] Linderoth, J. T., & Savelsbergh, M. W. (1999). A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, *11*(2), 173-187.

[25] Padberg, M. W., Van Roy, T. J., & Wolsey, L. A. (1985). Valid linear inequalities for fixed charge problems. *Operations Research*, *33*(4), 842-861.

[26] Patel, J., & Chinneck, J. W. (2007). Active-constraint variable ordering for faster feasibility of mixed integer linear programs. *Mathematical Programming*, *110*(3), 445-474.

[27] Powell, W. B., & Sheffi, Y. (1989). OR Practice—Design and Implementation of an Interactive Optimization System for Network Design in the Motor Carrier Industry. *Operations Research*, *37*(1), 12-29.

[28] Rabetanety, A., (2006). Airline Schedule Planning Integrated Flight Schedule Design and Product Line Design (Doctoral dissertation, Tese. Universität Karlsruhe. Karlsruhe).

[29] Savelsbergh, M. W. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, *6*(4), 445-454.

[30] SCIP (Solving Constraint Integer Programs). Accessed December 15, 2012, from *http://scip.zib.de/scip.shtml*

[31] Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, *5*(1), 3-55.

[32] Wolsey, L. A. (1998). Integer programming *Wiley-Interscience, Hoboken, New Jersey.*

APPENDIX I

RESULTS WITH SCIP: Branching Rules

Part A: Variant_B_C

Time:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time in seconds	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		487.54	253.62	2,524.59	3,891.32	233.74	1,254.42	138.36	491.75
r07_6	10,82,10	F,L (10,2)		0.39	0.47	0.65	0.69	0.65	0.65	0.65	0.41
r08_6	10,83,25	F,L (10,2)		5.62	6.66	10.64	9.22	8.85	7.10	8.60	5.00
r09_6	10,83,50	F,L (10,2)		2,274.02	1,521.54	5,455.43	2,645.11	465.14	2,547.86	265.94	1,512.23
r10_6	20,120,40	F,L (10,2)		3,878.25	1,343.92	229.62	169.74	86.30	141.50	146.23	73.55
r14_6	20,220,100	F,L (10,2)		14,402.30	14,402.60	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_6	20,220,200	F,L (10,2)		14,412.60	14,411.10	14,400.00	14,400.00	14,400.00	14,401.20	14,400.00	14,400.60
r16_6	20,314,40	F,L (10,2)		14,400.10	14,400.20	14,400.00	14,400.00	6,172.02	14,400.00	2,603.21	11,382.60
r17_6	20,318,100	F,L (10,2)		14,403.80	14,402.50	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_6	20,315,200	F,L (10,2)		14,415.70	14,406.40	14,400.00	14,400.00	14,400.00	14,400.00	14,401.40	14,400.00
r06_9	10,60,50	F,T (10,8)		15.95	16.39	22.41	20.75	16.78	17.23	13.13	18.87
r07_9	10,82,10	F,T (10,8)		22.09	12.08	97.95	87.78	22.17	33.39	8.57	50.60
r08_9	10,83,25	F,T (10,8)		200.30	88.81	1,472.28	2,429.41	45.95	608.37	40.93	112.45
r09_9	10,83,50	F,T (10,8)		22.98	11.34	29.22	138.33	13.35	87.35	12.01	16.33
r10_9	20,120,40	F,T (10,8)		1,004.26	241.26	14,400.00	7,251.00	154.61	4,031.43	74.22	3,718.73
r14_9	20,220,100	F,T (10,8)		14,400.20	14,401.10	14,400.00	14,400.00	14,400.00	14,400.00	12,153.90	14,400.00
r15_9	20,220,200	F,T (10,8)		78.09	80.86	90.20	94.44	121.67	96.84	118.27	88.74
r16_9	20,314,40	F,T (10,8)		14,400.00	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r17_9	20,318,100	F,T (10,8)		14,402.40	14,402.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_9	20,315,200	F,T (10,8)		14,405.30	14,407.80	14,400.00	14,400.30	14,401.70	14,401.20	14,401.60	14,400.00
			Time in sec: Geometric Mean	841.90	647.64	1,237.92	1,272.56	526.16	979.68	435.18	719.05
			Variance	7,040.64	7,197.30	6,985.72	6,790.82	7,058.05	7,007.19	6,992.24	6,976.49
			NonOpt_Avg	10,153.53	9,787.19	10,114.90	9,939.40	8,016.83	10,074.55	8,638.24	9,107.95

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	MIP Gap	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		0.88	0.99	1.00	1.00	1.00	1.00	1.00	1.00
r07_6	10,82,10	F,L (10,2)		0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
r08_6	10,83,25	F,L (10,2)		1.00	0.80	0.90	0.82	0.94	0.63	0.98	0.67
r09_6	10,83,50	F,L (10,2)		0.83	0.92	1.00	1.00	1.00	1.00	1.00	1.00
r10_6	20,120,40	F,L (10,2)		0.76	0.76	0.99	0.99	1.00	0.97	0.97	0.98
r14_6	20,220,100	F,L (10,2)		7.03	6.80	5.39	4.51	5.35	4.58	4.36	5.63
r15_6	20,220,200	F,L (10,2)		7.68	7.35	6.84	7.56	7.48	7.68	7.37	7.68
r16_6	20,314,40	F,L (10,2)		5.26	3.65	2.69	1.63	1.00	1.52	1.00	1.00
r17_6	20,318,100	F,L (10,2)		8.72	8.62	5.72	5.28	5.09	5.10	5.06	5.55
r18_6	20,315,200	F,L (10,2)		5.35	5.09	4.52	4.75	5.01	4.82	5.35	5.22
r06_9	10,60,50	F,T (10,8)		0.80	0.97	0.96	0.98	0.98	1.00	0.97	1.00
r07_9	10,82,10	F,T (10,8)		0.86	0.76	1.00	1.00	0.99	0.99	0.83	1.00
r08_9	10,83,25	F,T (10,8)		0.91	0.87	1.00	1.00	0.99	1.00	0.95	1.00
r09_9	10,83,50	F,T (10,8)		0.79	0.96	0.97	1.00	0.91	1.00	0.98	0.97
r10_9	20,120,40	F,T (10,8)		0.97	0.97	1.84	1.00	0.99	1.00	0.93	1.00
r14_9	20,220,100	F,T (10,8)		2.72	1.83	3.67	3.34	1.61	3.28	0.98	3.23
r15_9	20,220,200	F,T (10,8)		0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56
r16_9	20,314,40	F,T (10,8)		7.10	6.63	5.62	5.08	4.58	4.81	3.87	5.99
r17_9	20,318,100	F,T (10,8)		5.65	5.42	5.69	4.68	4.76	5.01	4.05	5.32
r18_9	20,315,200	F,T (10,8)		4.88	4.38	4.60	4.50	5.64	5.60	3.84	4.34
			MIP Gap: Geometric Mean	1.77	1.70	1.78	1.64	1.57	1.63	1.45	1.63
			Variance	2.94	2.78	2.23	2.17	2.28	2.26	2.08	2.41
			NumOpt	11.00	11.00	10.00	11.00	12.00	11.00	13.00	12.00
			NonOpt_Avg	6.04	5.53	4.66	4.96	4.94	4.71	4.84	5.37

Solving Nodes:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Total Solving Nodes	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		543.00	691.00	152,741.00	189,101.00	9,741.00	89,631.00	4,851.00	49,721.00
r07_6	10,82,10	F,L (10,2)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r08_6	10,83,25	F,L (10,2)		1.00	2.00	681.00	197.00	94.00	171.00	2.00	117.00
r09_6	10,83,50	F,L (10,2)		870.00	1,249.00	645,941.00	79,621.00	13,621.00	149,111.00	6,541.00	166,381.00
r10_6	20,120,40	F,L (10,2)		217.00	194.00	4,171.00	631.00	411.00	1,181.00	691.00	1,261.00
r14_6	20,220,100	F,L (10,2)		17.00	45.00	27,243.00	8,016.00	9,235.00	13,553.00	6,714.00	6,707.00
r15_6	20,220,200	F,L (10,2)		1.00	2.00	1,103.00	57.00	180.00	1.00	100.00	1.00
r16_6	20,314,40	F,L (10,2)		56.00	459.00	220,994.00	87,412.00	23,751.00	124,033.00	6,261.00	86,041.00
r17_6	20,318,100	F,L (10,2)		2.00	10.00	21,165.00	5,863.00	7,868.00	10,942.00	5,438.00	10,598.00
r18_6	20,315,200	F,L (10,2)		1.00	2.00	435.00	236.00	228.00	214.00	1.00	16.00
r06_9	10,60,50	F,T (10,8)		6.00	29.00	1,211.00	461.00	201.00	504.00	24.00	521.00
r07_9	10,82,10	F,T (10,8)		67.00	181.00	61,211.00	27,181.00	5,961.00	9,871.00	765.00	31,421.00
r08_9	10,83,25	F,T (10,8)		281.00	411.00	301,151.00	226,241.00	2,671.00	81,091.00	2,281.00	18,151.00
r09_9	10,83,50	F,T (10,8)		16.00	15.00	552.00	5,332.00	122.00	2,092.00	19.00	532.00
r10_9	20,120,40	F,T (10,8)		423.00	372.00	602,833.00	171,881.00	1,951.00	115,931.00	275.00	77,671.00
r14_9	20,220,100	F,T (10,8)		94.00	234.00	22,749.00	9,660.00	9,553.00	8,630.00	19,939.00	21,651.00
r15_9	20,220,200	F,T (10,8)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r16_9	20,314,40	F,T (10,8)		87.00	243.00	119,241.00	35,656.00	38,534.00	51,748.00	50,791.00	84,241.00
r17_9	20,318,100	F,T (10,8)		7.00	52.00	1,635.00	3,781.00	756.00	758.00	3,030.00	3,666.00
r18_9	20,315,200	F,T (10,8)		1.00	2.00	459.00	401.00	1.00	1.00	53.00	1,251.00
			Solving Nodes: Geometric Mean	17.64	36.70	5,256.22	2,554.81	622.75	1,340.66	300.34	1,601.16
			Variance	231.59	313.36	195,256.89	71,244.99	9,828.66	49,779.98	11,680.36	44,111.08

Part B: Variant_B_B

Time:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time in seconds	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		4,364.73	1,354.11	14,400.00	14,400.00	351.48	14,400.00	233.24	1,279.01
r07_6	10,82,10	F,L (10,2)		0.10	0.18	0.15	0.18	0.17	0.16	0.14	0.16
r08_6	10,83,25	F,L (10,2)		4.83	5.26	7.11	5.50	4.72	4.27	6.62	2.27
r09_6	10,83,50	F,L (10,2)		14,400.00	3,204.92	14,400.00	14,400.00	990.37	14,400.00	461.51	453.93
r10_6	20,120,40	F,L (10,2)		14,400.00	3,991.87	14,400.00	10,990.50	1,093.08	14,400.00	561.88	3,266.30
r14_6	20,220,100	F,L (10,2)		14,401.60	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r16_6	20,314,40	F,L (10,2)		14,400.40	11,412.40	14,400.00	14,400.00	2,769.70	14,400.00	1,026.87	14,400.00
r17_6	20,318,100	F,L (10,2)		14,401.00	14,404.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_6	20,315,200	F,L (10,2)		14,415.60	14,407.40	14,400.00	14,400.00	14,400.00	14,432.60	14,400.00	14,400.00
r06_9	10,60,50	F,T (10,8)		376.23	64.95	2,797.34	1,026.96	71.92	3,382.49	35.26	26.60
r08_9	10,83,25	F,T (10,8)		0.16	0.18	0.23	0.18	0.25	0.22	0.24	0.18
r09_9	10,83,50	F,T (10,8)		1.03	0.80	1.13	1.13	1.00	1.05	1.00	0.88
r10_9	20,120,40	F,T (10,8)		0.40	0.49	0.64	0.55	0.49	0.49	0.60	0.52
r14_9	20,220,100	F,T (10,8)		14,404.70	14,400.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_9	20,220,200	F,T (10,8)		14,400.90	14,408.90	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r16_9	20,314,40	F,T (10,8)		1,474.62	2,761.11	14,400.00	14,400.00	993.07	14,400.00	880.73	14,400.00
r17_9	20,318,100	F,T (10,8)		14,400.30	14,402.50	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_9	20,315,200	F,T (10,8)		14,400.90	14,405.80	14,400.00	14,400.00	14,400.00	14,400.00	14,400.10	14,400.00
			Time in sec: Geometric Mean	560.29	434.51	835.61	758.47	315.33	806.50	261.66	388.25
			Variance	7,035.92	6,726.60	6,786.41	6,813.53	6,968.50	6,754.01	7,083.34	7,161.15
			NonOpt_Avg	8,936.18	8,767.52	11,782.52	11,179.66	7,807.04	11,782.25	7,508.76	11,208.35

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	MIP gap	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		0.89	0.87	2.40	1.07	1.00	1.27	0.99	1.00
r07_6	10,82,10	F,L (10,2)		0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
r08_6	10,83,25	F,L (10,2)		0.60	0.91	0.65	0.99	0.93	0.96	0.53	0.98
r09_6	10,83,50	F,L (10,2)		3.19	0.96	3.21	1.69	1.00	1.73	1.00	1.00
r10_6	20,120,40	F,L (10,2)		4.77	0.92	3.85	1.00	0.92	1.83	0.93	1.00
r14_6	20,220,100	F,L (10,2)		24.36	23.47	24.34	23.37	16.82	23.75	22.51	22.59
r16_6	20,314,40	F,L (10,2)		3.50	0.89	3.28	2.07	1.00	2.19	1.00	1.13
r17_6	20,318,100	F,L (10,2)		11.38	10.82	11.38	10.51	10.23	10.51	9.79	9.71
r18_6	20,315,200	F,L (10,2)		22.42	22.42	22.42	22.21	21.92	22.30	22.06	21.74
r06_9	10,60,50	F,T (10,8)		0.90	0.91	1.00	1.00	0.45	1.00	0.48	0.96
r08_9	10,83,25	F,T (10,8)		0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
r09_9	10,83,50	F,T (10,8)		0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
r10_9	20,120,40	F,T (10,8)		0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66
r14_9	20,220,100	F,T (10,8)		3.93	2.96	4.12	3.49	2.55	3.66	1.73	3.79
r15_9	20,220,200	F,T (10,8)		2.74	2.70	2.74	2.69	2.62	2.67	2.64	2.71
r16_9	20,314,40	F,T (10,8)		1.00	0.98	2.39	1.68	1.00	1.48	1.00	1.53
r17_9	20,318,100	F,T (10,8)		18.59	17.40	18.59	17.78	17.31	18.36	16.47	18.28
r18_9	20,315,200	F,T (10,8)		16.04	15.63	16.04	15.87	15.75	15.93	15.74	15.93
			MIP Gap: Geometric Mean	2.39	1.89	2.65	2.18	1.79	2.27	1.72	2.02
			Variance	8.22	8.20	8.13	8.09	7.39	8.17	7.98	8.01
			NumOpt	8.00	11.00	6.00	7.00	11.00	6.00	11.00	9.00
			NonOpt_Avg	8.46	13.63	9.56	9.31	11.02	8.81	11.49	12.03

Solving Nodes:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Total Solving Nodes	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		218.00	661.00	1,279,631.00	886,882.00	18,661.00	1,239,302.00	6,381.00	131,621.00
r07_6	10,82,10	F,L (10,2)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r08_6	10,83,25	F,L (10,2)		7.00	6.00	1,436.00	280.00	150.00	141.00	11.00	83.00
r09_6	10,83,50	F,L (10,2)		1.00	293.00	1,243,973.00	529,863.00	26,291.00	836,463.00	6,985.00	20,511.00
r10_6	20,120,40	F,L (10,2)		1.00	183.00	470,747.00	325,541.00	14,641.00	503,581.00	4,809.00	66,071.00
r14_6	20,220,100	F,L (10,2)		1.00	6.00	7,491.00	7,730.00	6,312.00	11,336.00	1,390.00	5,035.00
r16_6	20,314,40	F,L (10,2)		3.00	62.00	257,926.00	124,116.00	19,222.00	167,781.00	5,422.00	184,281.00
r17_6	20,318,100	F,L (10,2)		1.00	2.00	1,751.00	7,446.00	5,410.00	3,590.00	7,404.00	12,378.00
r18_6	20,315,200	F,L (10,2)		1.00	1.00	1,241.00	353.00	665.00	838.00	148.00	697.00
r06_9	10,60,50	F,T (10,8)		10.00	23.00	798,192.00	160,232.00	4,990.00	596,962.00	244.00	1,975.00
r08_9	10,83,25	F,T (10,8)		202.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r09_9	10,83,50	F,T (10,8)		17.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r10_9	20,120,40	F,T (10,8)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r14_9	20,220,100	F,T (10,8)		1.00	8.00	2,456.00	13,686.00	17,199.00	17,366.00	19,506.00	9,140.00
r15_9	20,220,200	F,T (10,8)		1.00	2.00	3,310.00	1,933.00	4,055.00	5,453.00	3,810.00	3,225.00
r16_9	20,314,40	F,T (10,8)		9.00	162.00	663,042.00	296,593.00	26,122.00	427,801.00	17,792.00	426,336.00
r17_9	20,318,100	F,T (10,8)		1.00	3.00	1,687.00	2,902.00	1,572.00	2,385.00	2,521.00	3,437.00
r18_9	20,315,200	F,T (10,8)		1.00	1.00	688.00	226.00	517.00	523.00	145.00	658.00
			Solving Nodes: Geometric Mean	3.22	8.20	2,561.64	1,883.58	730.10	2,481.22	331.18	1,111.02
			Variance	66.96	167.24	440,925.01	241,783.89	9,499.09	363,738.55	5,896.46	107,514.62

Part C: Variant_I_C

Time:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time in seconds	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		717.67	734.79	14,400.00	13,909.20	519.07	6,968.09	237.17	1,741.00
r07_6	10,82,10	F,L (10,2)		0.85	0.66	0.81	0.56	0.67	0.70	0.74	0.44
r08_6	10,83,25	F,L (10,2)		7.10	5.85	6.15	7.03	6.28	6.05	4.61	5.05
r09_6	10,83,50	F,L (10,2)		2,252.78	1,826.07	14,400.00	14,400.00	638.57	11,185.40	435.49	3,783.66
r10_6	20,120,40	F,L (10,2)		3,572.62	3,327.20	14,400.00	10,650.90	1,646.93	14,400.00	1,144.58	14,400.00
r14_6	20,220,100	F,L (10,2)		14,401.90	14,402.80	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_6	20,220,200	F,L (10,2)		14,415.40	14,433.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r16_6	20,314,40	F,L (10,2)		14,400.10	9,108.97	12,588.20	11,892.00	2,429.68	3,197.31	2,675.11	4,442.12
r17_6	20,318,100	F,L (10,2)		14,419.20	14,407.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_6	20,315,200	F,L (10,2)		14,409.50	14,418.30	14,400.00	14,410.00	14,400.00	14,400.00	14,400.00	14,401.50
r06_9	10,60,50	F,T (10,8)		151.36	41.10	11.09	12.42	12.08	16.16	18.11	29.81
r07_9	10,82,10	F,T (10,8)		13.91	18.28	57.00	69.93	12.62	71.47	10.54	77.68
r08_9	10,83,25	F,T (10,8)		4.76	25.88	69.64	73.98	27.37	48.96	5.58	28.11
r09_9	10,83,50	F,T (10,8)		28.38	86.57	153.62	74.74	182.70	115.21	48.23	230.95
r10_9	20,120,40	F,T (10,8)		8,590.86	12,412.80	14,400.00	14,400.00	6,184.22	14,400.00	2,710.12	14,400.00
r14_9	20,220,100	F,T (10,8)		14,400.10	14,400.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_9	20,220,200	F,T (10,8)		971.00	1,006.92	540.03	211.36	247.74	265.48	425.97	388.52
r16_9	20,314,40	F,T (10,8)		14,400.10	14,401.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r17_9	20,318,100	F,T (10,8)		14,401.30	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_9	20,315,200	F,T (10,8)		14,400.40	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.10	14,400.00
			Time in sec: Geometric Mean	1,040.43	1,094.93	1,633.09	1,484.48	801.77	1,387.55	636.28	1,282.80
			Variance	6,865.40	6,801.54	6,932.34	6,828.02	6,877.47	6,812.75	6,957.40	6,935.36
			NonOpt_Avg	9,960.09	10,433.39	10,371.64	9,483.60	8,513.57	10,116.34	8,069.47	9,524.53

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	MIP Gap	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		0.71	0.99	1.12	1.00	1.00	1.00	1.00	1.00
r07_6	10,82,10	F,L (10,2)		0.60	0.06	0.97	0.79	0.92	0.49	0.06	0.72
r08_6	10,83,25	F,L (10,2)		0.80	0.65	0.81	0.92	0.92	0.99	0.72	0.92
r09_6	10,83,50	F,L (10,2)		0.96	1.00	1.93	1.23	1.00	1.00	1.00	1.00
r10_6	20,120,40	F,L (10,2)		0.99	0.89	2.24	1.00	1.00	1.97	1.00	2.08
r14_6	20,220,100	F,L (10,2)		7.34	7.60	6.51	5.97	4.44	4.85	6.02	5.86
r15_6	20,220,200	F,L (10,2)		55.42	55.65	5.33	4.97	4.78	5.03	3.83	4.80
r16_6	20,314,40	F,L (10,2)		3.45	0.59	1.00	1.00	1.00	1.00	0.99	1.00
r17_6	20,318,100	F,L (10,2)		7.25	7.25	6.88	5.67	6.20	5.91	5.96	6.46
r18_6	20,315,200	F,L (10,2)		7.75	7.25	5.45	7.75	7.51	7.73	7.09	7.75
r06_9	10,60,50	F,T (10,8)		0.50	0.88	0.97	0.93	0.97	0.97	0.97	0.97
r07_9	10,82,10	F,T (10,8)		0.97	0.95	1.00	1.00	0.99	1.00	1.00	1.00
r08_9	10,83,25	F,T (10,8)		0.86	0.91	1.00	0.91	0.99	1.00	0.99	0.99
r09_9	10,83,50	F,T (10,8)		0.81	1.00	0.88	1.00	0.95	0.97	0.99	0.97
r10_9	20,120,40	F,T (10,8)		1.00	1.00	2.73	2.57	1.00	2.43	1.00	2.62
r14_9	20,220,100	F,T (10,8)		3.18	3.32	3.30	3.04	3.05	3.06	2.60	3.00
r15_9	20,220,200	F,T (10,8)		0.98	0.67	0.85	1.00	0.78	0.98	1.00	0.65
r16_9	20,314,40	F,T (10,8)		6.72	3.87	3.46	3.23	2.40	3.02	2.83	3.08
r17_9	20,318,100	F,T (10,8)		3.72	4.40	3.94	3.52	3.41	3.58	3.40	3.74
r18_9	20,315,200	F,T (10,8)		2.81	2.81	2.61	2.21	2.57	2.69	2.99	2.78
			MIP Gap: Geometric Mean	2.06	1.69	2.01	1.84	1.69	1.85	1.49	1.88
			Variance	12.06	12.15	2.02	2.09	2.01	2.02	2.04	2.14
			NumOpt	11.00	12.00	8.00	10.00	12.00	10.00	12.00	10.00
			NonOpt_Avg	9.86	11.52	4.04	4.02	4.29	4.03	4.34	4.22

Solving Nodes:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Total Solving Nodes	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		1,290.00	1,941.00	2,546,571.00	950,241.00	36,641.00	394,001.00	14,211.00	364,351.00
r07_6	10,82,10	F,L (10,2)		11.00	36.00	151.00	109.00	141.00	101.00	38.00	154.00
r08_6	10,83,25	F,L (10,2)		3.00	2.00	131.00	53.00	32.00	111.00	2.00	87.00
r09_6	10,83,50	F,L (10,2)		1,301.00	2,221.00	972,185.00	403,725.00	31,531.00	575,261.00	12,191.00	233,571.00
r10_6	20,120,40	F,L (10,2)		561.00	1,262.00	973,494.00	292,341.00	29,001.00	381,219.00	15,421.00	514,911.00
r14_6	20,220,100	F,L (10,2)		11.00	22.00	9,631.00	4,431.00	5,849.00	5,805.00	1,119.00	19,978.00
r15_6	20,220,200	F,L (10,2)		4.00	3.00	2,213.00	927.00	1,157.00	1,119.00	788.00	2,184.00
r16_6	20,314,40	F,L (10,2)		41.00	110.00	196,981.00	13,601.00	3,351.00	12,191.00	1,561.00	25,181.00
r17_6	20,318,100	F,L (10,2)		2.00	7.00	29,731.00	2,063.00	2,497.00	2,360.00	964.00	706.00
r18_6	20,315,200	F,L (10,2)		1.00	2.00	1,131.00	1.00	81.00	10.00	146.00	1.00
r06_9	10,60,50	F,T (10,8)		210.00	167.00	551.00	291.00	181.00	691.00	261.00	1,571.00
r07_9	10,82,10	F,T (10,8)		333.00	1,161.00	57,571.00	61,341.00	6,441.00	40,291.00	2,871.00	41,061.00
r08_9	10,83,25	F,T (10,8)		3.00	402.00	14,701.00	9,817.00	1,711.00	8,261.00	453.00	3,081.00
r09_9	10,83,50	F,T (10,8)		3.00	433.00	10,400.00	3,111.00	3,940.00	6,286.00	851.00	6,840.00
r10_9	20,120,40	F,T (10,8)		4,822.00	13,141.00	703,333.00	286,279.00	300,391.00	503,376.00	98,051.00	399,178.00
r14_9	20,220,100	F,T (10,8)		228.00	769.00	45,626.00	20,231.00	20,928.00	31,794.00	24,574.00	17,939.00
r15_9	20,220,200	F,T (10,8)		33.00	201.00	217.00	45.00	98.00	105.00	503.00	98.00
r16_9	20,314,40	F,T (10,8)		189.00	931.00	116,518.00	65,636.00	90,539.00	84,009.00	71,125.00	124,999.00
r17_9	20,318,100	F,T (10,8)		11.00	69.00	21,180.00	4,852.00	5,690.00	1,636.00	3,779.00	3,400.00
r18_9	20,315,200	F,T (10,8)		17.00	61.00	3,383.00	4,272.00	1,882.00	3,450.00	226.00	264.00
			Solving Nodes: Geometric Mean	39.61	141.73	16,364.82	4,549.57	2,989.70	5,833.92	1,399.99	4,835.00
			Variance	1101.49	2901.09	620030.86	231331.86	67863.70	189782.43	25946.00	158234.21

Results for the solution based branching rule with approach 2 (that calculates average utilization of the last copy of the arc)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time (sec)	MIP Gap	Total Solving Nodes
r06_6	10,60,50	F,L (10,2)	3,512.03	1.00	439,601.00
r07_6	10,82,10	F,L (10,2)	0.71	0.72	160.00
r08_6	10,83,25	F,L (10,2)	5.87	0.92	113.00
r09_6	10,83,50	F,L (10,2)	3,140.79	1.00	228,881.00
r10_6	20,120,40	F,L (10,2)	14,400.00	2.11	598,054.00
r14_6	20,220,100	F,L (10,2)	14,400.00	4.96	23,308.00
r15_6	20,220,200	F,L (10,2)	14,400.00	4.80	2,089.00
r16_6	20,314,40	F,L (10,2)	4,025.38	1.00	25,621.00
r17_6	20,318,100	F,L (10,2)	14,400.00	6.14	5,703.00
r18_6	20,315,200	F,L (10,2)	14,400.10	7.75	7.00
r06_9	10,60,50	F,T (10,8)	20.87	1.00	1,071.00
r07_9	10,82,10	F,T (10,8)	42.66	1.00	34,551.00
r08_9	10,83,25	F,T (10,8)	31.53	0.98	3,971.00
r09_9	10,83,50	F,T (10,8)	257.39	0.98	8,832.00
r10_9	20,120,40	F,T (10,8)	14,400.00	2.61	340,344.00
r14_9	20,220,100	F,T (10,8)	14,400.00	2.81	12,287.00
r15_9	20,220,200	F,T (10,8)	433.47	0.79	58.00
r16_9	20,314,40	F,T (10,8)	14,400.00	3.04	174,326.00
r17_9	20,318,100	F,T (10,8)	14,400.00	3.66	8,473.00
r18_9	20,315,200	F,L (10,8)	14,400.00	2.73	2,601.00
		Geometric Mean	1,310.20	1.87	6,931.38

Part C: Variant_I_B

Time:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time in seconds	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		5,293.32	5,488.07	14,400.00	14,400.00	1,760.68	14,400.00	551.21	14,400.00
r07_6	10,82,10	F,L (10,2)		0.18	0.19	0.19	0.18	0.30	0.23	0.19	0.17
r08_6	10,83,25	F,L (10,2)		1.15	1.21	1.71	1.17	1.33	1.29	1.52	1.73
r09_6	10,83,50	F,L (10,2)		14,400.00	10,671.80	14,400.00	14,400.00	3,838.40	14,400.00	1,222.35	14,400.00
r10_6	20,120,40	F,L (10,2)		1,242.28	1,310.02	14,400.00	5,221.75	396.64	14,008.00	194.91	633.05
r14_6	20,220,100	F,L (10,2)		14,403.50	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r16_6	20,314,40	F,L (10,2)		14,400.40	9,404.75	14,400.00	14,400.00	1,078.02	14,400.00	1,149.62	810.86
r17_6	20,318,100	F,L (10,2)		14,408.60	14,401.50	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r18_6	20,315,200	F,L (10,2)		14,412.30	14,412.10	14,400.00	14,400.10	14,400.00	14,400.00	14,400.00	14,400.00
r06_9	10,60,50	F,T (10,8)		87.63	29.10	728.56	738.25	23.91	1,600.54	15.21	7.57
r08_9	10,83,25	F,T (10,8)		0.59	0.70	0.72	0.72	0.77	0.68	0.77	0.66
r09_9	10,83,50	F,T (10,8)		12.27	13.08	4.90	3.91	3.99	3.64	7.24	3.92
r10_9	20,120,40	F,T (10,8)		2.44	3.16	2.89	2.88	3.06	2.76	3.07	3.01
r14_9	20,220,100	F,T (10,8)		14,400.00	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r15_9	20,220,200	F,T (10,8)		14,400.60	14,400.10	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
r16_9	20,314,40	F,T (10,8)		237.69	159.49	413.56	152.27	69.87	53.39	112.69	81.81
r17_9	20,318,100	F,T (10,8)		14,400.60	14,400.10	14,400.00	14,400.00	14,400.00	14,400.00	14,401.70	14,400.00
r18_9	20,315,200	F,T (10,8)		14,400.50	14,400.20	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00	14,400.00
			Time in sec: Geometric Mean	536.05	492.09	749.05	646.04	331.40	678.56	286.03	372.84
			Variance	7,118.51	6,762.31	7,143.01	7,068.64	6,957.66	7,096.05	7,084.16	7,323.76
			NonOpt_Avg	8,166.30	7,725.60	10,121.82	7,577.76	6,392.51	10,046.54	6,380.28	5,591.31

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	MIP Gap	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		0.97	0.72	2.57	1.71	1.00	1.61	1.00	1.00
r07_6	10,82,10	F,L (10,2)		0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
r08_6	10,83,25	F,L (10,2)		0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
r09_6	10,83,50	F,L (10,2)		2.42	0.93	3.91	2.49	1.00	2.82	1.00	2.17
r10_6	20,120,40	F,L (10,2)		0.70	0.89	1.51	1.00	0.99	1.00	0.97	1.00
r14_6	20,220,100	F,L (10,2)		11.62	10.79	11.55	10.78	10.27	10.43	9.86	10.52
r16_6	20,314,40	F,L (10,2)		2.57	0.99	2.03	1.06	1.00	1.03	0.99	1.00
r17_6	20,318,100	F,L (10,2)		12.60	12.03	11.81	11.46	11.39	11.08	10.70	11.35
r18_6	20,315,200	F,L (10,2)		15.85	15.85	12.75	15.79	15.64	15.05	14.91	13.14
r06_9	10,60,50	F,T (10,8)		0.79	0.83	1.00	1.00	0.96	1.00	0.97	0.98
r08_9	10,83,25	F,T (10,8)		0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
r09_9	10,83,50	F,T (10,8)		0.94	0.99	0.99	0.99	0.99	0.99	0.99	0.99
r10_9	20,120,40	F,T (10,8)		0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
r14_9	20,220,100	F,T (10,8)		9.76	9.44	7.48	9.32	7.75	5.88	7.84	8.52
r15_9	20,220,200	F,T (10,8)		2.49	2.40	2.47	2.31	2.20	2.30	2.04	2.32
r16_9	20,314,40	F,T (10,8)		0.86	0.92	1.00	0.99	0.99	1.00	0.99	1.00
r17_9	20,318,100	F,T (10,8)		8.56	8.37	8.33	7.92	8.25	8.07	8.15	8.09
r18_9	20,315,200	F,T (10,8)		12.22	11.94	10.74	12.05	11.21	11.07	10.71	12.08
			MIP Gap: Geometric Mean	2.38	2.12	2.62	2.39	2.16	2.32	2.12	2.26
			Variance	5.32	5.25	4.52	5.08	4.97	4.71	4.74	4.74
			NumOpt	9.00	11.00	7.00	8.00	11.00	8.00	11.00	10.00
			NonOpt_Avg	8.68	10.12	6.83	8.46	9.53	6.39	9.17	8.52

Solving Nodes:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Total Solving Nodes	Full Strong+ Branching	Full Strong Branching	Least Infeasible Branching	Most Infeasible Branching	Pseudo-cost Branching	Random Branching	Reliability Branching	Solution Based Rule
r06_6	10,60,50	F,L (10,2)		363.00	1,965.00	1,272,046.00	902,861.00	95,502.00	1,230,455.00	31,242.00	1,692,802.00
r07_6	10,82,10	F,L (10,2)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r08_6	10,83,25	F,L (10,2)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r09_6	10,83,50	F,L (10,2)		353.00	3,918.00	626,779.00	849,766.00	211,041.00	944,262.00	66,691.00	447,061.00
r10_6	20,120,40	F,L (10,2)		5.00	133.00	640,251.00	147,332.00	11,232.00	630,132.00	3,242.00	17,892.00
r14_6	20,220,100	F,L (10,2)		1.00	7.00	27,771.00	8,148.00	8,340.00	5,322.00	4,691.00	8,268.00
r16_6	20,314,40	F,L (10,2)		3.00	68.00	333,596.00	184,709.00	7,282.00	257,111.00	3,922.00	7,652.00
r17_6	20,318,100	F,L (10,2)		1.00	2.00	14,314.00	3,167.00	3,905.00	4,090.00	3,536.00	8,412.00
r18_6	20,315,200	F,L (10,2)		1.00	1.00	2,064.00	353.00	745.00	355.00	387.00	487.00
r06_9	10,60,50	F,T (10,8)		14.00	22.00	160,482.00	171,412.00	3,252.00	405,642.00	51.00	371.00
r08_9	10,83,25	F,T (10,8)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r09_9	10,83,50	F,T (10,8)		3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
r10_9	20,120,40	F,T (10,8)		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
r14_9	20,220,100	F,T (10,8)		1.00	102.00	42,621.00	25,679.00	8,711.00	37,985.00	13,236.00	10,870.00
r15_9	20,220,200	F,T (10,8)		1.00	417.00	10,793.00	43,219.00	28,491.00	45,954.00	27,654.00	22,825.00
r16_9	20,314,40	F,T (10,8)		4.00	3.00	22,973.00	1,053.00	543.00	463.00	453.00	983.00
r17_9	20,318,100	F,T (10,8)		1.00	16.00	8,969.00	7,522.00	6,167.00	7,235.00	1,580.00	6,796.00
r18_9	20,315,200	F,T (10,8)		1.00	16.00	5,793.00	3,177.00	7,742.00	10,193.00	3,667.00	2,699.00
			Solving Nodes: Geometric Mean	2.97	15.30	2,632.77	1,552.06	717.04	1,901.38	378.15	798.65
			Variance	115.02	998.32	343,317.78	278,520.13	52,210.65	369,850.36	17,170.05	405,193.75

Results for the solution based branching rule with approach 2 (that calculates average utilization of the last copy of the arc)

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Time (sec)	MIP Gap	Total Solving Nodes
r06_6	10,60,50	F,L (10,2)	14,400.00	1.66	833,096.00
r07_6	10,82,10	F,L (10,2)	0.19	0.43	1.00
r08_6	10,83,25	F,L (10,2)	1.33	0.94	1.00
r09_6	10,83,50	F,L (10,2)	14,400.00	1.94	835,646.00
r10_6	20,120,40	F,L (10,2)	401.55	1.00	18,592.00
r14_6	20,220,100	F,L (10,2)	14,400.00	10.42	20,474.00
r16_6	20,314,40	F,L (10,2)	634.22	1.00	7,802.00
r17_6	20,318,100	F,L (10,2)	14,400.00	11.10	11,950.00
r18_6	20,315,200	F,L (10,2)	14,400.00	13.14	511.00
r06_9	10,60,50	F,T (10,8)	6.64	0.97	232.00
r08_9	10,83,25	F,T (10,8)	0.94	0.94	1.00
r09_9	10,83,50	F,T (10,8)	5.47	0.99	3.00
r10_9	20,120,40	F,T (10,8)	3.90	0.99	1.00
r14_9	20,220,100	F,T (10,8)	14,400.00	8.53	5,886.00
r15_9	20,220,200	F,T (10,8)	14,400.00	2.33	12,232.00
r16_9	20,314,40	F,T (10,8)	75.97	1.00	843.00
r17_9	20,318,100	F,T (10,8)	14,400.00	8.20	6,263.00
r18_9	20,315,200	F,L (10,8)	14,400.00	10.21	9,162.00
		Geometric Mean	370.53	2.29	823.90

APPENDIX II

RESULTS WITH GLPK

In order to analyze the influence of the SCIP solver on the performance of the branching rules, we also analyzed the Variant_B_C with GLPK solver for the existing branching rules.

Experimental Setup

We have used the default best-bound method for the node selection. The integer feasible solution is obtained for all the instances by using the SCIP solver. Time limit is set to 30 hours for obtaining this solution. This solution is then used as a heuristic primal solution while implementing various branching rules.

Different variations of the network design problem formulation are used for testing the branching rules with GLPK. These variations include formulation with and without disaggregation constraints. GLPK default branching strategy also implements the pseudo-cost based node selection strategy along with the variable selection. After selecting the branching variable with maximum pseudo-cost, this strategy further selects one of the two active branches with minimum pseudo-cost value. In order to analyze whether this pseudo-cost based node selection strategy has significant effect on performance, we also tested the variable selection methods with pseudo-cost based node selection strategy and the default best bound node selection strategy.

The statistics such as MIP Gap and the solution time needed to solve the problem instances within the specified time and MIP gap limits are used to determine which branching rule performs the best for network design problems. The methods such as reliability branching, for which the performance depends on certain parameter values, are tested for different parameter settings, and the parameter values which give the best performance for the network design problem are determined.

Parameter Settings:

The following table presents the parameter settings that we have used for our GLPK experiments.

Table 21 : Parameter settings with GLPK

Branching Rule	Parameters	Values	Description
Reliability Branching	mu	1/6	for assigning the score to candidate variables using (1- mu) * min + mu * max;
	gamma	2*gamma_avg	The iteration limit for strong branching evaluations(gamma_avg is the average number of simplex iterations per LP needed so far)
	lambda	8	maximal number of further variables evaluated without better score
	etarel	8	Reliability parameter
Pseudo-Cost Branching	mu	1/6	for assigning the score to candidate variables using (1- mu) * min + mu * max;
Branching rules based on the LP relaxation of the child nodes – (strong, entropic lookahead, reliability, pseudocost)	simplexparm2.it_lim	2*gamma_avg;	Simplex iteration limit while calculating the LP relaxation of child nodes. (gamma_avg is the average number of simplex iterations per LP needed so far)

Results for the Existing Branching Rules

Table 22 presents the summary of the results with GLPK for the formulation with disaggregation constraints and with the default (best-bound) node selection strategy considering the pseudo-

costs. Table 23 presents the computational results for the formulation with the pseudo-cost based node selection strategy. From the geometric mean of the time utilized and MIP gap presented in these results, the pseudo-cost branching and the reliability branching methods perform the best among the existing branching rules.

Table 22: GLPK formulation with disaggregation, with the default (best-bound) node selection strategy

	Best Estimate Method	Entropic Branching	Entropic Lookahead	Pseudo-Cost	Reliability	Most Fractional	Strong Branching
Time in seconds: Geometric Mean	7,387.08	14,187.99	12,201.83	**6,045.11**	6,248.78	10,009.04	7,404.05
MIP Gap: Geometric Mean	10.24	15.79	17.85	**10.45**	10.68	18.13	11.57

Table 23: GLPK formulation with disaggregation, with the pseudo-cost based node selection

	Best Estimate Method	Entropic Branching	Entropic Lookahead	Pseudo-Cost	Reliability	Most Fractional	Strong Branching
Time in seconds: Geometric Mean	7,723.20	14,172.12	12,783.17	**5,338.63**	6,957.64	10,466.93	8,945.31
MIP Gap: Geometric Mean	9.57	14.66	18.27	**9.43**	10.31	17.37	10.98

In the results presented in Appendix I, we highlight the cell that represents winner for a particular instance after considering both MIP gap, and the time utilized for solving that instance. We present our conclusions for the winning branching rule based on the average over all the instances for a particular branching rule.

Thus, we conclude from the results with GLPK that the pseudo-cost branching and the reliability branching are the 'winners' among the existing branching rule for the network design problem. The other branching rules such as strong branching, and the best estimate methods also perform remarkably well with respect to the solution time and the MIP gap.

As mentioned in Achterberg et al. (2005) [1], we also noticed the observation that most fractional branching rule is basically as good as random branching. Furthermore, the entropic

branching and the entropic look-ahead methods are not as competitive compared to other branching rules. Hence, we refrain from considering both the rules any further in our discussion and in the analysis of the branching rules with SCIP.

After determining the existing branching strategy that performs best for the network design problems, we further analyzed certain statistics for developing a new branching rule. We studied the statistics such as current and average utilization of first five variables that are branched on, average change in bounds for different utilization levels, variables that are branched on most of the times, and record of the paths from root node to node level 5. We studied if there is any relation between the variables selected for branching and the structure of the network in the particular data-set. The variations in these statistics are analyzed for the different branching rules.

Part A: With Disaggregation Constraints and with the pseudo-cost based node selection strategy:

Time Utilized:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Best Objective Estimate	Entropic Branching	Entropic Lookahead	PseudoCost WithHistory	Reliability	Most Fractional	Strong Branching
r06_6	10,60,50	F,L (10,2)	14398.42	14398.49	14290.35	14398.54	14398.3	14398.47	14399.1
r07_6	10,82,10	F,L (10,2)	295.85	14251.81	865.81	154.91	210.5	86.97	325.4
r08_6	10,83,25	F,L (10,2)	7103.41	14244.88	14277.36	14397.9	3552.24	14398.44	9085.08
r09_6	10,83,50	F,L (10,2)	14404.78	14250.14	14279.61	14398.28	14398.08	14398.02	14397.74
r10_6	20,120,40	F,L (10,2)	14402.89	14255.22	14283.47	14398.59	14411.98	14397.9	14300.64
r14_6	20,220,100	F,L (10,2)	14434	14000	14000	14000	14000	14000	14000
r15_6	20,220,200	F,L (10,2)	14000	14000	14000	14000	14000	14000	14000
r16_6	20,314,40	F,L (10,2)	14403.07	14400.7	14285.17	14398.43	14278.07	14400.76	14295.69
r17_6	20,318,100	F,L (10,2)	14000	14000	14000	14000	14000	14000	14000
r18_6	20,315,200	F,L (10,2)	14000	14000	14000	14000	14000	14000	14000
r06_9	10,60,50	F,T (10,8)	8575.79	14267.02	14398.14	819.08	3122.04	2181.02	14400.44
r07_9	10,82,10	F,T (10,8)	138.97	14252.86	14397.41	12.8	111.84	14243.96	216.35
r08_9	10,83,25	F,T (10,8)	5034.43	14259.04	14397.51	987.95	3554.23	14259.42	9320.57
r09_9	10,83,50	F,T (10,8)	1389.95	14398.77	11357.55	14255.31	691.44	14236.49	221.45
r10_9	20,120,40	F,T (10,8)	14277.88	14398.88	14413.52	14256.87	14287.4	14253.26	14404.71
r14_9	20,220,100	F,T (10,8)	14000	14000	14000	14000	14000	14000	14000
r15_9	20,220,200	F,T (10,8)	14000	14000	14000	14000	14000	14000	14000
r16_9	20,314,40	F,T (10,8)	14292.28	14401	14420.88	14260.89	14276.71	14260.29	14310.93
r17_9	20,318,100	F,T (10,8)	14000	14000	14000	14000	14000	14000	14000
r18_9	20,315,200	F,L (10,8)	14000	14000	14000	14000	14000	14000	14000
		Average Time Utilized:	7387.08	14187.99	12201.83	6045.11	6248.78	10009.04	7404.05

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Best Objective Estimate	Entropic Branching	Entropic Lookahead	PseudoCost WithHistory	Reliability	Most Fractional	Strong Branching
r06_6	10,60,50	F,L (10,2)	7.08	9.16	16.28	6.90	4.94	18.09	8.38
r07_6	10,82,10	F,L (10,2)	0.96	12.47	0.90	1.00	1.03	0.92	0.99
r08_6	10,83,25	F,L (10,2)	0.94	10.08	9.89	4.49	0.99	7.06	0.97
r09_6	10,83,50	F,L (10,2)	15.22	15.17	20.88	9.85	16.35	19.76	18.41
r10_6	20,120,40	F,L (10,2)	22.02	17.78	29.12	16.10	21.84	23.83	28.76
r14_6	20,220,100	F,L (10,2)	42.30	32.30	43.70	35.20	43.40	40.70	43.00
r15_6	20,220,200	F,L (10,2)	38.10	32.10	38.00	29.80	38.20	36.80	37.90
r16_6	20,314,40	F,L (10,2)	57.90	55.31	57.49	50.11	55.15	53.17	56.32
r17_6	20,318,100	F,L (10,2)	53.30	50.80	54.10	47.80	53.30	51.40	53.30
r18_6	20,315,200	F,L (10,2)	41.50	32.10	41.80	38.40	41.80	39.60	41.30
r06_9	10,60,50	F,T (10,8)	0.99	5.15	6.23	0.97	0.95	0.91	5.16
r07_9	10,82,10	F,T (10,8)	0.99	2.38	7.48	0.97	0.99	7.86	0.95
r08_9	10,83,25	F,T (10,8)	0.94	3.13	16.79	0.99	0.99	20.18	1.00
r09_9	10,83,50	F,T (10,8)	1.00	2.14	0.90	9.86	1.00	13.31	0.91
r10_9	20,120,40	F,T (10,8)	6.74	13.62	17.54	6.42	13.38	23.39	9.56
r14_9	20,220,100	F,T (10,8)	36.90	30.30	36.70	18.90	36.90	36.20	35.00
r15_9	20,220,200	F,T (10,8)	21.50	14.70	20.30	16.10	21.50	21.00	20.40
r16_9	20,314,40	F,T (10,8)	34.70	31.72	37.62	24.05	32.57	34.31	35.25
r17_9	20,318,100	F,T (10,8)	33.00	26.90	33.00	21.50	33.00	31.40	32.50
r18_9	20,315,200	F,L (10,8)	44.90	43.80	44.90	36.40	44.90	44.60	44.90
		Average MIP Gap:	10.24	15.79	17.85	10.45	10.68	18.13	11.57

Part B: With Disaggregation Constraints and with the default (best bound) node selection strategy:

Time Utilized:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Best Objective Estimate	Entropic Branching	Entropic Lookahead	PseudoCost WithHistory	Reliability	Most Fractional	Strong Branching
r06_6	10,60,50	F,L (10,2)	14398.42	14398.49	14290.35	14398.54	14398.30	14398.47	14399.10
r07_6	10,82,10	F,L (10,2)	295.85	14251.81	865.81	154.91	210.50	86.97	325.40
r08_6	10,83,25	F,L (10,2)	7103.41	14244.88	14277.36	14397.90	3552.24	14398.44	9085.08
r09_6	10,83,50	F,L (10,2)	14404.78	14250.14	14279.61	14398.28	14398.08	14398.02	14397.74
r10_6	20,120,40	F,L (10,2)	14402.89	14255.22	14283.47	14398.59	14411.98	14397.90	14300.64
r14_6	20,220,100	F,L (10,2)	14434.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r15_6	20,220,200	F,L (10,2)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r16_6	20,314,40	F,L (10,2)	14403.07	14400.70	14285.17	14398.43	14278.07	14400.76	14295.69
r17_6	20,318,100	F,L (10,2)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r18_6	20,315,200	F,L (10,2)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r06_9	10,60,50	F,T (10,8)	8575.79	14267.02	14398.14	819.08	3122.04	2181.02	14400.44
r07_9	10,82,10	F,T (10,8)	138.97	14252.86	14397.41	12.80	111.84	14243.96	216.35
r08_9	10,83,25	F,T (10,8)	5034.43	14259.04	14397.51	987.95	3554.23	14259.42	9320.57
r09_9	10,83,50	F,T (10,8)	1389.95	14398.77	11357.55	14255.31	691.44	14236.49	221.45
r10_9	20,120,40	F,T (10,8)	14277.88	14398.88	14413.52	14256.87	14287.40	14253.26	14404.71
r14_9	20,220,100	F,T (10,8)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r15_9	20,220,200	F,T (10,8)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r16_9	20,314,40	F,T (10,8)	14292.28	14401.00	14420.88	14260.89	14276.71	14260.29	14310.93
r17_9	20,318,100	F,T (10,8)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
r18_9	20,315,200	F,L (10,8)	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00	14000.00
		Average Time Utilized:	7387.081247	14187.993	12201.8277	6045.111151	6248.78	10009.042	7404.0476

MIP Gap:

Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	Best Objective Estimate	Entropic Branching	Entropic Lookahead	PseudoCost WithHistory	Reliability	Most Fractional	Strong Branching
r07_6	10,82,10	F,L (10,2)	0.98	9.30	0.90	0.97	0.98	0.97	0.97
r08_6	10,83,25	F,L (10,2)	0.94	7.21	8.50	2.32	1.00	4.76	0.97
r09_6	10,83,50	F,L (10,2)	12.59	13.82	20.40	13.32	15.86	17.98	16.54
r10_6	20,120,40	F,L (10,2)	17.76	14.77	23.31	17.08	17.79	20.87	17.64
r14_6	20,220,100	F,L (10,2)	39.20	36.70	43.00	34.50	40.10	37.90	43.00
r15_6	20,220,200	F,L (10,2)	38.60	28.00	38.50	34.30	37.90	35.60	38.20
r16_6	20,314,40	F,L (10,2)	52.54	51.99	54.34	47.69	51.67	49.42	51.34
r17_6	20,318,100	F,L (10,2)	52.00	48.50	53.00	46.40	51.30	48.70	52.80
r18_6	20,315,200	F,L (10,2)	41.50	34.60	42.00	37.40	41.00	38.50	41.90
r06_9	10,60,50	F,T (10,8)	1.00	3.56	3.72	1.00	0.91	0.91	1.00
r07_9	10,82,10	F,T (10,8)	0.99	2.07	6.48	1.00	0.91	7.06	1.00
r08_9	10,83,25	F,T (10,8)	1.00	3.20	13.88	1.00	0.97	19.45	1.00
r09_9	10,83,50	F,T (10,8)	1.24	3.93	9.11	6.03	2.88	16.48	4.37
r10_9	20,120,40	F,T (10,8)	5.13	12.43	10.89	8.05	5.09	23.18	8.80
r14_9	20,220,100	F,T (10,8)	28.90	29.30	36.60	33.50	36.60	35.70	36.60
r15_9	20,220,200	F,T (10,8)	20.30	13.30	20.80	1.00	21.10	21.90	20.80
r16_9	20,314,40	F,T (10,8)	26.23	27.85	33.40	31.66	29.26	32.66	26.07
r17_9	20,318,100	F,T (10,8)	28.30	22.50	33.00	28.90	32.10	30.40	33.10
r18_9	20,315,200	F,L (10,8)	44.90	41.30	46.10	44.80	46.00	45.10	46.10
		Average MIP gap	9.57	14.66	18.27	9.43	10.31	17.37	10.98

APPENDIX III

Results for analyzing the effect of quality and quantity of the feasible solutions:

The following tables present the results for the solution based approach for all the tested MIP gap ranges and quantity of feasible solutions. We were unable to collect the feasible solutions in all the MIP gap ranges for some datasets. Hence, the results for such occurrences are blank (represented by '-') in the following tables.

2000 feasible solutions

Results with 2000 feasible solutions			TIME						MIP Gap				
Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	0 to 10	10 to 20	20 to 30	30 to 40	40 to 50		0 to 10	10 to 20	20 to 30	30 to 40	40 to 50
r06_6	10,60,50	F,L (10,2)	393.63	481.05	388.75	573.17	1879.99		1	0.99	1	1	1
r07_6	10,82,10	F,L (10,2)	3.85	3.71	3.8	3.1	6.15		0.89	0.96	0.96	0.99	0.96
r08_6	10,83,25	F,L (10,2)	8.52	-	9.41	8.69	11.47		0.95	-	0.92	0.92	0.95
r09_6	10,83,50	F,L (10,2)	449.72	725.46	454.06	199.28	755.17		1	1	1	1	1
r10_6	20,120,40	F,L (10,2)	-	1154.03	847.4	2215.24	3865.97		-	1	1	1	1
r16_6	20,314,40	F,L (10,2)	-	-	-	14400	14400		-	-	-	1.63	1.07
r06_9	10,60,50	F,T (10,8)	8.11	10.7	12.69	8.98	14.64		0.97	0.95	0.99	0.87	0.93
r07_9	10,82,10	F,T (10,8)	30.49	36.84	47.24	37.62	15.87		1	1	1	1	0.99
r08_9	10,83,25	F,T (10,8)	90.96	262.5	343.51	425.94	337.42		1	1	1	1	1
r09_9	10,83,50	F,T (10,8)	33.34	41.28	67.95	57.09	56.19		1	0.85	1	0.99	0.99
r10_9	20,120,40	F,T (10,8)	2681.1	8145.28	11585	14316.5	14400		1	1	1	1	1.2
r14_9	20,220,100	F,T (10,8)	-	-	-	14400	14400		-	-	-	4.6	5.42
r15_9	20,220,200	F,T (10,8)	218.47	-	258.16	473.64	-		0.45	-	0.45	0.45	-
r18_9	20,315,200	F,L (10,8)	-	-	-	-	14403.8		-	-	-	-	5.79

500 feasible solutions

Results with 500 feasible solutions			TIME						MIP Gap				
Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	0 to 10	10 to 20	20 to 30	30 to 40	40 to 50		0 to 10	10 to 20	20 to 30	30 to 40	40 to 50
r06_6	10,60,50	F,L (10,2)	360.04	508.95	555.38	906.52	1270.86		0.99	1	1	1	1
r07_6	10,82,10	F,L (10,2)	4.27	4.09	2.95	4.06	3.86		0.95	0.96	0.96	0.86	0.96
r08_6	10,83,25	F,L (10,2)	11.31	-	7.39	9.23	9.58		0.95	-	0.96	0.84	0.9
r09_6	10,83,50	F,L (10,2)	687.33	802.47	269.93	365.36	463.1		1	1	1	1	1
r10_6	20,120,40	F,L (10,2)	-	1191.38	694.82	3149.86	3109.92		-	0.94	1	1	1
r16_6	20,314,40	F,L (10,2)	-	-	-	14400	14400		-	-	-	4.26	1.05
r06_9	10,60,50	F,T (10,8)	10.96	7.42	11.84	8.24	10		0.9	0.95	0.99	0.83	0.93
r07_9	10,82,10	F,T (10,8)	42.04	24.17	34.15	39.37	20.17		1	1	1	1	0.99
r08_9	10,83,25	F,T (10,8)	89.29	380.68	478.63	287.09	281.27		1	1	1	1	1
r09_9	10,83,50	F,T (10,8)	31.43	35.74	47.82	42.4	32.02		0.84	1	1	0.99	0.95
r10_9	20,120,40	F,T (10,8)	4712.8	14400	6014.1	14400	14400		1	1.27	1	1.1	1.09
r14_9	20,220,100	F,T (10,8)	-	-	-	14400	14400		-	-	-	0.45	5.03
r15_9	20,220,200	F,T (10,8)	380.99	-	155.02	218.43	-		0.45	-	0.45	1.45	-
r18_9	20,315,200	F,L (10,8)	-	-	-	-	14400		-	-	-	-	4

100 feasible solutions

Results with 100 feasible solutions			TIME						MIP Gap					
Dataset	(N,A,K)	(Fixed Cost rank, Capacity rank) (1,5,10) , (1,2,8)	0 to 10	10 to 20	20 to 30	30 to 40	40 to 50		0 to 10	10 to 20	20 to 30	30 to 40	40 to 50
r06_6	**10,60,50**	**F,L (10,2)**	465.25	674.2	638.6	1269.2	2592.38		1	1	1	1	1
r07_6	**10,82,10**	**F,L (10,2)**	2.51	3.05	3.08	3.22	3.79		0.96	0.95	0.95	0.99	0.96
r08_6	**10,83,25**	**F,L (10,2)**	13.95	-	11.01	9.91	8.81		0.86	-	0.92	0.65	0.94
r09_6	**10,83,50**	**F,L (10,2)**	1256.15	391.69	231.54	181.16	358.91		1	1	1	1	1
r10_6	**20,120,40**	**F,L (10,2)**	-	887.48	1221.02	3052.37	2700.97		-	0.88	1	1	1
r16_6	**20,314,40**	**F,L (10,2)**	-	-	-	14400	14400		-	-	-	1.01	4.23
r06_9	**10,60,50**	**F,T (10,8)**	6.86	11.13	8.86	6.38	9.79		0.92	0.96	0.99	1	0.97
r07_9	**10,82,10**	**F,T (10,8)**	25.88	37.33	30.48	44.79	15.26		1	1	1	1	0.99
r08_9	**10,83,25**	**F,T (10,8)**	144.35	459.61	286.88	263.59	266.01		1	1	1	1	1
r09_9	**10,83,50**	**F,T (10,8)**	35.6	48.95	62.77	52.07	20.99		0.97	1	0.99	0.99	0.98
r10_9	**20,120,40**	**F,T (10,8)**	5260.31	14400	12952.7	14400	14400		1	1.8	1	1.14	1.07
r14_9	**20,220,100**	**F,T (10,8)**	-	-	-	14400.2	14403.8		-	-	-	4.18	1.01
r15_9	**20,220,200**	**F,T (10,8)**	440.11	-	232.14	377.65	-		0.45	-	0.45	0.45	-
r18_9	**20,315,200**	**F,L (10,8)**	-	-	-	-	14400		-	-	-	-	5.76

APPENDIX IV

File Formats

1. Format of the data-files

The format of the data files that we used for all the instances is as follows:

- number_of_nodes, number_of_arcs, number_of_commodities
- for each arc:
  from_node, to_node, variable_cost, capacity, fixed_cost, any_number, any_number
- for each commodity:
  from_node, to_node, demand (>0)

2. Format of the files containing the set of feasible solutions

We have collected the set of feasible solutions in the file that represent each arc and the total flow of all the commodities on each arc.

- Index of the variable representing arc: total flow on that arc

3. Format of the files containing the primal heuristic solutions

There is separate file containing the primal heuristic solution for each of the tested data-sets. These files contain the values of all the variables in a particular solution.

Implementation details for SCIP

Collecting feasible solutions:

For implementing the solution based approach of variable selection, we initially collected the pool of feasible solutions as described above for each variant of the network design problem. For this, we initially implemented the formulation of the MIP in SCIP.

We can enumerate the feasible solutions of a given mixed integer program with the methods available in SCIP. These methods return the values of all the variables that are active in the current sub-problem. In addition to these active variables, there can also be some aggregated, multi-aggregated, and the fixed variables present in the sub-problem whose values are not reflected in the solutions returned by SCIP methods. Thus for obtaining the complete solution, we turned the aggregation and multi-aggregation of the variables off for our experiments. The methods for obtaining the values and the indices of the fixed variables are available in SCIP. Using this data, we inserted the values of the fixed variables at the appropriate positions in order to obtain the values of all variables in the collected feasible solutions. Thus, we created the final solution files has the data that represents the total flow on each arc for each feasible solution.

We can also specify the number of feasible solutions with the parameter "constraints/countsols/sollimit". Furthermore, we can specify the MIP gap range for these feasible solutions by specifying the bounds on the objective function as presented in the section 7.1.

Adding a user-defined branching rule

We have added our own branching rules by defining the plug-ins 'branch_mybranchingrule.h' and 'branch_mybranchingrule.c'. We need to make sure that we adjust the makefile such that these files are compiled and linked to our main project. SCIP provides some interface methods that are responsible for notifying SCIP that the new branching rule is available along with the default list of branching rules. SCIP also provides various callback methods for defining and using the user-defined branching rules. The most important callback method BRANCHEXECLP that we have used performs the actual task of generating the branching. This callback method is executed while branching at the time of node processing if a fractional LP solution is available.

Assigning the variable specific data

The SCIP solver provides the methods that we can use to handle the data specific to the problem variables. This data is assigned to the variables at the time of variable creation. We have used the plug-ins ‘scipvardata.h’ and ‘scipvardata.c’ in our implementation for assigning the variable data. We can retrieve this variable specific data anytime during the solving process. Hence in our implementation, we have stored the average utilization of each arc as the variable data corresponding to the particular arc. We then use this data while implementing the solution based branching rule.

Selecting a particular existing branching rule and separator (cutting planes)

SCIP solver has some existing branching rules implemented in the source-code. We can select a particular branching rule by increasing the priority value for a particular branching rule. Similarly, a particular separator can be selected by increasing the priority value.