A Feasible Lagrangian Approach with Application to the Generalized Assignment Problem

by

Shahroz Saleem Punjwani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Lagrangian relaxation is a widely used decomposition approach to solve difficult optimization problems that exhibit special structure. It provides a lower bound on the optimal objective of a minimization problem. On the other hand, an upper bound and quality feasible solutions may be obtained by perturbing solutions of the subproblem. In this thesis, we enhance the Lagrangian approach by using information at the subproblem to push for feasibility to the original problem. We exploit the idea that if the solution for the subproblem is pushed towards feasibility to the original problem, it may lead to improved lower bounds as well as good feasible solutions. Our proposed strategy is to solve the subproblem repeatedly at each iteration of the Lagrangian procedure and strengthen it with valid inequalities. As cuts are added to the subproblem, it inevitably becomes harder to solve. We propose to solve it under a time limit and adjust the Lagrangian bound accordingly. Two variants of the approach are explored that we call a Modified Lagrangian approach and a Feasible Lagrangian approach.

We use the Generalized Assignment Problem for testing. We develop two methodologies based on minimal covering inequalities. The first solves the subproblem repeatedly for a given number of iterations and generates minimal cover inequalities that are either discarded or passed on to subsequent Lagrangian iterations. The second starts with initial multipliers and repeatedly solves the subproblem until a feasible solution is attained. At that point, the regular Lagrangian approach is used to find a lower bound. We test on GAP instances from the literature and compare the lower bound to the Lagrangian bound and the feasible solution to the best known solution in the literature. The results demonstrate that the proposed feasible Lagrangian approach leads to improved lower bounds and good quality feasible solutions.

Acknowledgements

I would like to thank my supervisors Professor Gzara and Professor Elhedhli for their unlimited support and guidance throughout the program, and for making this thesis possible.

Table of Contents

List of Figures vii

List of Tables viii

1 Introduction 1

2 Background and Literature Review 4

2.1 Lagrangian Relaxation 4

2.2 Benders and Cross Decomposition 6

2.3 Generalized Assignment Problem (GAP) 8

2.4 Applications of GAP 9

2.5 Lagrangian Relaxation and GAP 11

3 Lagrangian Relaxations for GAP 15

3.1 Relaxing the Capacity Constraints 16

3.2 Relaxing the Assignment Constraints 18

4 Improved Lagrangian Approach **19**

4.1 General Framework 19

4.2 Application to the GAP 21

4.2.1 Minimal Cover Inequalities 22

4.3 Proposed Lagrangian approaches 24

4.3.1 Modified Lagrangian approach 24

4.3.2 Feasible Lagrangian approach 28

5 Computational Results **30**

5.1 Data 30

5.2 Numerical Results 31

5.2.1 Results of Modified Lagrangian approach with cut accumulation 34

5.2.2 Results of Modified Lagrangian approach with restart 39

5.2.3 Feasible Lagrangian approach 43

5.3 Results comparison of the proposed methodologies 45

5.4 Results Summary 48

6 Conclusion **49**

References **51**

References 51

List of Figures

4.1 Modified Lagrangian approach 27

4.2 Feasible Lagrangian approach 29

5.1 Comparison of results from different approaches for small instances . 45

5.2 Comparison of results from different approaches for medium instances 47

List of Tables

5.1 Results of the cut accumulation approach on small instances 35

5.2 Results of cut accumulation approach on medium instances, $k = 20$ and $tLim = 5$ seconds 36

5.3 Results of the cut accumulation approach on medium instances, $k = 20$ and $tLim = 10$ seconds 37

5.4 Results of the cut accumulation approach on medium instances, $k = 40$ and $tLim = 5$ seconds 38

5.5 Results of the restart approach on small instances 40

5.6 Results of the restart approach on medium instances, $k = 20$ and $tLim = \{5,10\}$ 41

5.7 Results of the restart approach on medium instances, $k = 30$ and $tLim = \{5,10\}$ 42

5.8 Results of the Feasible Lagrangian approach on small instances 43

5.9 Results of the Feasible Lagrangian approach on medium instances . . 44

Chapter 1

Introduction

Optimization problems are broadly categorized as "easy" problems, which can be solved efficiently and "hard" problems, which require exponential time in the worst case. Most of the problems fall into the latter category. In 1970s, there was a breakthrough in solving the hard combinatorial problems when it was observed that some of these were actually easy problems, complicated by a relatively small set of constraints. It was proposed that dualizing these complicating constraints produces a Lagrangian subproblem, which is easier to solve and provides a bound on the optimal value of the original problem. This is known as Lagrangian Relaxation (LR) (Held and Karp, 1970). It was also noted that for many problems, LR provided better bounds than the linear relaxation. Hence, this meant that LR could now be used to provide better bounds in branch and bound approaches. Although this was the most obvious use of LR, over the time it has been applied in many heuristic approaches to find good feasible solutions by perturbing the solution of the subproblem.

In this thesis, our goal is to exploit Lagrangian relaxation in an approach that combines both the strengths of LR, i.e., providing tighter bounds on the original

problem and good feasible solutions. Hence in general, the proposed approach is motivated by the idea that if the solution for the Lagrangian subproblem (SP) can be pushed towards feasibility for the original problem, it can result in improved LR bounds as well as quality feasible solutions. The strategy we propose here is to push SP solution towards feasibility by adding valid cuts based on SP solutions to further push solutions closer to feasibility for the original problem. By incorporating this idea into the Lagrangian methodology, we hope to find improved bounds and obtain good feasible solutions.

To test this approach, we select the Generalized Assignment Problem (GAP), which is known to be NP-hard. It has a readily apparent structure for the application of LR. The problem has been widely studied in the literature because of its several real-life applications e.g. transportation and routing, production and planning, and location problems, and many exact and heuristic methodologies have been proposed to solve it. To test our approach we develop two methodologies for the GAP with the objectives of improving bounds and obtaining good feasible solutions. In these methods, we use minimal cover inequalities to push SP solutions towards feasibility for the GAP. For the first approach, we test two variants, with accumulation and without accumulation of minimal cover inequalities. In the first variant, the cover inequalities are allowed to accumulate from one LR iteration to the next. In the second variant, the inequalities are not carried forward to the following iteration. We keep adding the minimal cover inequalities to SP until either SP solution is feasible for the GAP, or a certain number of cover inequalities generation iterations, k, has been completed. As SP becomes more difficult to solve because of the cover inequalities, we impose a time limit, $tLim$, on the solution of SP. In the second methodology we initialize the values of the Lagrangian multipliers with the optimal dual solution of the relaxed constraints obtained by solving the linear programming

relaxation of the GAP. We then keep adding the minimal cover inequalities to SP until a feasible solution to the GAP is obtained. Then SP, with all the minimal cover inequalities added up till now, and MP are solved iteratively in a regular Lagrangian fashion to obtain the optimal Lagrangian bound. In this approach, SP is solved to optimality without any limit on the solution time. The results from testing these methodologies on benchmark GAP data sets demonstrate that adding minimal cover inequalities in SP leads to improved bounds and quality feasible solutions.

The remainder of the thesis is organized as follows. In Chapter 2, we introduce the GAP and review the literature on the real-life applications of GAP in different domains. In this chapter we also discuss the Lagrangian relaxation methodology and review the literature on the application of Lagrangian relaxation in the exact and heuristics approaches for solving the GAP. In Chapter 3, we present the mathematical formulation for the GAP and its two different Lagrangian relaxations. In Chapter 4, a detailed description of our methodologies is presented. The results from the numerical testing of these methods and their analysis are presented in Chapter 5.

Chapter 2

Background and Literature Review

In this chapter, we explain the Lagrangian relaxation (LR) approach and review similar decomposition approaches. We then introduce the Generalized Assignment Problem (GAP), present its mathematical formulation and discuss some practical applications of the GAP. Lastly, we review the literature on the application of LR in developing exact and heuristic solution methodologies for GAP.

2.1 Lagrangian Relaxation

The Lagrangian approach was introduced by Held and Karp (1970), who used it based on minimum spanning trees to devise an algorithm for the traveling salesman problem. Since then Lagrangian relaxation has been used to tackle some of the most challenging combinatorial optimization problems (Fisher, 2004). It has been observed that many of the difficult integer problems can be reduced to easier problems if a relatively small set of complicating constraints is removed. Hence, LR exploits this fact by removing the complicating side constraints and introducing them in the cost

function with corresponding penalty. To illustrate the LR approach, consider the optimization problem:

$$\begin{aligned} [P]: \quad \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & Bx \geq d \\ & x \geq 0 \\ & x_j \; integer \; \forall j \in J \end{aligned}$$

Let us assume that $Ax \geq b$ are the complicating constraints. By relaxing them in a Lagrangian fashion with a set of Lagrangian multipliers $\lambda \geq 0$, we get the subproblem (SP):

$$\begin{aligned} SP(\lambda): \quad \min \quad & c^T x + \lambda^T (b - Ax) \\ \text{s.t.} \quad & Bx \geq d \\ & x \geq 0 \\ & x_j \; integer \; \forall j \in J \end{aligned}$$

SP(λ) provides a lower bound on the objective value of the original minimization problem, $z_{SP}(\lambda)$, for a given set of Lagrangian multipliers λ:

$$z_{SP}(\lambda) \leq z*$$

To find the best possible lower bound overall, the following Lagrangian dual problem is solved:

$$[LD]: \quad \max_{\lambda \geq 0} \; z_{SP}(\lambda)$$

which can be formulated as a linear program, known as the Master problem:

$$
\begin{aligned}
[MP]: \quad \max \quad & \theta \\
\text{s.t} \quad & \theta \leq cx^t + \lambda^t(b - Ax^t) \qquad \forall t \in H \\
& \lambda \geq 0
\end{aligned}
$$

where H is the index set of feasible solutions to:

$$\{x: \ Bx \geq d, \ x \geq 0, \ x_j \ integer, \ \forall j \in J\}$$

which is assumed to be bounded. The Lagrangian master problem can be solved using the cutting plane or the subgradient method (Fisher, 1985).

2.2 Benders and Cross Decomposition

Benders decomposition is an exact approach to solve the mixed integer programming problems (MIP) proposed by Benders in 1962. Similar to Lagrangian relaxation which is based on the idea of complicating constraints, Benders decomposition is based on the notion of complicating variables. These are the variables which if fixed, render the remaining problem a linear program, parameterized by the value of complicating variables. The first-stage master problem is solved for the complicating variables, and the values of the remaining variables are determined by a second-stage subproblem given the values of the first-stage variables. If the subproblem determines that the proposed first-stage decisions are infeasible, then one or more constraints are generated and added to the master problem, which then is solved again.

As opposed to Lagrangian relaxation which is a dual decomposition method where the complicating constraints are dualized and the Lagrangian multipliers are obtained

from the dual master problem, Benders decomposition is a primal decomposition method because the primal or Benders subproblem restricts the original problem by fixing the values of some primal variables. Hence both approaches exploit either the primal or the dual substructure of the problem. But there are many problems which may have both easy to solve primal and dual subproblems. For example, the capacitated facility location problem reduces to the transportation problem when the binary location variables are fixed or to continuous knapsack problems for each facility when the assignment constraints are relaxed. Based on this idea, Van Roy (1983) proposes the cross-decomposition approach i.e. to use both subproblems in one single decomposition procedure. This approach starts with Lagrangian subproblem and an initial set of Lagrangian multipliers. But based on the results of the convergence tests on the subproblem(s) solutions, the algorithm later switches between Benders decomposition and Lagrangian relaxation. In his other paper, Van Roy (1986) applies the cross decomposition approach to solve the capacitated facility location problem. Ogbe and Li (2016) propose an exact solution approach to improve the solution of scenario based two-stage stochastic MIPs using the cross decomposition method by combining the Benders Decomposition (BD) and Dantzig/Wolfe decomposition (DWD). Their method switches between DWD iterations and BD iterations, where DWD restricted master problems and BD primal problems produce a sequence of upper bounds and BD relaxed master problems yield a sequence of lower bounds.

In the next section, we discuss the Generalized Assignment Problem (GAP), present its mathematical formulation and review the real-life applications of the GAP discussed in the literature.

2.3 Generalized Assignment Problem (GAP)

The Generalized Assignment Problem (GAP) is a well-known problem in operations research. The goal is to minimize (maximize) the cost (profit) of assigning n items to m knapsacks while ensuring that every item is assigned to exactly one knapsack and that the capacity of each knapsack is not violated. In the literature, the GAP has also been discussed using the scheduling terminology i.e. jobs, machines, and processing times. The GAP formulation is as follows:

$$[GAP] \qquad \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

$$\text{s.t} \quad \sum_{j=1}^{n} w_{ij}x_{ij} \leq b_i \qquad \forall i \in I \tag{2.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in J \tag{2.3}$$

$$x_{ij} \in \{0,1\} \tag{2.4}$$

where c_{ij} is the cost of assigning item j to knapsack i and w_{ij} is the capacity of knapsack i consumed by item j. x_{ij} is a binary decision variable that takes value 1 if item j is assigned to knapsack i, or 0 otherwise. Constraints (2.2) ensure that the capacity b_i of each knapsack i is not violated and constraints (2.3) ensure that each item j is assigned to exactly one knapsack i. The classical assignment problem with one-to-one assignment of items to the knapsacks can be solved in polynomial time, however GAP is an NP-hard problem (Sahni and Gonzalez, 1976).

2.4 Applications of GAP

Several real-life applications of the GAP have been discussed in the literature. The GAP has been widely applied to scheduling problems such as employee scheduling, machine scheduling, workforce planning, etc. Within this domain, another extension of GAP is the Load Balancing Problem (Harvey et al. 2006). This problem can have different types of objective functions such as minimization of makespan, or the average flow-time, or the maximization of fairness in job assignment.

In the transportation and routing applications, the GAP appears as a sub-problem in the well-known Vehicle Routing Problem. Fisher and Jaikumar (1981) have proposed a generalized assignment approach for the solution of the VRP. Foulds and Wilson (1997) propose heuristic algorithms for a variation of the GAP arising in the New Zealand dairy industry: the Covering Assignment Problem (CAP). The CAP aims to minimize transportation cost from farms (suppliers) to factories (customers), such that each farm is assigned to exactly one factory and the demand of each factory must be satisfied.

Bressoud et al. (2003) discuss the application of the GAP in telecommunication for the optimal configuration for Border Gateway Protocol (BGP) which are crucial in controlling transit traffic flows from customers and providers. Barbas and Marin (2004) introduce another telecommunication application as an extension of the GAP, that is the maximal covering code multiplexing access telecommunication networks with power and flow capacity constraints.

There are many applications of the GAP in production planning as well. Dobson and Nambimadom (2001) divide the NP-Hard Batch Loading and Scheduling Problem into two nested problems: batch-loading (BLP) and batch-scheduling. Authors show that BLP, which assigns jobs to the batches given a sequence, is a special case

of the GAP. A vast application area of the GAP arises in Group Technology where efficient partitioning of manufacturing parts into families, namely the Group Formation Problem (GFP), plays a crucial role in the overall system performance. Shtub (1989) has shown that the GFP and its extension, the Generalized Group Formation Problem (GGFP), are equivalent to the GAP.

For location problems, Ross and Soland (1977) have shown that a set of public and private facility location problems such as the p-Median Problem, Capacitated p-Median Problem, Uncapacitated Facility Location Problem, Capacitated Facility Location Problem and Facility Location Problem with a choice of facility type can be modeled as a GAP. As an extension of the paper by Ross and Soland (1977), Klastorin (1979) has noted how the Maximal Covering Location Problem can also be formulated as GAP.

There are many other interesting works on the applications of GAP in other domains. Nowakovski et al. (1999) formulated the problem of scheduling the tasks during a half year mission phase of the international spatial telescope ROSAT as a GAP. The problem being the assignment of objects to be observed to time intervals (slots) with the objective of maximizing the observation time for all objects. Gavish and Pirkul (1986) have addressed the problem of partitioning of the central database in a distributed computer system such that the total communication cost is minimized and the processing, storage and communication capacities of the processors are not violated. This problem arises in the banking networks (Boffey, 1989) and is formulated as a special case of the GAP.

Some recent applications of GAP include the work by Luo et al. (2015), that presents distributed algorithms for multi-robot task assignment. As each robot has a limited battery life, the tasks have to be completed within given deadlines, giving an upper limit on the amount of time available to perform the tasks. They formulate

this problem as a special case of GAP with additional task deadline constraints.

Shi and Hong (2011) formulate a multi-level GAP (MGAP) for virtual machine (VM) placement in a data center. Maintaining such large-scale infrastructures for data centers consumes an enormous amount of energy and imposes a large carbon footprint on the environment. Hence for their problem, MGAP seeks to maximize the profit under the service level agreement and power budget constraints. The MGAP extends GAP by allowing the agents to perform tasks at different efficiency levels.

Cohen et al. (2015) present an application of GAP to Network Function Virtualization (NFV) location problem in computer networking. They are the first to address this problem of locating network functions to servers such that the overall network cost is minimized, while adhering to the allocation size of the servers. They formulate this problem as a combination of GAP and facility location problem.

2.5 Lagrangian Relaxation and GAP

As the GAP was shown to be NP-Hard, there is no polynomial time algorithm which finds a feasible solution to an arbitrary instance of the GAP. The exact solution approaches in the literature only deal with small-sized instances. To overcome the limitations of exact methods a lot of work has been done in devising several heuristic and metaheuristic approaches. The benefits of heuristics are twofold; they can be used as stand-alone algorithms to obtain good solutions within reasonable time and they can also be used to obtain upper bounds in exact solution methods such as the branch-and-bound procedure. One such approach that has been widely used in exact and heuristics methods for the GAP is Lagrangian Relaxation. Following sections review the existing work on the application of Lagrangian relaxation in both exact and heuristic methodologies for the GAP.

Exact Algorithms

Guignard and Rosenwein (1989) propose an enhanced Lagrangian dual ascent procedure for GAP that solves a Lagrangian dual at each enumeration node. They obtain the subproblem by relaxing the assignment constraints. In case of violation of assignment constraints at any enumeration node, a surrogate constraint is added to the subproblem to strengthen the Lagrangian bound.

Nauss (2003) considers a minimization version of the GAP and devises a branch and bound algorithm to solve it. To increase the lower bounds he uses linear programming cuts, Lagrangian relaxation, penalties, feasibility based tests, and logical feasibility tests. Nauss has tried to decrease the upper bound by means of feasible solution generators such as tabu-search and Lagrangian heuristic to obtain an initial feasible solution and complete enumeration to improve the current solution.

Haddadi and Ouzia (2004) consider a maximization case of the GAP and employ a breadth-first search strategy in their branch and bound algorithm. In the upper bounding procedure, a Lagrangian relaxation is obtained by dualizing the semi assignment constraints at each node of the decision tree. The relaxed problem then separates into m independent knapsack problems and a standard subgradient method is used to solve for the Lagrangian multipliers. As a lower bounding procedure, Haddadi and Ouzia transform the solution of the Lagrangian relaxation into a feasible assignment.

Pessoa et al. (2010) propose two exact algorithms for GQAP (generalized quadratic assignment problem), where the cost function is quadratic. The heuristics combine a previously proposed branch-and-bound scheme with a new Lagrangian relaxation procedure over a known RLT (Reformulation-Linearization Technique) formulation. They also apply transformational lower bounding techniques to improve the perfor-

mance of the new procedure.

Lagrangian Relaxation Based Heuristics

Lorena and Narciso (1999) propose Lagrangian/surrogate relaxation to the GAP. They derive three relaxations by relaxing the assignment, capacity and variable splitting constraints respectively. First, a set of constraints is relaxed in a surrogate way and then the Lagrangian relaxation of the surrogate constraint is obtained. They show that Lagrangian/surrogate can give improved local bounds at the application of a subgradient method, resulting in less computational times. The relaxation multipliers are used with constructive heuristics to find good feasible solutions.

Haddadi (1999) proposes a Lagrangian relaxation decomposition based heuristic in which a substitution of variables is performed and the constraints defining the substituted variables are dualized using Lagrangian relaxation. The problem decomposes into a set of knapsack problems and a transportation problem, which can be solved in polynomial time. The author has followed a modified version of the heuristic by Mazzola (1989).

Haddadi and Ouzia (2001) propose another Lagrangian heuristic which finds and improves feasible GAP solutions. They dualize the capacity constraints and use the subgradient optimization procedure. At each iteration of the Lagrangian heuristic, the heuristic approach by Martello and Toth (1981) is used to find the initial assignment. Then iterative feasibility and solution improvement heuristics by Mazzola (1989) and a three-way interchange procedure is used to obtain a feasible improved solution.

Jeet and Kutanoglu (2007) propose a Lagrangian relaxation based problem space search heuristics where they combine both iterative search capability of the subgra-

dient optimization and problem space search scheme by perturbation. They dualize capacity constraints and use the subgradient optimization procedure. To find feasible solutions they propose three feasibility restoration heuristics and use Problem Space Search (PSS) metaheuristic technique to increase the chance of obtaining feasible solutions from their heuristics.

Litvinchev et. al. (2010) propose a procedure to tighten the Lagrangian bounds based on the search for a tighter estimation of the penalty term arising in the Lagrangian problem. The new bounds are illustrated by a small example and studied numerically for a class of the generalized assignment problems which can be characterized as a many-to-many assignment.

Mazzola and Neebe (2012) consider a generalization of the GAP over discrete time periods encompassed within a finite planning horizon. The resulting model, MultiGAP is solved using Lagrangian relaxation based heuristic as well as a branch and bound algorithm.

Litvinchev et.al (2017) propose modified Lagrangian bounds for the GAP. The approach is based on a double decomposable structure of the formulation. A family of greedy heuristics is considered to get Lagrangian based feasible solutions. Numerical results for problem instances with the number of agents close to the number of tasks are provided.

Our main contribution is to propose an improved Lagrangian approach that makes use of the information from the subproblem solutions to generate valid cuts. These cuts are added back to the subproblem to strengthen it and to push its solutions towards feasibility to the original problem. This leads to feasible solutions for the original problem as well as improved bounds due to more information being added to the subproblem.

Chapter 3

Lagrangian Relaxations for GAP

In this section, we derive the two possible Lagrangian relaxations and their respective master problems for the GAP. Revisiting the GAP formulation presented in section 2.2:

$$[GAP] \qquad \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3.1}$$

$$\text{s.t} \quad \sum_{j=1}^{n} w_{ij}x_{ij} \leq b_i \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in J \tag{3.3}$$

$$x_{ij} \in \{0,1\} \tag{3.4}$$

Based on the formulation above, the first LR is obtained by relaxing the capacity constraints (3.2), while the second is obtained by relaxing the assignment constraints (3.3). In the following sections, we present Lagrangian relaxations for both relaxations and derive their respective subproblems and master problems.

3.1 Relaxing the Capacity Constraints

By relaxing the capacity constraints with Lagrangian multipliers $\lambda_i \geq 0, i \in I$, we get the following Lagrangian subproblem:

$$
\begin{aligned}
[SP]: \qquad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+\lambda_i w_{ij})x_{ij} - \sum_{i=1}^{m}\lambda_i b_i & \\
\text{s.t} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \forall j \in J \\
& x_{ij} \in \{0,1\} &
\end{aligned}
$$

which can be written as:

$$
\begin{aligned}
[SP]: \qquad z_{SP} = \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+\lambda_i w_{ij})x_{ij} & \\
\text{s.t} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \forall j \in J \\
& x_{ij} \in \{0,1\} &
\end{aligned}
$$

Note that the SP is further decomposable by items $j \in J$, leading to n subproblems:

$$
\begin{aligned}
[SP_j]: v_j = \qquad \min \quad & \sum_{i=1}^{m}(c_{ij}+\lambda_i w_{ij})x_{ij} \\
\text{s.t} \quad & \sum_{i=1}^{m} x_{ij} = 1 \\
& x_{ij} \in \{0,1\}
\end{aligned}
$$

This problem can be easily solved by determining the knapsack i with $\min(c_{ij} + \lambda_i w_{ij})$ for each item j and setting the corresponding x_{ij}=1. Remaining x_{ij} are set to zero. The corresponding Lagrangian master problem is:

$$
\begin{aligned}
[MP]: \qquad z_{MP} = \max \quad & \sum_{j=1}^{n} \theta_j - \sum_{i=1}^{m} \lambda_i b_i \\
\text{s.t} \quad & \theta_j \leq \sum_{i=1}^{m} (c_{ij} + \lambda_i w_{ij}) x_{ij}^h \qquad \forall h \in H_j, j \in J \\
& \lambda_i \geq 0, i \in I
\end{aligned}
$$

where $x_{ij}^h, \forall h \in H$ are feasible solutions to $[SP_j]$

SP and MP are solved iteratively using the cutting plane method to obtain a Lagrangian bound, LB, on the optimal value of the original problem. SP is solved with the set of Lagrangian multipliers λ as parameters, obtained from MP. The value of the solution of SP, $(\sum_{j=1}^{n} v_j - \sum_{i=1}^{m} \lambda_i b_i)$, gives a lower bound, lb, on the Lagrangian bound, LR^*. Cuts based on SP solution are added to MP, to obtain improved multipliers. The solution of MP provides an upper bound, ub, on the value of LR^*. The solution of SP and MP continue until lb and ub converge to a desired tolerance. The value LR^* at which both bounds converge is the Lagrangian lower bound on z^*, the optimal solution for the GAP:

$$LR^* \leq z^*$$

We note that the Lagrangian multipliers can also be updated using the well-known subgradient optimization method (Fisher 1985).

3.2 Relaxing the Assignment Constraints

The second relaxation can be obtained by relaxing the assignment constraints with Lagrangian multipliers $u_j, j \in J$. We get the following Lagrangian subproblem:

$$
\begin{aligned}
[SP2]: \qquad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+u_j)x_{ij} - \sum_{j=1}^{n} u_j \\
\text{s.t} \quad & \sum_{j=1}^{n} w_{ij}x_{ij} \le b_i \qquad \forall i \in I \\
& x_{ij} \in \{0,1\}
\end{aligned}
$$

which reduces to m binary knapsack problems:

$$
\begin{aligned}
[SP_i]: v_i = \qquad \min \quad & \sum_{j=1}^{n}(c_{ij}+u_j)x_{ij} \\
\text{s.t} \quad & \sum_{j=1}^{n} w_{ij}x_{ij} \le b_i \\
& x_{ij} \in \{0,1\}
\end{aligned}
$$

The Lagrangian bound in this case is $\sum_{i=1}^{m} v_i - \sum_{j=1}^{n} u_j$. The best bound is the solution of the Lagrangian master problem:

$$
\begin{aligned}
[MP2]: \qquad \max \quad & \sum_{i=1}^{m}\theta_i - \sum_{j=1}^{n} u_j \\
\text{s.t} \quad & \theta_i - \sum_{j=1}^{n} x_{ij}^h u_j \le \sum_{j=1}^{n} c_{ij}x_{ij}^h \qquad \forall h \in H_i, i \in I \\
& u_j \ge 0 \, \forall j \in J
\end{aligned}
$$

where H_i are the index sets of feasible solutions to $[SP_i]$.

Chapter 4

Improved Lagrangian Approach

In this chapter, we present a general framework for the improved Lagrangian approach. We then apply it to the GAP and present the two different methodologies we devise.

4.1 General Framework

Consider the general MIP from Chapter 2:

$$
\begin{aligned}
[P]: \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax \geq b \\
& Bx \geq d \\
& x \geq 0 \\
& x_j \, integer \;\; \forall j \in J
\end{aligned}
$$

and the corresponding Lagrangian subproblem (SP):

$$
\begin{aligned}
[SP(\lambda)]: \quad \min \quad & (c^T - \lambda^T A)x \\
\text{s.t.} \quad & Bx \geq d \\
& x \geq 0,\ x_j \, integer \ \forall j \in J
\end{aligned}
$$

The main idea behind the improved approach is to use information based on subproblem solutions to generate valid cuts that are added back to the subproblem. This is based on checking their feasibility with respect to the relaxed constraints.

One potential drawback of this approach is that adding cuts to SP may make it more difficult to solve to optimality. To counter this drawback, we propose few modifications. Instead of solving SP to optimality, as in the standard Lagrangian approach, a time limit on SP solution is imposed. To maintain the validity of the Lagrangian lower bound, a lower bound on the solution of SP is used instead of the optimal value. For example in Gurobi, the MIPGap is used to obtain a lower bound. If SP value obtained with a solution time limit is 100, with MIPGap of 5% from Gurobi, then $100(1 - 0.05) = 95$ is used as a lower bound. Hence, these adjusted SP values at each cutting plane iteration can be used to update lb. This also implies that lb and ub may never converge as SP is not being solved to optimality. Therefore, the procedure is terminated when the Lagrangian multipliers become stagnant over a certain number of cutting plane iterations, indicating no change in the SP solution cuts being added to MP. The value of lb at termination is a valid lower bound. We only employ this stopping criterion and LB estimate when a time limit is imposed on the solution of SP, and hence may not be solved to optimality.

4.2 Application to the GAP

We apply the general approach discussed above to develop two methodologies for the GAP. In our case we work with the first form of LR of the GAP, presented in Section 3.2, which is obtained by dualizing the capacity constraints (3.2) with the Lagrangian multipliers $\lambda_i, \geq 0, i \in I$. We get the following SP:

$$
\begin{aligned}
[SP]: \qquad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+\lambda_i w_{ij})x_{ij} & \\
\text{s.t} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \forall j \in J \\
& x_{ij} \in \{0,1\} &
\end{aligned}
$$

For our methodologies, we do not decompose SP further into j subproblems as shown in Section 3.2 as the addition of valid cuts does not allow decomposability. The corresponding Lagrangian master problem is:

$$
\begin{aligned}
[MP]: \qquad z_{MP} = \max \quad & \theta - \sum_{i=1}^{m}\lambda_i b_i & \\
\text{s.t} \quad & \theta \leq \sum_{i=1}^{m}(c_{ij}+\lambda_i w_{ij})x_{ij}^{h} & \forall h \in H \\
& \lambda_i \geq 0, i \in I &
\end{aligned}
$$

where H is the index set of feasible solutions to SP.

Solutions from the subproblem are checked for feasibility with respect to the capacity constraints. If the capacity of any knapsack is violated, that SP solution is infeasible for the GAP. To eliminate this infeasible solution and to push future SP solutions towards feasibility for GAP, several schemes of generating valid cuts can

be used. One such scheme of generating cuts is based on minimal cover inequalities that we have employed in our methodology. An illustration and general procedure for generating the minimal cover inequalities is presented in the following section.

4.2.1 Minimal Cover Inequalities

The cuts we add to SP are minimal cover inequalities. Following is an illustration of a minimal cover inequality generated for a particular knapsack. Let us assume that in SP solution, the following items have been allocated to a knapsack with capacity $b_1 = 50$:

Item	**Weight**
1	10
2	15
3	20
4	30
5	40
6	50

It is evident that since the total weight is 165, this solution is not feasible for the GAP because the capacity is violated. If we were to maximize the number of items in the knapsack then at most three items can fit in the knapsack. Thus the minimal cover inequality for this knapsack is:

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} \leq 3$$

This type of inequality ensures that from a set of items, S_i, assigned to knapsack i, the total number of items in the knapsack cannot exceed the maximum number that it can contain. The general idea to generate minimal cover inequalities is to

sort the items in order of increasing weight and count how many would fit in the knapsack. The following algorithm summarizes the procedure.

Algorithm 1 Minimal Cover Inequalities Generation Procedure

Notation:
S_i: set of items j assigned to knapsack i in SP solution, sorted in increasing order of weight w_{ij}
b_i: capacity of knapsack i

initialize $count = 0$
for item $j \in S_i$ **do**
  while $w_{ij} < b_i$ **do**
    $count = count + 1$
    $b_i = b_i - w_{ij}$
  end while
  Add inequality $(\sum_{j \in S_i} x_{ij} \leq count)$ to SP
end for

4.3 Proposed Lagrangian approaches

Based on the improved Lagrangian approach discussed in section 4.1, we develop two approaches for the GAP: Modified Lagrangian and Feasible Lagrangian approach. Modified Lagrangian approach has two variants: with accumulation of minimal cover inequalities and with restart.

4.3.1 Modified Lagrangian approach

This method begins with the standard Lagrangian procedure. Once the upper bound obtained from MP, ub, is within 10% of the lower bound obtained from SP, lb, the SP solution is checked for feasibility to GAP. In the case of infeasibility, minimal cover inequalities are generated and added to SP. SP is then solved again with the newly added cuts. To solve SP quickly, a time limit, $tLim$, is imposed. The generation of cover inequalities and the solution of subproblems continues until a feasible solution is found or a certain number of these iterations, k, is achieved. As SP is not solved to optimality, the lower bound is adjusted based on the optimality gap.

Once the iterative solution of SP concludes, cut based on the last SP solution is added to MP. This completes one iteration of the cutting plane method. The complete procedure is stopped when the Lagrangian multipliers, λ, stay stagnant over a certain number of cutting plane iterations.

We test two variants of this approach:

1. Modified Lagrangian approach with cut accumulation: Minimal cover inequalities added in SP are carried forward from one cutting plane iteration to another, i.e., cuts keep accumulating in SP throughout the algorithm.

2. Modified Lagrangian approach with restart: Minimal cover inequalities are discarded when moving from one cutting plane iteration to the next, i.e., when the next SP is solved with updated λ values, it does not contain any inequalities from the previous iterations.

Modified Lagrangian approach with cut accumulation

The algorithm steps of each approach are presented next.

1. **Step 1:** Start with the standard Lagrangian approach, iterating between MP and SP until the ub is within 10% of lb.
2. **Step 2:** Solve SP and check if its solution is feasible to the GAP.
3. **Step 3:** If SP solution is not feasible then generate minimal cover inequalities.
4. **Step 4:** Solve SP with time limit $tLim$.
5. **Step 5:** Use the objective value of SP, adjusted with the optimality gap, to update the lower bound, lb.
6. **Step 6:** Keep solving SP and adding minimal cover inequalities until a feasible solution is reached or a given number of iterations, k, is performed.
7. **Step 7:** Add a cut to MP based on the last solution obtained from SP, and solve MP to get ub and new Lagrangian multipliers λ.
8. **Step 8:** Terminate if $(ub - lb)/ub < \epsilon$ or when the Lagrangian multiplier values become stagnant over a certain number of iterations (5 in our case).
9. **Step 9:** Otherwise, use the new values of λ and repeat **Step 2-8**, using SP with all the minimal cover inequalities added in the previous iterations.

Modified Lagrangian approach with restart

In this version all the algorithm steps from **(1)-(7)** are the same except that at **Step 8**, when SP is solved again with new Lagrangian multiplier values λ, all the minimal cover inequalities added in SP are discarded, i.e, in the new iteration, SP does not contain any minimal cover inequalities from the previous iterations.

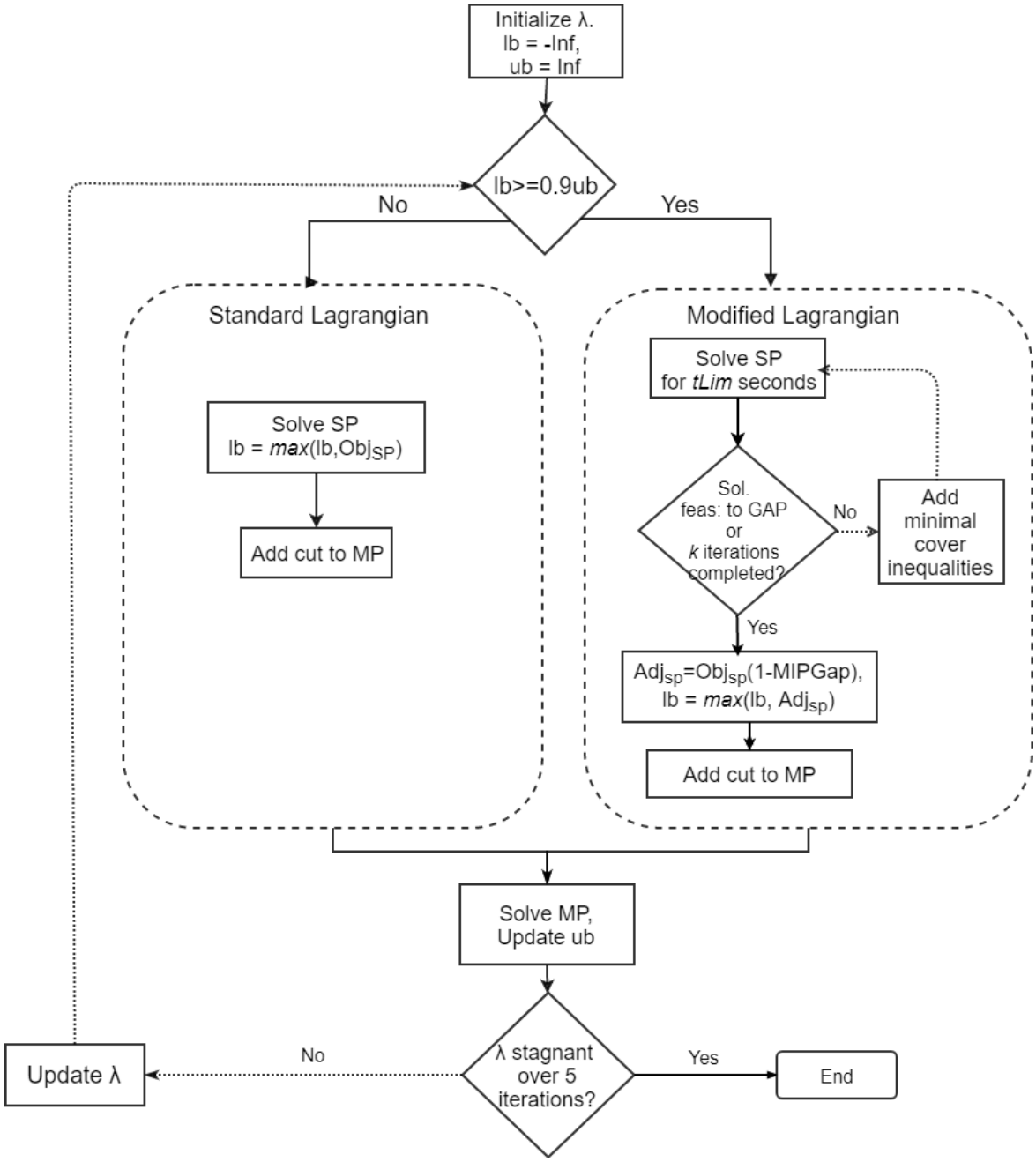

Figure 4.1: Modified Lagrangian approach

4.3.2 Feasible Lagrangian approach

In the Feasible Lagrangian approach, the minimal cover inequalities are added to SP until a feasible solution to GAP is obtained. Then the SP with all the added minimal cover inequalities is used in the cutting plane method to generate improved bounds. This approach differs from Modified Lagrangian approach, as instead of strengthening the subproblem with valid cuts at every Lagrangian iteration, a strengthened SP is obtained first and then the regular Lagrangian procedure is performed.

The initial Lagrangian multipliers are set to the optimal dual variables of the relaxed constraints in the linear programming (LP) relaxation of GAP. We exploit the fact that relaxing the capacity constraints leads to a subproblem with integrality property. Therefore, first the LP relaxation of GAP is solved to initialize the Lagrangian multipliers. Also, in the cutting plane method, SP is solved to optimality without any time limit.

The steps of the algorithm are:

Step 1: Solve LP relaxation of GAP:

$$[GAP] \qquad \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{4.1}$$

$$\text{s.t} \quad \sum_{j=1}^{n} w_{ij}x_{ij} \leq b_i \qquad \forall i \in I \tag{4.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in J \tag{4.3}$$

$$0 \leq x_{ij} \leq 1 \tag{4.4}$$

Step 2: Set the Lagrangian multipliers, λ_i, to the optimal dual values, μ_i, of the capacity constraints (4.2).

Step 3: Solve SP and check the solution for violation of capacity constraints.

Step 4: If not feasible, add minimal cover inequalities until a feasible solution is found.

Step 5: Add a cut to MP based on the feasible SP solution.

Step 6: Start regular Lagrangian procedure, iterating between SP and MP.

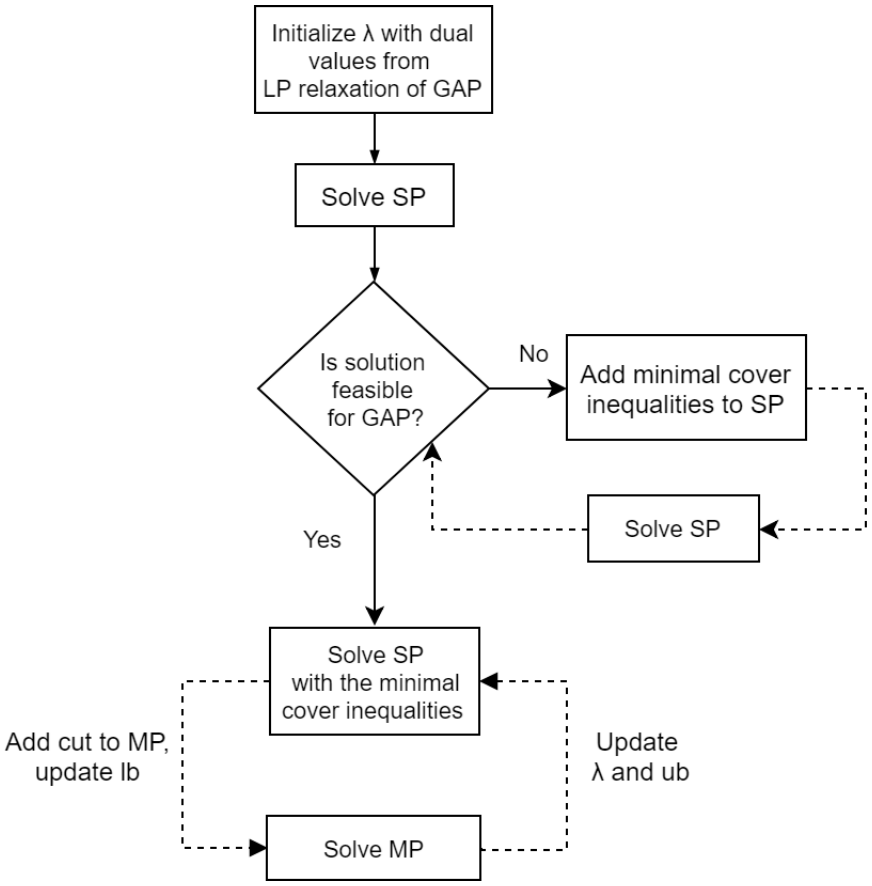

Figure 4.2: Feasible Lagrangian approach

Chapter 5

Computational Results

In this chapter, the proposed methodologies are tested on benchmark instances. First, we describe the test instances used in the experiments. Second, we present and analyze results. Finally, we compare the results and derive insights.

5.1 Data

The data used for testing is generated based on a scheme presented by Martello and Toth (1981). The scheme defines four different methods of generating data, type A to D, varying in terms of difficulty. In addition, we also use the instances, type E, presented by Yagiura et al.(2006). The generation scheme is as follows.

- **Type A:** w_{ij} are integers from $U(5, 25)$, c_{ij} are integers from $U(10, 50)$, and $b_i = 0.6(n/m)15 + 0.4 \max_{i \in I} \sum_{j \in J,\, I_j = i} w_{ij}$, where $I_j = \min \{i | c_{ij} \leq c_{kj} \, \forall \, k \in I\}$.
- **Type B:** w_{ij} and c_{ij} are the same as Type A and b_i is set to 70% of the value given in Type A.

- **Type C:** w_{ij} and c_{ij} are the same as Type A and $b_i = 0.8 \sum_{j \in J} w_{ij}/m$.
- **Type D:** w_{ij} are integers from $U(1, 100)$, $c_{ij} = 111 - w_{ij} + e_1$, where e_1 are integers from $U(-10, 10)$, and $b_i = 0.8 \sum_{j \in J} w_{ij}/m$.
- **Type E:** $w_{ij} = 1 - 10\ln e_2$, where e_2 are numbers from $U(0, 1]$, $c_{ij} = 1000/w_{ij} - 10e_3$, where e_3 are numbers from $U(0, 1)$, and $b_i = 0.8 \sum_{j \in J} w_{ij}/m$

These types are ordered based on the level of difficulty. In type D and E, c_{ij} and w_{ij} are inversely correlated, which makes them harder to solve compared to the other three types. To better asses the effectiveness of our methodologies, we test them on the more difficult types C, D, and E.

Our data is divided into small and medium size based on the number of knapsacks, m, and the number of items, n. We use the data available online, that was generated based on the scheme mentioned above. Small instances are from Cattrysse et al.(1994) and belong to type C. For these instances n ranges from 15 to 60 and m ranges from 5 to 10. There are five instances for each problem size (m, n). In the results tables for the small instances, we report the average results of five instances for each problem size. Medium instances of types C and D are from Chu and Beasley (1997) and type E instances are from Yagiura et al.(2006). For these instances, n ranges from 100 to 200 and m ranges from 5 to 20.

5.2 Numerical Results

This section reports the results from testing our proposed methodologies presented in Chapter 4. The solution approaches are coded in Python and solved using Gurobi 8.0 on a computer with 2.6 GHz Inter Core i7 and 8GB of memory. As adding minimal

cover inequalities to SP makes it more difficult to solve to optimality quickly, we test our methods with different time limits on SP solution, $tLim$. We use $tLim \in \{1, 5\}$ seconds for small instances and $tLim \in \{5, 10\}$ seconds for medium instances. In the experiments we also vary the number of minimal cover inequalities generation iterations, k, to capture the effect of increasing the number of these inequalities on the lower bound and on feasible solutions. We set $k \in \{5, 10, 15\}$ for small instances and $k \in \{20, 30, 40\}$ for medium instances.

The following notations are used to report the results in the tables:

- **Iter**: Total number of cutting plane iterations performed.
- **LB**: Lower bound obtained by the proposed approach.
- **Lag**: Lower bound obtained from the standard Lagrangian method. It is used as benchmark for LB.
- **Sol**: Best feasible solution obtained by the proposed approach.
- **BSol**: Best known solutions. For small instances, the solution is obtained using Gurobi. For medium instances, best known solution from Yagiura et al.(2006) is used.
- **GapR**: Refers to the reduction in the optimality gap achieved by LB. It is calculated by $\frac{(Gap_{Lag} - Gap_{LB})}{Gap_{Lag}} \times 100$, where $Gap_{lag} = BSol - Lag$ and $Gap_{LB} = BSol - LB$.
- **GapS**: Refers to the percentage gap between Sol and BSol calculated by $\frac{Sol - BSol}{BSol} \times 100$.
- **GapO**: Refers to the percentage gap between Sol and LB calculated by $\frac{Sol - LB}{LB} \times 100$.

- **Clckt**: Total clock time used. (Since, the MP solution time is negligible, requiring just few milliseconds, Clckt is mostly total clock time used by the SP).
- **CPUt**: Total CPU time used.
- **Cuts**: Total minimal cover inequalities accumulated in the SP for the cut accumulation method.
- **Feas**: Total number of small instances for which a feasible solution was found by the method.
- **Opt**: Total number of small instances for which optimal solutions were found by the proposed approach.

5.2.1 Results of Modified Lagrangian approach with cut accumulation

Table 5.1 reports the average results obtained from solving five instances for each problem size (m, n) using the Modified Lagrangian approach with cut accumulation. It can be observed that there is a significant improvement in the lower bounds, measured by $GapR$, achieving on average more than 60% improvement for all the combinations of k and $tLim$ values. The highest average improvement in lower bounds of 76.4% is attained for $k = 15$ and $tLim = 5$. The method also yields quality feasible solutions, with on average only 0.3% $GapS$, i.e., the gap between Sol and $BSol$. The quality of feasible solutions does not vary much when k and $tLim$ values are increased. However, the setting with $k = 15$ and $tLim = 5$ is able to find optimal solutions for 2 out of 5 instances on average, for each problem size (m,n), that is the highest among all the settings. For the largest small instance, ($m = 10$,$n = 60$), the lower bound improvement, $GapR$, is decreased from 68.7% to 19.7% when the value of k is increased from 5 to 15. The decline in $GapR$ is observed because SP becomes more difficult to solve near optimality in the given solution time limit when the number of cover inequalities increases. Similarly, for any value of k, when $tLim$ is increased from 1 to 5 seconds, an average improvement of around 10% is observed in $GapR$.

Similar observations can be made in the case of medium instances. In Table 5.2 the highest average $GapR$ of 5.2% in lower bounds is achieved by ($k = 20$, $tLim = 5$) for type C instances. As type D and E instances are more difficult to solve compared to type C instances, adding inequalities in SP makes them difficult to solve closer to optimality in limited solution time, leading to worse lower bounds and negative $GapR$. In terms of finding the feasible solutions, for all type C instances and for 4

	k = 5 cuts, tLim = 1									k = 5 cuts, tLim = 5								
n-m	Iter	GapR	GapS	GapO	Clckt	CPUt	Cuts	Feas	Opt	Iter	GapR	GapS	GapO	Clckt	CPUt	Cuts	Feas	Opt
5-15	13	67.3%	0.0%	0.8%	1	1	75	4	4	13	67.3%	0.0%	0.8%	1	2	75	4	4
5-20	16	64.4%	0.8%	1.6%	1	3	92	5	3	16	64.4%	0.8%	1.6%	1	3	92	5	3
5-25	20	68.1%	0.2%	0.6%	6	16	186	5	2	20	68.1%	0.2%	0.6%	6	17	186	5	2
5-30	25	51.1%	0.6%	1.5%	7	18	208	5	1	25	51.1%	0.6%	1.5%	7	18	208	5	1
8-24	19	89.2%	0.3%	0.4%	5	11	171	5	2	19	89.2%	0.3%	0.4%	5	11	171	5	2
8-32	25	78.3%	0.2%	0.4%	16	47	316	5	3	25	78.3%	0.2%	0.4%	17	47	316	5	3
8-40	41	66.9%	0.3%	0.6%	82	274	581	5	1	39	77.6%	0.2%	0.5%	108	383	567	5	2
8-48	57	43.1%	0.2%	0.6%	196	648	928	3	0	55	62.5%	0.3%	0.6%	313	1136	853	3	1
10-30	35	75.6%	0.3%	0.9%	53	185	410	4	0	33	87.8%	0.3%	0.6%	72	259	394	3	0
10-40	43	78.3%	0.1%	0.3%	81	272	594	5	3	38	80.5%	0.0%	0.3%	117	419	580	5	4
10-50	47	86.0%	0.3%	0.4%	107	355	694	5	1	45	107.2%	0.3%	0.3%	173	617	694	5	1
10-60	68	36.6%	0.1%	0.5%	285	888	1356	5	1	70	68.7%	0.2%	0.4%	811	2960	1299	5	0
Min		36.6%	0.0%	0.3%	1	1		3	0		51.1%	0.0%	0.3%	1	2		3	0
Max		89.2%	0.8%	1.6%	285	888		5	4		107.2%	0.8%	1.6%	811	2960		5	4
Avg		67.1%	0.3%	0.7%	70	226		4.7	1.8		75.2%	0.3%	0.7%	136	489		4.6	1.9
	k = 10 cuts, tLim = 1									k = 10 cuts, tLim = 5								
5-15	12	71.4%	0.1%	1.1%	1	3	94	5	4	12	71.4%	0.1%	1.1%	1	3	94	5	4
5-20	15	81.4%	1.3%	1.6%	2	5	127	5	2	15	81.4%	1.3%	1.6%	2	5	127	5	2
5-25	18	68.1%	0.2%	0.6%	14	42	274	5	2	18	68.1%	0.2%	0.6%	14	42	274	5	2
5-30	22	61.8%	0.3%	1.0%	17	51	300	4	1	22	61.8%	0.3%	1.0%	17	51	300	4	1
8-24	16	85.2%	0.1%	0.3%	8	22	219	4	3	16	85.2%	0.1%	0.3%	8	21	219	4	3
8-32	23	78.3%	0.3%	0.5%	30	94	411	5	4	23	78.3%	0.3%	0.5%	31	94	411	5	4
8-40	42	55.8%	0.2%	0.7%	184	610	909	5	1	36	83.5%	0.2%	0.4%	291	1068	810	5	1
8-48	53	30.7%	0.4%	1.0%	350	1111	1456	3	0	54	60.3%	0.5%	0.9%	1097	4065	1375	4	0
10-30	31	77.8%	0.3%	0.7%	72	252	477	5	0	28	89.7%	0.3%	0.5%	144	538	470	5	0
10-40	38	62.3%	0.1%	0.5%	159	527	901	5	3	37	72.5%	0.1%	0.4%	403	1494	860	5	3
10-50	46	77.8%	0.3%	0.6%	214	688	1075	5	1	43	99.2%	0.3%	0.4%	536	1989	986	5	1
10-60	61	13.0%	0.3%	0.8%	485	1431	2142	5	1	67	40.0%	0.2%	0.5%	2032	7347	2299	5	0
Min		13.0%	0.1%	0.3%	1	3		3	0		40.0%	0.1%	0.3%	1	3		4	0
Max		85.2%	1.3%	1.6%	485	1431		5	4		99.2%	1.3%	1.6%	2032	7347		5	4
Avg		63.6%	0.3%	0.8%	128	403		4.7	1.8		74.3%	0.3%	0.7%	381	1393		4.8	1.8
	k = 15 cuts, tLim = 1									k = 15 cuts, tLim = 5								
5-15	11	74.9%	0.0%	0.6%	2	4	107	4	4	11	74.9%	0.0%	0.6%	2	4	107	4	4
5-20	14	93.9%	1.0%	1.3%	3	8	147	5	2	14	93.9%	1.0%	1.3%	3	8	147	5	2
5-25	19	86.0%	0.2%	0.4%	30	94	382	5	2	19	86.0%	0.2%	0.4%	30	94	382	5	2
5-30	22	59.3%	0.3%	1.1%	34	108	399	5	2	21	62.1%	0.3%	1.0%	34	104	400	5	2
8-24	16	99.8%	0.3%	0.3%	9	24	235	5	2	16	99.8%	0.3%	0.3%	9	25	235	5	2
8-32	25	89.4%	0.1%	0.3%	71	231	553	5	3	24	89.4%	0.2%	0.3%	81	275	555	5	3
8-40	36	50.8%	0.1%	0.6%	236	781	1117	5	3	35	75.6%	0.2%	0.4%	520	1912	1036	5	1
8-48	55	11.7%	0.4%	1.2%	618	1878	2098	4	0	55	53.3%	0.3%	0.7%	2089	7692	2012	5	2
10-30	29	76.1%	0.3%	0.8%	107	372	615	5	1	31	92.2%	0.3%	0.5%	331	1262	634	5	1
10-40	35	64.7%	0.1%	0.5%	220	738	1072	5	3	34	78.3%	0.1%	0.3%	540	2015	982	5	3
10-50	43	65.5%	0.5%	0.9%	293	931	1286	5	0	42	91.2%	0.4%	0.5%	848	3143	1236	5	1
10-60	53	16.4%	0.2%	0.7%	671	1935	2869	4	0	70	19.7%	0.1%	0.6%	3285	11643	3121	5	1
Min		11.7%	0.0%	0.3%	2	4		4	0		19.7%	0.0%	0.3%	2	4		4	1
Max		99.8%	1.0%	1.3%	671	1935		5	4		99.8%	1.0%	1.3%	3285	11643		5	4
Avg		65.7%	0.3%	0.7%	191	592		4.8	1.8		76.4%	0.3%	0.6%	648	2348		4.9	2.0

Table 5.1: Results of the cut accumulation approach on small instances

out of 6 type D and E instances, feasible solutions are found. The solutions can be considered high quality as the average *GapS* is around 1% for type C and E instances,

and around 4% for type D instances on average. However, the computational times for this method are very high with an average of around 20,000 seconds of CPU time used to solve the instances.

Problem	Iter	LB	Lag	GapR	Sol	BSol	GapS	GapO	Clockt	CPUt	Cuts
c5-100	37	1930	1924	85.7%	1945	1931	0.7%	0.8%	331	1130	1442
c10-100	77	1386	1388	-14.3%	1409	1402	0.5%	1.7%	3360	13267	4084
c20-100	108	1220	1219	4.2%	1252	1243	0.7%	2.6%	4848	17990	5373
c5-200	50	3453	3451	40.0%	3467	3456	0.3%	0.4%	458	1497	1799
c10-200	119	2794	2796	-20.0%	2823	2806	0.6%	1.0%	8920	27889	9550
c20-200	204	2368	2377	-64.3%	2422	2391	1.3%	2.3%	17122	46986	28296
Min				-64.3%			0.3%	0.4%	331	1130	
Max				85.7%			1.3%	2.6%	17122	46986	
Avg				5.2%			0.7%	1.5%	5840	18126	
d5-100	57	6346	6346	0.0%	6424	6353	1.1%	1.2%	1128	4611	2371
d10-100	98	6324	6324	0.0%	7130	6355	12.2%	12.7%	8487	31934	9941
d20-100	170	6115	6143	-41.2%	-	6211	-	-	14153	44505	22275
d5-200	67	12736	12737	-14.3%	12814	12744	0.5%	0.6%	773	2153	3079
d10-200	142	12418	12419	-5.3%	12557	12438	1.0%	1.1%	13092	44418	13939
d20-200	287	12214	12218	-7.8%	-	12269	-	-	29164	66597	53319
Min				-41.2%			0.5%	0.6%	773	2153	
Max				0.0%			12.2%	12.7%	29164	66597	
Avg				-11.4%			3.7%	3.9%	11133	32370	
e5-100	55	12653	12642	28.2%	12831	12681	1.2%	1.4%	624	2418	1855
e10-100	130	11541	11544	-9.1%	11794	11577	1.9%	2.2%	7999	28819	7807
e20-100	215	8305	8360	-66.3%	-	8443	-	-	14011	41564	23558
e5-200	65	24922	24922	0.0%	25002	24930	0.3%	0.3%	225	406	2110
e10-200	146	23291	23294	-23.1%	23559	23307	1.1%	1.2%	11154	34974	12626
e20-200	324	22319	22356	-160.9%	-	22379	-	-	29253	60297	56654
Min				-160.9%			0.3%	0.3%	225	406	
Max				28.2%			1.9%	2.2%	29253	60297	
Avg				-38.5%			1.1%	1.3%	10544	28080	

Table 5.2: Results of cut accumulation approach on medium instances, $k = 20$ and $tLim = 5$ seconds

Table 5.3 shows the results for increasing the time limit on the solution of SP, $tLim$, from 5 to 10 seconds while keeping the number of cover inequalities generation iterations $k = 20$, that is same as in the previous Table 5.2. We find that allowing more solution time for SP leads to an average improvement of around 20% in $GapR$ for type C instances. For types D and E, the average $GapR$ is still negative but it has improved. For type E instances the average $GapR$ reduced from -38.5% to -26.6% due to increase in $tLim$. In terms of finding feasible solutions, 2 out of 5 instances of type C for which a feasible solution was found with $tLim = 5$, no

feasible solution was found by increasing $tLim$ to 10. For one instance in type D, a feasible solution was found with $tLim = 10$, for which no feasible solution was found with $tLim = 5$. This shows that by increasing the SP solution time, the likelihood of finding a feasible solution is not necessarily increased. In terms of quality the feasible solutions, increasing $tLim$ from 5 to 10 seconds leads to noticeable improvement in type D instances with average $GapS$ reducing from 3.7% to 1.5%. For type C and E, the average $GapS$ remained almost the same.

Problem	**Iter**	**LB**	**Lag**	**GapR**	**Sol**	**BSol**	**GapS**	**GapO**	**Clockt**	**CPUt**	**Cuts**
c5-100	37	1930	1924	85.7%	1945	1931	0.7%	0.8%	331	1127	1442
c10-100	69	1395	1388	50.0%	1409	1402	0.5%	1.0%	5919	25747	4100
c20-100	103	1226	1219	29.2%	-	1243	-	-	8754	39058	5203
c5-200	50	3453	3451	40.0%	3467	3456	0.3%	0.4%	462	1518	1799
c10-200	129	2795	2796	-10.0%	2834	2806	1.0%	1.4%	18576	62567	10476
c20-200	215	2369	2377	-57.1%	-	2391	-	-	35666	107339	29377
Min				-57.1%			0.3%	0.4%	331	1127	
Max				85.7%			1.0%	1.4%	35666	107339	
Average				23.0%			0.6%	0.9%	11618	39559	
d5-100	52	6346	6346	0.0%	6424	6353	1.1%	1.2%	696	3460	2104
d10-100	121	6326	6324	6.5%	6401	6355	0.7%	1.2%	20977	83312	11539
d20-100	167	6124	6143	-27.9%	6464	6211	4.1%	5.6%	26852	96991	23274
d5-200	61	12737	12737	0.0%	12811	12744	0.5%	0.6%	778	1746	2875
d10-200	129	12418	12419	-5.3%	12596	12438	1.3%	1.4%	20008	78838	12294
d20-200	278	12209	12218	-17.6%	-	12269	-	-	52946	142571	52413
Min				-27.9%			0.5%	0.6%	696	1746	
Max				6.5%			4.1%	5.6%	52946	142571	
Average				-7.4%			1.5%	2.0%	20376	67820	
e5-100	58	12654	12642	30.8%	12831	12681	1.2%	1.4%	777	3159	2101
e10-100	142	11546	11544	6.1%	11752	11577	1.5%	1.8%	19481	73646	9077
e20-100	207	8301	8360	-71.1%	-	8443	-	-	25749	85699	22191
e5-200	65	24922	24922	0.0%	25002	24930	0.3%	0.3%	264	413	2110
e10-200	146	23293	23294	-7.7%	-	23307	-	-	20998	75985	11604
e20-200	309	22329	22356	-117.4%	-	22379	-	-	51539	131491	50764
Min				-117.4%			0.3%	0.3%	264	413	
Max				30.8%			1.5%	1.8%	51539	131491	
Average				-26.6%			1.0%	1.2%	19801	61732	

Table 5.3: Results of the cut accumulation approach on medium instances, $k = 20$ and $tLim = 10$ seconds

Table 5.4 shows the results from increasing the value of k from 20 to 40 while the solution time limit on SP is the same, 5 seconds, so as to capture the effect of increasing the number of minimal cover inequalities in SP. The new setting leads to worse lower bounds compared to previous settings, with average $GapR$ reducing

from 5.2% ($k = 20$, $tLim = 5$) to -16%($k = 40$, $tLim = 5$) for type C instances. However, there is an improvement in the feasible solutions in all types of instances, with the average $GapS$ for type D being reduced from 3.7% to 1.9%.

Problem	Iter	LB	Lag	GapR	Sol	BSol	GapS	GapO	Clockt	CPUt	Cuts
c5-100	46	1928	1924	57.1%	1931	1931	0.0%	0.2%	2137	7819	2811
c10-100	73	1386	1388	-14.3%	1410	1402	0.6%	1.7%	8214	27914	8809
c20-100	106	1221	1219	8.3%	1251	1243	0.6%	2.5%	8113	28182	7903
c5-200	57	3453	3451	40.0%	3463	3456	0.2%	0.3%	3688	12627	3705
c10-200	99	2793	2796	-30.0%	2824	2806	0.6%	1.1%	14611	42289	16027
c20-200	162	2355	2377	-157.1%	-	2391	-	-	24518	61910	43327
Min				-157.1%			0.0%	0.2%	2137	7819	
Max				57.1%			0.6%	2.5%	24518	61910	
Avg				-16.0%			0.4%	1.1%	10214	30124	
d5-100	52	6346	6346	0.0%	6386	6353	0.5%	0.6%	4044	17737	4054
d10-100	105	6315	6324	-29.0%	6440	6355	1.3%	2.0%	19026	61853	18458
d20-100	158	6108	6143	-51.5%	6593	6211	6.2%	7.9%	25209	70224	42683
d5-200	64	12736	12737	-14.3%	12779	12744	0.3%	0.3%	4159	12019	5310
d10-200	142	12415	12419	-21.1%	12564	12438	1.0%	1.2%	29309	78672	29699
d20-200	221	12205	12218	-25.5%	-	12269	-	-	42199	73879	77635
Min				-51.5%			0.3%	0.3%	4044	12019	
Max				0.0%			6.2%	7.9%	42199	78672	
Avg				-23.6%			1.9%	2.4%	20658	52397	
e5-100	54	12654	12642	30.8%	12831	12681	1.2%	1.4%	3060	12175	3448
e10-100	124	11535	11544	-27.3%	11797	11577	1.9%	2.3%	16668	52600	15434
e20-100	169	8243	8360	-141.0%	-	8443	-	-	17157	49372	31887
e5-200	70	24921	24922	-12.5%	25045	24930	0.5%	0.5%	3845	10875	5411
e10-200	134	23291	23294	-23.1%	-	23307	-	-	20763	58797	21091
Min				-141.0%			0.5%	0.5%	3060	10875	
Max				30.8%			1.9%	2.4%	20763	58797	
Avg				-32.8%			1.4%	1.6%	13692	39369	

Table 5.4: Results of the cut accumulation approach on medium instances, $k = 40$ and $tLim = 5$ seconds

Hence, drawing from the results for the Modified Lagrangian approach with accumulation of cover inequalities, we observe that, for the small and medium instances and for different combinations of k and $tLim$ values, increasing the number of minimal cover inequalities does certainly lead to improvement in lower bounds and quality feasible solutions, given that the SP is solved closer to optimality.

5.2.2 Results of Modified Lagrangian approach with restart

In this section, we present the results for a slightly different version of the previous methodology. In this version the minimal cover inequalities that are added to SP in one iteration of the cutting plane method, are not carried forward to the following iterations. Because the cover inequalities are not accumulated, the number of these inequalities in SP can grow to a maximum of $m \times k$. This is a much smaller number of cuts compared to the previous method where the number of inequalities in SP, in the last cutting plane iteration, can grow to a maximum of $m \times k \times iter$, where $iter$ is the total number of cutting plane iterations performed. Hence, we hope to obtain improved lower bounds and good feasible solutions from this method with less SP solution time.

Table 5.5 presents the results for the small instances. In terms of $GapR$, it can be observed that this method achieves as much as 57.6% improvement on average in the lower bounds for the setting ($k = 15$, $tLim = 1$), but the improvement is around 20% less when compared to the cut accumulation method ($k = 15$, $tLim = 5$). When the value of k is increased from 5 to 15, the average $GapR$ improves by around 17.6%. In this method, finding a feasible solution is less likely compared to the cut accumulation method. For $k = 5$ no feasible solutions are found for any instances for 6 out 12 problem sizes. As k increases from 5 to 15, the number of times an optimal solution is found on average increases from 0.42 to 1.17. However, it is still lower compared to the cut accumulation method which finds an optimal solution for 2 out of 5 times on average, in the best case ($k = 15$, $tLim = 5$). In terms of the quality of solutions, the best average $GapS$ achieved by the current method is slightly worse, 0.4% compared to 0.3% for the cut accumulation method. Although this approach does not perform as well as the accumulation method on all the metrics discussed

	k = 5 cuts, tLim = 1									k = 5 cuts, tLim = 5								
n-m	Iter	GapR	GapS	GapO	Clckt	CPUt	Cuts	Feas	Opt	Iter	GapR	GapS	GapO	Clckt	CPUt	Cuts	Feas	Opt
5-15	20.6	39.6%	0.0%	1.3%	1	1	29	2	2	20.6	39.6%	0.0%	1.3%	1	1	29	2	2
5-20	22.4	41.7%	1.6%	2.9%	2	1	36.6	3	2	22.4	41.7%	1.6%	2.9%	2	1	36.6	3	2
5-25	26.6	31.2%	0.8%	1.6%	2	2	38.8	3	1	26.6	31.2%	0.8%	1.6%	2	2	38.8	3	1
5-30	33.6	26.4%	2.8%	4.3%	3	3	43.6	3	0	33.6	26.4%	2.8%	4.3%	3	3	43.6	3	0
8-24	34.2	52.2%	1.5%	2.1%	4	3	43.2	1	0	34.2	52.2%	1.5%	2.1%	4	3	43.2	1	0
8-32	39	48.1%	-	-	5	5	57.8	0	0	39	48.1%	-	-	5	4	57.8	0	0
8-40	54	38.5%	-	-	5	4	64	0	0	54	38.5%	-	-	5	4	64	0	0
8-48	70.6	25.0%	0.8%	1.1%	7	6	68.4	1	0	70.6	25.0%	0.8%	1.1%	7	7	68.4	1	0
10-30	51.8	43.6%	-	-	4	4	57.8	0	0	51.8	43.6%	-	-	4	4	57.8	0	0
10-40	72	43.9%	-	-	7	6	66	0	0	72	43.9%	-	-	7	6	66	0	0
10-50	61.6	47.1%	-	-	7	6	76.8	0	0	61.6	47.1%	-	-	7	6	76.8	0	0
10-60	90.6	42.3%	-	-	12	11	88.8	0	0	90.6	42.3%	-	-	12	12	88.8	0	0
Min		25.0%	0.0%	1.1%	1	1		0	0		25.0%	0.0%	1.1%	1	1		0	0
Max		52.2%	2.8%	4.3%	12	11		3	2		52.2%	2.8%	4.3%	12	12		3	2
Avg		40.0%	1.3%	2.2%	5	4		1.08	0.42		40.0%	1.3%	2.2%	5	4		1.08	0.42
	k = 10 cuts, tLim = 1									k = 10 cuts, tLim = 5								
5-15	17	53.4%	0.0%	0.9%	2	2	39.4	2	2	17	53.4%	0.0%	0.9%	2	2	39.4	2	2
5-20	22	55.8%	0.1%	1.2%	3	3	46.6	5	4	22	55.8%	0.1%	1.2%	3	3	46.6	5	4
5-25	23	40.7%	0.6%	1.2%	4	5	50	4	1	23	40.7%	0.6%	1.2%	4	5	50	4	1
5-30	36.4	31.8%	3.3%	4.6%	7	9	54.6	3	0	36.4	31.8%	3.3%	4.6%	7	8	54.6	3	0
8-24	27.6	61.2%	0.7%	1.1%	5	7	58	3	0	27.6	61.2%	0.7%	1.1%	5	6	58	3	0
8-32	45.8	55.9%	1.0%	1.5%	13	20	74.8	5	2	45.8	55.9%	1.0%	1.5%	13	19	74.8	5	2
8-40	45.6	52.1%	0.2%	0.5%	9	15	80.6	2	0	45.6	52.1%	0.2%	0.5%	9	15	80.6	2	0
8-48	67.6	30.4%	-	-	17	33	85.6	0	0	67.6	30.4%	-	-	17	32	85.6	0	0
10-30	49.8	51.7%	0.4%	0.8%	9	14	80.8	1	0	49.8	51.7%	0.4%	0.8%	9	14	80.8	1	0
10-40	64.8	49.0%	-	-	15	25	88.8	0	0	64.8	49.0%	-	-	15	24	88.8	0	0
10-50	58.4	69.6%	-	-	14	23	94.2	0	0	58.4	69.6%	-	-	14	23	94.2	0	0
10-60	96	42.3%	0.3%	0.5%	32	62	109	1	0	96	42.3%	0.3%	0.5%	33	62	109	1	0
Min		30.4%	0.0%	0.5%	2	2		0	0		30.4%	0.0%	0.5%	2	2		0	0
Max		69.6%	3.3%	4.6%	32	62		5	4		69.6%	3.3%	4.6%	33	62		5	4
Avg		49.5%	0.7%	1.4%	11	18		2.17	0.75		49.5%	0.7%	1.4%	11	18		2.17	0.75
	k = 15 cuts, tLim = 1									k = 15 cuts, tLim = 5								
5-15	15	64.5%	0.0%	0.9%	3	4	49.6	3	3	15	64.5%	0.0%	0.9%	3	4	49.6	3	3
5-20	18.6	62.8%	0.1%	1.0%	4	6	61.8	5	3	18.6	62.8%	0.1%	1.0%	4	6	61.8	5	3
5-25	24	54.0%	0.9%	1.5%	8	14	64.2	5	2	24	54.0%	0.9%	1.5%	8	14	64.2	5	2
5-30	32	34.6%	1.1%	2.4%	11	18	65.6	3	0	32	34.6%	1.1%	2.4%	11	18	65.6	3	0
8-24	26	77.7%	1.0%	1.5%	9	16	71.6	4	1	26	77.7%	1.0%	1.5%	9	16	71.6	4	1
8-32	41.4	64.8%	0.0%	0.4%	22	41	90.4	4	3	41.4	64.8%	0.0%	0.4%	22	42	90.4	4	3
8-40	53.8	57.9%	0.2%	0.5%	22	49	91.4	2	0	53.8	57.9%	0.2%	0.5%	23	49	91.4	2	0
8-48	74.8	38.2%	-	-	40	101	99.6	0	0	73.6	38.2%	3.6%	4.2%	41	104	100.4	1	0
10-30	49.8	58.9%	0.0%	0.2%	19	39	102.8	1	1	49.8	58.9%	0.0%	0.2%	19	39	102.8	1	1
10-40	62.6	54.5%	0.0%	0.3%	28	58	109	1	1	62.6	54.5%	0.0%	0.3%	28	57	109	1	1
10-50	63.6	72.5%	0.3%	0.5%	34	81	115.2	2	0	62.6	72.5%	0.3%	0.5%	34	80	115.4	1	0
10-60	94	48.0%	0.3%	0.5%	76	202	131.2	1	0	94.2	48.0%	1.2%	1.4%	74	202	131.6	3	0
Min		34.6%	0.0%	0.2%	3	4		0	0		34.6%	0.0%	0.2%	3	4		1	0
Max		77.7%	1.1%	2.4%	76	202		5	3		77.7%	3.6%	4.2%	74	202		5	3
Avg		57.4%	0.4%	0.9%	23	53		2.58	1.17		57.4%	0.7%	1.2%	23	53		2.75	1.17

Table 5.5: Results of the restart approach on small instances

above, it is more time efficient as expected. For this method, the highest average CPU time taken is 53 seconds($k = 15$, $tLim = 5$), which is around 4 times less than

the lowest average CPU time used by the cut accumulation method 226 seconds ($k = 5$, $tLim = 1$).

	k = 20, tLim=5 seconds									k = 20, tLim=10 seconds										
Problem	Iter	LB	GapR	Sol	GapS	GapO	Clckt	CPUt	Cuts	Iter	LB	GapR	Sol	GapS	GapO	Clckt	CPUt	Cuts	Lag	BSol
c5-100	40	1928	57.1%	1953	1.1%	1.3%	33	78	159	40	1928	57.1%	1953	1.1%	1.3%	33	78	159	1924	1931
c10-100	98	1392	28.6%	1442	2.9%	3.6%	141	366	204	98	1392	28.6%	1442	2.9%	3.6%	124	331	204	1388	1402
c20-100	231	1230	45.8%	-	-	-	1270	4410	264	231	1230	45.8%	-	-	-	1176	4172	264	1219	1243
c5-200	64	3452	20.0%	3960	14.6%	14.7%	66	135	243	64	3452	20.0%	3960	14.6%	14.7%	62	115	243	3451	3456
c10-200	142	2798	20.0%	2827	0.7%	1.0%	384	1131	306	160	2798	20.0%	2821	0.5%	0.8%	468	1605	294	2796	2806
c20-200	350	2382	35.7%	-	-	-	4795	15667	356	337	2382	35.7%	-	-	-	5385	26932	384	2377	2391
Min			20.0%		0.7%	1.0%	33	78				20.0%		0.5%	0.8%	33	78			
Max			57.1%		14.6%	14.7%	4795	15667				57.1%		14.6%	14.7%	5385	26932			
Avg			34.5%		4.8%	5.2%	1115	3631				34.5%		4.8%	5.1%	1208	5539			
d5-100	63	6346	0.0%	6432	1.2%	1.4%	205	959	139	53	6346	0.0%	6432	1.2%	1.4%	92	378	154	6346	6353
d10-100	162	6326	6.5%	-	-	-	933	4102	194	131	6326	6.5%	-	-	-	1131	5693	204	6324	6355
d20-100	310	6149	8.8%	-	-	-	2840	12110	292	330	6149	8.8%	-	-	-	3956	21109	290	6143	6211
d5-200	63	12737	0.0%	12814	0.5%	0.6%	60	113	261	55	12737	0.0%	12872	1.0%	1.1%	55	109	276	12737	12744
d10-200	151	12419	0.0%	-	-	-	871	3398	297	142	12419	0.0%	12841	3.2%	3.4%	1223	6396	297	12419	12438
d20-200	365	12220	3.9%	-	-	-	3647	12428	403	369	12220	3.9%	-	-	-	4487	21655	393	12218	12269
Min			0.0%		0.5%	0.6%	60	113				0.0%		1.0%	1.1%	55	109			
Max			8.8%		1.2%	1.4%	3647	12428				8.8%		3.2%	3.4%	4487	21655			
Avg			3.2%		0.9%	1.0%	1426	5518				3.2%		1.8%	1.9%	1824	9223			
e5-100	66	12646	10.3%	-	-	-	56.1	197.6	179	58	12646	10.3%	12831	1.2%	1.5%	91.7	259.3	154	12642	12681
e10-100	159	11551	21.2%	-	-	-	302.3	1277.7	181	150	11551	21.2%	-	-	-	514.2	1649.3	217	11544	11577
e20-100	360	8379	22.9%	-	-	-	4557.7	27114.9	273	356	8379	22.9%	-	-	-	7246.8	27086.5	278	8360	8443
e5-200	59	24922	0.0%	-	-	-	30.9	49.1	252	63	24922	0.0%	-	-	-	57	82.5	261	24922	24930
e10-200	152	23295	7.7%	-	-	-	386	1804.9	284	149	23295	7.7%	23748	1.9%	1.9%	675.3	2075.5	304	23294	23307
e20-200	427	22362	26.1%	-	-	-	4322.2	22224.6	359	435	22362	26.1%	-	-	-	8602.5	29980.2	379	22356	22379
Min			0.0%		-	-	30.9	49.1				0.0%		1.2%	1.5%		82.5	154.0		
Max			26.1%		-	-	4557.7	27114.9				26.1%		1.9%	1.9%		29980.2	379.0		
Avg			14.7%		-	-	1609.2	8778.1				14.7%		1.5%	1.7%		10188.9	265.5		

Table 5.6: Results of the restart approach on medium instances, $k = 20$ and $tLim = \{5,10\}$

For the medium instances, the results confirm the observations from the small instances. Increasing the k, from 20 (Table 5.5) to 30 (Table 5.6) leads to slight improvement of 3% percent in the lower bounds for type C instances, while no improvement is found for type D instances. Also, improvement in the quality of feasible solutions is observed as the *GapS* reduces from 5.2% to 0.9% for type C instances, when k increases from 20 to 30. Increasing the SP solution time limit, $tLim$, from 5 to 10 seconds, does not lead to any notable improvement in the metrics. When compared to the cut accumulation method for type C medium instances, Table

	k = 30, tLim=5 seconds									k = 30, tLim=10 seconds										
Problem	Iter	LB	GapR	Sol	GapS	GapO	Clckt	CPUt	Cuts	Iter	LB	GapR	Sol	GapS	GapO	Clckt	CPUt	Cuts	Lag	BSol
c5-100	40	1928	57.1%	1953	1.1%	1.3%	75	215	176	40	1928	57.1%	1953	1.1%	1.3%	76	214	176	1924	1931
c10-100	93	1393	35.7%	-	-	-	441	1456	233	93	1393	35.7%	-	-	-	438	1430	233	1388	1402
c20-100	230	1231	50.0%	-	-	-	5193	21153	341	218	1231	50.0%	-	-	-	5892	26608	346	1219	1243
c5-200	74	3452	20.0%	3474	0.5%	0.6%	144	369	254	74	3452	20.0%	3474	0.5%	0.6%	148	380	254	3451	3456
c10-200	149	2798	20.0%	2823	0.6%	0.9%	1289	4854	329	145	2798	20.0%	-	-	-	1231	4753	349	2796	2806
c20-200	352	2383	42.9%	-	-	-	13028	56139	456	350	2383	42.9%	-	-	-	18651	91261	466	2377	2391
Min			20.0%		0.5%	0.6%	75	215				20.0%		0.5%	0.6%	76	214			
Max			57.1%		1.1%	1.3%	13028	56139				57.1%		1.1%	1.3%	18651	91261			
Avg			37.6%		0.8%	0.9%	3362	14031				37.6%		0.8%	1.0%	4406	20774			
d5-100	54	6346	0.0%	6432	1.2%	1.4%	258	1108	194	64	6346	0.0%	6455	1.6%	1.7%	610	2953	204	6346	6353
d10-100	149	6326	6.5%	6489	2.1%	2.6%	2364	10606	263	130	6326	6.5%	6464	1.7%	2.2%	3302	16824	257	6324	6355
d20-100	324	6149	8.8%	-	-	-	10758	50170	387	331	6149	8.8%	-	-	-	13792	73491	374	6143	6211
d5-200	65	12737	0.0%	12812	0.5%	0.6%	157	468	239	57	12737	0.0%	13303	4.4%	4.4%	205	735	300	12737	12744
d10-200	154	12419	0.0%	12611	1.4%	1.5%	2253	9213	364	157	12419	0.0%	12612	1.4%	1.6%	3754	17407	329	12419	12438
d20-200	380	12220	3.9%	-	-	-	9204	37344	463	383	12220	3.9%	-	-	-	12199	55281	451	12218	12269
Min			0.0%		0.5%	0.6%	157	468				0.0%		1.4%	1.6%	205	735			
Max			8.8%		2.1%	2.6%	10758	50170				8.8%		4.4%	4.4%	13792	73491			
Avg			3.2%		1.3%	1.5%	4166	18151				3.2%		2.3%	2.5%	5644	27782			
e5-100	60	12649	17.9%	12831	1.2%	1.4%	126.5	640.4	184	68	12649	17.9%	12831	1.2%	1.4%	251.5	830.4	169	12642	12681
e10-100	149	11553	27.3%	11770	1.7%	1.9%	1459.3	8704.2	230	152	11553	27.3%	11800	1.9%	2.1%	1935.1	7044.5	224	11544	11577
e20-100	386	8381	25.3%	-	-	-	14493.9	92738	326	376	8381	25.3%	-	-	-	26696	103171.2	326	8360	8443
e5-200	56	24923	12.5%	25495	2.3%	2.3%	75.4	236.7	287	62	24923	12.5%	25065	0.5%	0.6%	131.2	315	273	24922	24930
e10-200	149	23296	15.4%	-	-	-	1301.1	7549.2	355	173	23296	15.4%	-	-	-	2535.6	8890.2	356	23294	23307
e20-200	428	22362	26.1%	-	-	-	12045.3	65481.9	470	443	22362	26.1%	-	-	-	22484.1	82400.4	443	22356	22379
Min			12.5%		1.2%	1.4%	75.4	236.7				12.5%		0.5%	0.6%	131.2	315.0			
Max			27.3%		2.3%	2.3%	14493.9	92738.0				27.3%		1.9%	2.1%	26696.0	103171.2			
Avg			20.7%		1.7%	1.9%	4916.9	29225.1				20.7%		1.2%	1.4%	9005.6	33775.3			

Table 5.7: Results of the restart approach on medium instances, $k = 30$ and $tLim = \{5,10\}$

5.3 ($k = 20$,$tLim = 10$), this method achieves higher improvement in $GapR$ of 34.5% using much less CPU time of around 3600 seconds on average, compared to 23% for cut accumulation method with around 40,000 seconds. For type D $GapR$ improves from -7.4% from the accumulation method to 3.2%. Hence, the restart method is more effective in terms of finding improved lower bounds with less SP solution time. However, this method is not effective in finding feasible solutions as feasible solutions for none of the type E instances were found.

5.2.3 Feasible Lagrangian approach

m-n	Iter	GapR	GapS	GapO	Clckt	CPUt	Cuts	Feas	Opt
5-15	21.2	48.4%	0.1%	1.7%	0.6	0.8	37.6	5	4
5-20	26	35.3%	0.6%	1.5%	0.7	1.0	43.8	5	2
5-25	28.8	17.0%	0.7%	0.8%	0.6	0.4	37	5	2
5-30	31.4	24.9%	1.7%	1.3%	0.9	0.7	49.4	5	0
8-24	35.4	63.8%	0.2%	0.5%	1.8	2.9	70.8	5	2
8-32	48.8	45.6%	0.6%	0.5%	2.4	3.1	76.4	5	1
8-40	56.6	37.0%	0.5%	0.6%	2.9	3.9	82.2	5	0
8-48	69.4	46.2%	0.7%	0.4%	142.8	817.5	321	5	0
10-30	70	43.8%	0.5%	0.9%	6.3	13.1	128.6	5	0
10-40	76.6	44.4%	0.4%	0.6%	6.9	13.2	122	5	0
10-50	71	60.1%	0.4%	0.4%	6.6	11.2	135.4	5	1
10-60	91.8	39.1%	0.3%	0.4%	7.7	10.7	133.4	5	1
Min	21.2	17.0%	0.1%	0.4%	0.6	0.4	37.0	5.0	0.0
Max	91.8	63.8%	1.7%	1.7%	142.8	817.5	321.0	5.0	4.0
Avg	52.25	42.1%	0.6%	0.8%	15.0	73.2	103.1	5.0	1.1

Table 5.8: Results of the Feasible Lagrangian approach on small instances

In this method, we keep adding the minimal cover inequalities in the SP until a feasible solution is found. Subsequently, using the SP with all the cover inequalities and the MP, cutting plane method is used to obtain lower bounds. Table 5.8 presents the results from Feasible Lagrangian approach for small instances. This method finds feasible solutions for all the instances with an average gap of 0.6% from the optimal solutions. It also achieves an average improvement in $GapR$ of 42.1%.

For medium instances in Table 5.9, it yields high quality feasible solutions for almost all the instances with an average $GapS$ of around 1%. It also achieves positive $GapR$ for all the instances with improvement of around 23% for type C, which is same as the cut accumulation method($k = 20$, $tLim = 10$), but around 15% less than the method with restart($k = 30$, $tLim = 5$). The method uses CPU time with an average of around 2000 seconds for type C instances. This is smaller compared to

Problem	Iter	LB	Lag	GapR	Sol	BSol	GapS	GapO	Clockt	CPUt	Cuts
c5-100	46	1926	1924	29%	1976	1931	2.3%	2.6%	3	2	119
c10-100	102	1391	1388	21%	1410	1402	0.6%	1.4%	10	11	158
c20-100	210	1233	1219	58%	1245	1243	0.2%	1.0%	1727	11787	803
c5-200	57	3452	3451	20%	3469	3456	0.4%	0.5%	7	8	272
c10-200	140	2797	2796	10%	2830	2806	0.9%	1.2%	23	21	227
c20-200	348	2383	2377	43%	2403	2391	0.5%	0.8%	216	426	585
c10-400	174	5592	5592	0%	5603	5597	0.1%	0.2%	65	58	512
c20-400	411	4776	4775	14%	4795	4782	0.3%	0.4%	278	272	644
c40-400	1234	4237	4236	11%	4250	4245	0.1%	0.3%	2446	5247	1127
Min				0%			0%	0%	3	2	
Max				58%			2%	3%	2446	11787	
Average				23%			1%	1%	531	1981	
d5-100	47	6346	6346	0%	6432	6353	1.2%	1.4%	3	2	108
d10-100	138	6325	6324	3%	6563	6355	3.3%	3.8%	13	12	146
d20-100	391	6147	6143	6%	6580	6211	5.9%	7.0%	74	71	277
d5-200	57	12737	12737	0%	12814	12744	0.5%	0.6%	5	4	207
d10-200	139	12419	12419	0%	12615	12438	1.4%	1.6%	24	22	273
d20-200	411	12219	12218	2%	12683	12269	3.4%	3.8%	145	132	456
d10-400	157	24957	24956	8%	25136	24969	0.7%	0.7%	59	51	520
d20-400	395	24554	24553	3%	24949	24587	1.5%	1.6%	272	268	974
Min				0%			1%	1%	3	2	
Max				8%			6%	7%	272	268	
Average				3%			2%	3%	74	70	
e5-100	60	12648	12642	15%	12831	12681	1.2%	1.4%	5	6	182
e10-100	169	11549	11544	15%	11789	11577	1.8%	2.1%	17	18	196
e20-100	-	-	-	-	-	-	-	-	10000	10000	-
e5-200	76	24924	24922	25%	25112	24930	0.7%	0.8%	7	7	217
e10-200	181	23296	23296	0%	23603	23307	1.3%	1.3%	65	71	788
e20-200	-	-	-	-	-	-	-	-	10000	10000	-
e10-400	164	45740	45740	0%	45933	45745	0.4%	0.4%	964	1522	4345
Min				0%			0%	0%	5	6	
Max				25%			2%	2%	10000	10000	
Average				11%			1%	1%	3008	3089	

Table 5.9: Results of the Feasible Lagrangian approach on medium instances

around 14,000 seconds used by the restart method. Due to the time efficiency of this method compared to the other methods, we were able to solve some larger instances with $n = 400$ and obtained quality feasible solutions with less than 1% $GapS$. Thus, this methodology proves more effective in yielding quality feasible solutions and in improving the lower bounds with much less computational time.

5.3 Results comparison of the proposed methodologies

In this section, we compare the performance of the two variants of the Modified Lagrangian approach and the Feasible Lagrangian approach, for both small and medium instances.

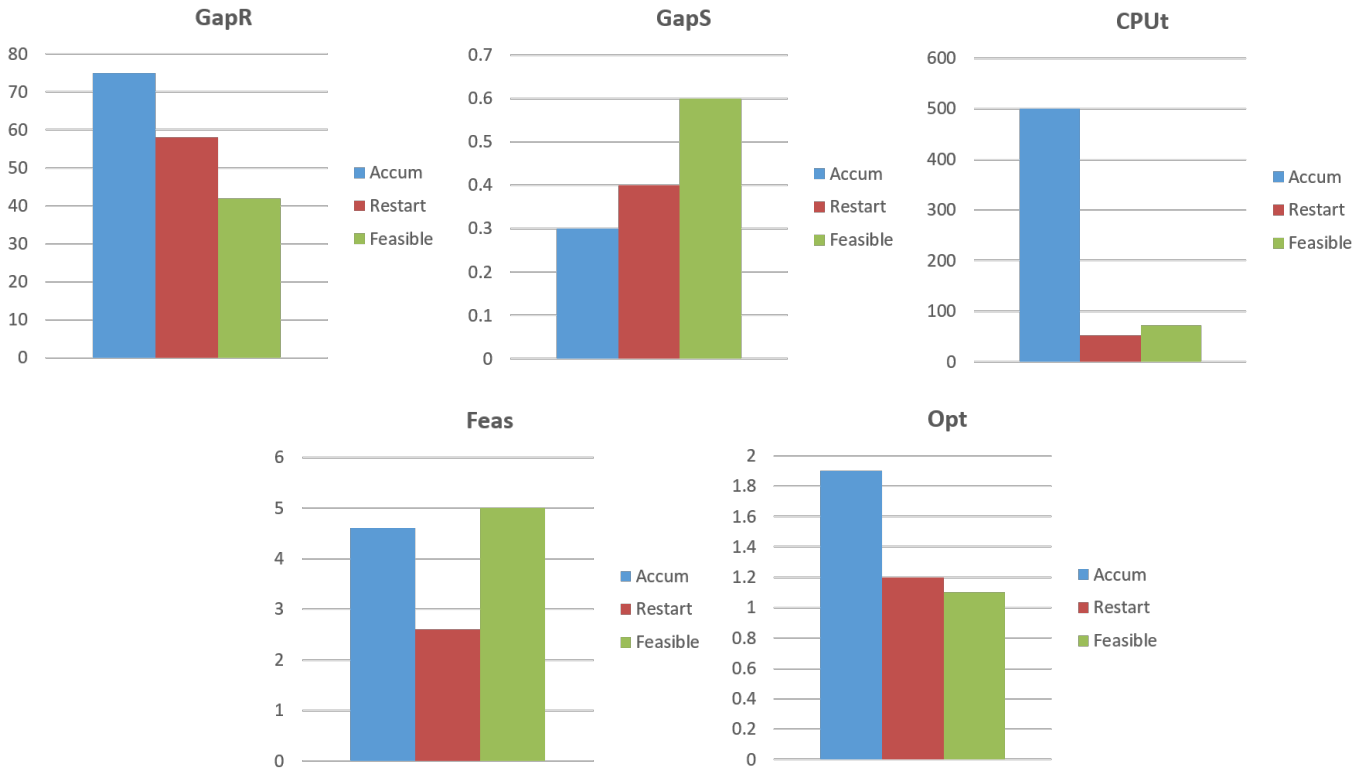

Figure 5.1: Comparison of results from different approaches for small instances

Figure 5.1 above compares the average results for small instances for the cut accumulation approach with setting $k = 15$ and $tLim = 5$ and the restart approach with setting $k = 15$ and $tLim = 1$. The comparison shows that the cut accumula-

tion approach achieves the highest reduction in the optimality gap, *GapR*, of around 75%. It finds feasible solutions for almost all the instances within an average gap of 0.3% from the optimal solutions. However, the CPU time used by this method is very high, around 500 seconds on average, compared to 53 seconds for the restart approach. Restart method also achieves high improvement in *GapR* of around 58%, but it is less successful in finding feasible solutions. It finds feasible solutions for only 2.6 out of 5 instances for each problem size on average, compared to 4.6 for the accumulation approach. Restart approach offers efficiency in solution time compared to the accumulation approach. Feasible Lagrangian approach achieves the least improvement in *GapR* and lower quality of feasible solutions compared to the previous two approaches, but it is able to yield feasible solutions for all the instances, which the other two approaches failed to achieve.

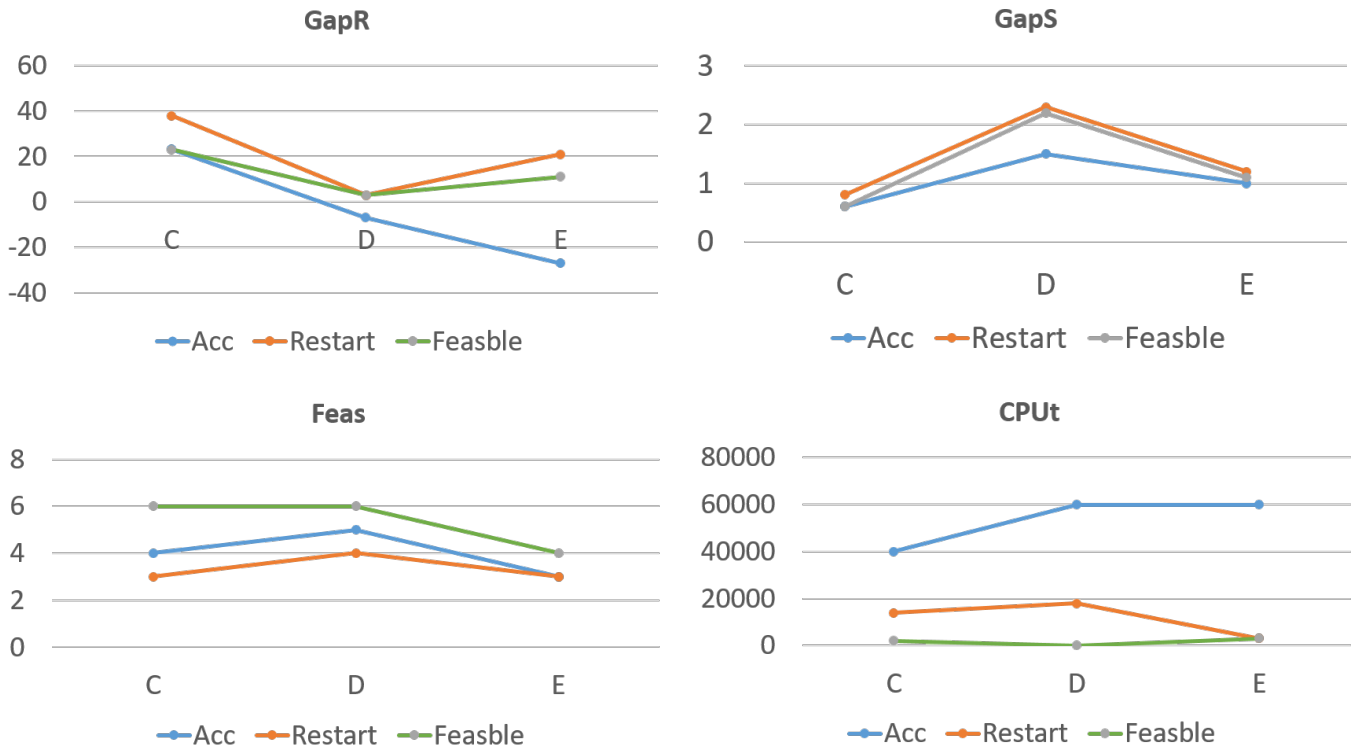

Figure 5.2: Comparison of results from different approaches for medium instances

For the medium instances, the accumulation method ($k = 20, tLim = 10$) leads to negative $GapR$, i.e., LB bounds being lower than the Lagrangian bounds, Lag. This is because of the problem becoming too difficult to be solved near optimality in the imposed solution time limit. The restart approach ($k = 30, tLim = 5$) achieves the highest improvement in lower bounds for all data types, but it is not able to find feasible solutions for around half of the instances. Feasible Lagrangian approach shows the best performance on medium instances. It achieves $GapR$ improvement nearly as good as the restart method and finds feasible solutions to almost all the instances, with less than an average gap of 2% from the best-known solutions. It also uses very less CPU time compared to the other two approaches.

5.4 Results Summary

The results from testing our methodologies on the small and medium instances demonstrate that our proposed approach leads to improved lower bounds and good quality feasible solutions. For the small instances, the best results are achieved by Modified Lagrangian approach with cut accumulation, obtaining at least 60% improvement in the lower bound for all the instances in all different settings (k,$tLim$). High quality feasible solutions are obtained, that lie within 1% of the optimal solutions. However, as the problem size increases, the subproblem difficulty also increases. As the subproblem is not solved to optimality, it leads to a decline in the performance. Modified Lagrangian approach with restart is able to consistently achieve improvement in lower bounds in both small and medium instances because the subproblem is solved near optimality in short time. However, its success in finding feasible solutions is less than the cut accumulation method. The Feasible Lagrangian approach proves to be the most robust of all, with an average improvement in lower bounds ranging from 3% to 23% for type D. It was also able to yield quality feasible solutions for almost all the instances within 1% gap from the best-known solution in the literature (Yagiura et al. 2006).

Chapter 6

Conclusion

In this thesis, we proposed two main modifications to the standard Lagrangian approach, with the objective of achieving improved bounds and quality feasible solutions. We presented a general framework based on the idea that if the information generated by the subproblem solutions can be used to strengthen the subproblem with valid cuts, it may lead to feasible solutions for the original problem as well as tighter bounds. The general strategy we proposed is to repeatedly solve the subproblem at each iteration of the Lagrangian procedure and strengthen it with violated valid inequalities so that its solutions can be pushed towards feasibility for the original problem.

To test the proposed approach, we focused on the generalized assignment problem and generated valid cuts based on minimal cover inequalities. In the first methodology, at each iteration of the cutting plane method we repeatedly added minimal cover inequalities to strength the subproblem. As the subproblem became more difficult to solve with the added inequalities, we imposed a time limit on the subproblem

solution. For this method we proposed two variants: Modified Lagrangian approach with accumulation of cover inequalities, where the cover inequalities accumulate in the subproblem from one Lagrangian iteration to the other, and Modified Lagrangian approach with restart, where the inequalities added in the subproblem in one cutting plane iteration are discarded in the following iteration. For the second methodology, Feasible Lagrangian Method, first the minimal cover inequalities are added to the subproblem until a feasible solution to the GAP is obtained. Then the subproblem, with all the added inequalities, is solved in a Lagrangian fashion to find a lower bound.

We tested these methods on small and medium size benchmark GAP instances. For the first methodology, we used different settings, $(k, tLim)$, where k is the number of minimal cover inequality generation iterations performed on the subproblem and $tLim$ is the time limit imposed on the subproblem solution. In the second method, the subproblem was solved to optimality, without imposing any time limit on the solution. The results demonstrate that our proposed approach succeeds in achieving improved bounds and quality feasible solutions. While all the methods were able to achieve improvement in the lower bounds and quality feasible solutions for small instances, for the medium instances, Feasible Lagrangian Method was able to consistently achieve improvement in lower bounds, with average improvement ranging from 3% to 23% for type D. It was also able to yield quality feasible solutions for almost all the instances within 1% gap from the best-known solution in the literature.

References

Barbas, J., & Marin, A. (2004). Maximal covering code multiplexing access telecommunication networks. *European Journal of Operational Research*, *159*(1), 219-238.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252.

Boffey, T. B. (1989). Location problems arising in computer networks. *The Journal of the Operational Research Society*, *40*(4), 347–354.

Bressoud, T. C., Rastogi, R., & Smith, M. A. (2003). Optimal configuration for BGP route selection. *IEEE INFOCOM*, *2*, 916-926.

Cattrysse, D. G., Salomon, M., & Van Wassenhove, L. N. (1994). A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, *72*, 167-174.

Ceselli, A., & Giovanni, R. (2006). A branch-and-price algorithm for the multilevel generalized assignment problem. *Operations Research*, *54*(6), 1172-1184.

Chu, P., & Beasley, J. (1997). A genetic algorithm for the generalised assignment problem. *Computers and Operations Research*, *24*(1), 17 - 23.

Cohen, R., Lewin-Eytan, L., Naor, J. S., & Raz, D. (2015). Near optimal placement of virtual network functions. *2015 IEEE Conference on Computer Communications (INFOCOM)*, 1346-1354.

Dobson, G., & Nambimadom, R. S. (2001). The batch loading and scheduling problem. *Operations Research*, *49*(1), 52-65.

Fisher, M. L. (1985). An applications oriented guide to Lagrangian relaxation. *INFORMS Journal on Applied Analytics*, *15*(2), 10-21.

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer pro-

gramming problems. *Management Science*, *50*(12), 1861-1871.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*(2), 109-124.

Foulds, L., & Wilson, J. (1997). A variation of the generalized assignment problem arising in the New Zealand dairy industry. *Annals of Operations Research*, 105114.

Ghoniem, A., Flamand, T., & Haouari, M. (2016). Optimization-based very large-scale neighborhood search for generalized assignment problems with location/allocation considerations. *INFORMS Journal on Computing*, *28*(3), 575-588.

Ghoniem, A., Tulay, F., & Haouari, M. (2016). Exact solution methods for a generalized assignment problem with location/allocation considerations. *INFORMS Journal on Computing*, *28*(3), 589-602.

Guignard, M., & Rosenwein, M. B. (1989). An improved dual based algorithm for the generalized assignment problem. *Operations Research*, *37*(4), 658–663.

Haddadi, S. (1999). Lagrangian decomposition based heuristic for the generalized assignment problem. *INFOR: Information Systems and Operational Research*, *37*(4), 392-402.

Haddadi, S., & Ouzia, H. (2004). Effective algorithm and heuristic for the generalized assignment problem. *European Journal of Operational Research*, *153*(1), 184 - 190.

Harvey, N. J., Ladner, R. E., Lovsz, L., & Tamir, T. (2006). Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms*, *59*(1), 53 - 78.

Held, M., & Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, *18*(6), 1138-1162.

Jeet, V., & Kutanoglu, E. (2007). Lagrangian relaxation guided problem space

search heuristics for generalized assignment problems. *European Journal of Operational Research*, *182*(3), 1039 - 1056.

Klastorin, T. D. (1979). Noteon the maximal covering location problem and the generalized assignment problem. *Management Science*, *25*(1), 107-112.

Krumke, S. O., & Thielen, C. (2013). The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, *228*(1), 46 - 55.

Litvinchev, I., Mata, M., Saucedo, J., & Rangel, S. (2017). Improved Lagrangian bounds and heuristics for the generalized assignment problem. *Journal of Computer and Systems Sciences International*, *56*(5), 803-809.

Litvinchev, I., Rangel, S., & Saucedo, J. (2010). A Lagrangian bound for many-to-many assignment problems. *Journal of Combinatorial Optimization*, *19*(3), 241-257.

Liu, L., Mu, H., Song, Y., Luo, H., Li, X., & Wu, F. (2012). The equilibrium generalized assignment problem and genetic algorithm. *Applied Mathematics and Computation*, *218*(11), 6526 - 6535.

Luo, L., Chakraborty, N., & Sycara, K. (2015). Distributed algorithms for multi-robot task assignment with task deadline constraints. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 876-888.

Maio, A. D., & Roveda, C. (1971). An all zero-one algorithm for a certain class of transportation problems. *Operations Research*, *19*(6), 1406-1418.

Martello, S., & Toth, P. (1981). Heuristic algorithms for the multiple knapsack problem. *Computing*, *27*, 93–112.

Mazzola, J. B. (1989). Generalized assignment with nonlinear capacity interaction. *Management Science*, *35*(8), 923-941.

Mazzola, J. B., & Neebe, A. W. (2012). A generalized assignment model for dynamic

supply chain capacity planning. *Naval Research Logistics*, *59*(6), 470-485.

Mitrovi-Mini, P. A. P., S. (2009). Local search intensified: Very large-scale variable neighborhood search for the multi-resource generalized assignment problem. *Discrete Optimization*, *6*(4), 370 - 377.

Narciso, M. G., & Lorena, L. A. N. (1999). Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, *114*(1), 165 - 177.

Nauss, R. M. (2003). Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing*, *15*(3), 249-266.

Nowakovski, J., Schwarzler, W., & Triesch, E. (1999). Using the generalized assignment problem in scheduling the ROSAT space telescope. *European Journal of Operational Research*, *112*(3), 531-541.

Ogbe, E., & Li, X. (2017). A new cross decomposition method for stochastic mixed-integer linear programming. *European Journal of Operational Research*, *256*(2), 487 - 499.

Oncan, T. (2007). A survey of the generalized assignment problem and its applications. *INFOR: Information Systems and Operational Research*, *45*(3), 123-141.

Ozbakir, L., Baykasoglu, A., & Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, *215*(11), 3782 - 3795.

Pessoa, A. A., Hahn, P. M., Guignard, M., & Zhu, Y.-R. (2010). Algorithms for the generalized quadratic assignment problem combining Lagrangean decomposition and the reformulation-linearization technique. *European Journal of Operational Research*, *206*(1), 54-63.

Pirkul, H. (1986). An integer programming model for the allocation of databases

in a distributed computer system. *European Journal of Operational Research*, *26*(3), 401-411.

Ross, G. T., & Soland, R. M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, *8*(1), 91-103.

Ross, G. T., & Soland, R. M. (1977). Modeling facility location problems as generalized assignment problems. *Management Science*, 345-357.

Sahni, S., & Gonzalez, T. (1976, July). P-complete approximation problems. *J. ACM*, *23*(3), 28-32.

Shi, W., & Hong, B. (2011). Towards profitable virtual machine placement in the data center. *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 138-145.

Shtubt, A. (1989). Modelling group technology cell formation as a generalized assignment problem. *International Journal of Production Research*, *27*(5), 775-782.

Van Roy, T. J. (1983). Cross decomposition for mixed integer programming. *Mathematical Programming*, *25*(1), 46–63.

Van Roy, T. J. (1986). A cross decomposition algorithm for capacitated facility location. *Operations Research*, *34*(1), 145-163.

Yagiura, M., Ibaraki, T., & Glover, F. (2006). A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, *169*(2), 548 - 569.