Journal of Advances in Science and Technology

Vol. VI, Issue No. XI, November-2013, ISSN 2230-9659

AN EVALUATION UPON THEORETICAL BACKGROUND FOR USING THE PRINCIPLE OF DECOMPOSITION METHODS FOR INTEGER LINEAR PROGRAMMING

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

Journal of Advances in Science and Technology
Vol. VI, Issue No. XI, November-2013, ISSN 2230-9659

An Evaluation upon Theoretical Background for Using the Principle of Decomposition Methods for Integer Linear Programming

Pal Dilip Kumar

Research Scholar, Sai Nath University, Ranchi, Jharkhand

Abstract – In this paper, we present a theoretical background for using the principle of decomposition to solve mixed integer linear programs (MILP). We focus on the common threads among three traditional methods for generating approximations to the convex hull of feasible solutions to an MILP. These include a method employing an outer approximation, the cutting-plane method, as well as two related methods employing inner approximations, the Dantzig-Wolfe method and the Lagrangian method. We then extend these traditional methods by allowing for the use of both outer and inner approximation simultaneously. This leads to the development of two bounding methods that generate even stronger bounds, known as price-and-cut and relax-and-cut.

We describe DIP (Decomposition for Integer Programming), a new open-source software framework that provides the algorithmic shell for implementation of these methods. DIP has been designed with the goal of providing a user with the ability to easily utilize various traditional and integrated decomposition methods while requiring only the provision of minimal problem-specific algorithmic components.

----------♦----------

INTRODUCTION

Within the field of mathematical programming, discrete optimization has become the focus of a vast body of research and development due to the increasing number of industries now employing it to model the decision analysis for their most complex systems. Mixed integer linear programming problems involve minimizing (or maximizing) the value of some linear function over a polyhedral feasible region subject to integrality restrictions on some of the variables. More formally, a *mixed integer linear program* (MILP) can be defined as

$$\min_{x \in \mathbb{R}^n} \left\{ c^\top x \mid Ax \geq b, x_i \in \mathbb{Z}\ \forall i \in I \right\},$$

where $c \in \mathbb{Q}^n$ is a given cost vector, $A \in \mathbb{Q}^{m \times n}$ is the constraint matrix, $b \in \mathbb{Q}^m$ is the right hand side, and $I \subseteq \{1, \ldots, n\}$ is the index set of variables that are restricted to integer values.

Two important special cases are when $I = \{1, \ldots, n\}$, which we refer to as a *(pure) integer linear-program* (ILP) and when $I = \emptyset$, which we refer to as a *linear program* (LP).

Solving an MILP is known to be an $\mathcal{NP}$-hard problem in general. However, due to recent breakthroughs in both the underlying theory and available computer implementations, discrete optimization is now a viable tool for optimizing some of the most complex systems. We are just now beginning to understand the impact that discrete optimization can have in helping organizations optimize the efficiency of their processes. In the past two decades, MILP has seen widespread adoption in a large and diverse array of industries, including logistics, finance, medical research, engineering design, retail, and many others.

In the following paragraphs, we attempt to put into context the direction of our research. For this purpose, we assume the reader has a working knowledge of the theory and practice of integer programming.

One of the most successful algorithms developed to date for solving MILPs is the *branch-and- bound* method. Branch and bound is a divide-and-conquer approach that reduces the original problem to a series of smaller subproblems and then recursively solves each subproblem. This dissertation focuses on

the development of a theoretical and computational framework for computing strong bounds to help improve the performance of branch-and-bound methods. Most bounding procedures for MILPs are based on the iterative construction and improvement of polyhedral approximations of $\mathcal{P}$, the *convex hull* of feasible solutions. Solving an optimization problem over such a polyhedral approximation, provided it fully contains $\mathcal{P}$, produces a bound that can be used to drive a branch-and-bound algorithm. The effectiveness of the bounding procedure depends largely on how well $\mathcal{P}$ can be approximated. The most straightforward approximation is the *continuous approximation*, consisting simply of the linear constraints present in the original formulation. The bound resulting from this approximation is frequently too weak to be effective, however. In such cases, it can be improved by dynamically generating additional polyhedral information that can be used to augment the approximation.

Traditional dynamic procedures for augmenting the continuous approximation can be grouped roughly into two categories. *Cutting-plane methods* improve the approximation by dynamically generating half-spaces that contain $\mathcal{P}$ but not the continuous approximation, i.e., valid inequalities. These half-spaces are then intersected with the current approximation, thereby improving it. With this approach, valid inequalities are generated by solution of an associated *separation problem.* Generally, the addition of each valid inequality reduces the hyper volume of the approximating polyhedron, resulting in a potentially improved bound. Because they dynamically generate part of the description of the final approximating polyhedron as the intersection of half-spaces (an *outer representation*), we refer to cutting-plane methods as *outer approximation methods.* Traditional *column-generation methods*, on the other hand, improve the approximation by dynamically generating the extreme points of a polyhedron containing $\mathcal{P}$, which is again intersected with the continuous approximation, as in the cutting-plane method, to yield a final approximating polyhedron. In this case, each successive extreme point is generated by solution of an associated *optimization problem* and at each step, the hypervolume of the approximating polyhedron is increased. Because decomposition methods dynamically generate part of the description of the approximating polyhedron as the convex hull of a finite set (an *inner representation*), we refer to these methods as *inner approximation methods.*

THE PRINCIPLE OF DECOMPOSITION

We now formalize some of the notions described in the introduction. Implementing a branch-and- bound algorithm for solving an ILP requires a procedure that generates a lower bound on the optimal value z_{IP}. The most commonly used method of bounding is to solve the linear programming relaxation obtained by removing the integrality requirement from the ILP formulation. The *LP bound* is given by

$$z_{\mathrm{LP}} = \min_{x \in \mathbb{R}^n} \left\{ c^\top x \mid Ax \geq b \right\} = \min_{x \in \mathcal{Q}} \left\{ c^\top x \right\}$$

and is obtained by solving a linear program with the original objective function cover the polyhedron $\mathcal{Q}$. It is clear that $z_{\mathrm{LP}} \leq z_{\mathrm{IP}}$ since $\mathcal{P} \subseteq \mathcal{Q}$. This LP relaxation is usually much easier to solve than the original ILP. but z_{LP} may be arbitrarily far away from z_{IP} in general, so we need to consider more effective procedures.

A description that is represented with a polynomial number of variables and constraints is called *compact.* In most cases, the description of $\mathcal{Q}$ is compact, it can be represented explicitly, and the bound computed using a standard linear programming algorithm. To improve the LP bound, decomposition methods construct a second approximating polyhedron that can be intersected with $\mathcal{Q}$ to form a better approximation. Unlike $\mathcal{Q}$, this second polyhedron usually has a description of exponential size, and we must generate portions of its description dynamically. Such a dynamic procedure is the basis both for cutting-plane methods, which generate an outer approximation, and for column-generation methods, such as the Dantzig-Wolfe method and the Lagrangian method, which generate inner approximations.

we consider the relaxation-

$$\min_{x \in \mathbb{Z}^n} \left\{ c^\top x \mid A'x \geq b' \right\} = \min_{x \in \mathcal{F}'} \left\{ c^\top x \right\} = \min_{x \in \mathcal{P}'} \left\{ c^\top x \right\},$$

Where $\mathcal{F} \subset \mathcal{F}' = \{x \in \mathbb{Z}^n \mid A'x \geq b'\}$ for some $A' \in \mathbb{Q}^{m' \times n}, b' \in \mathbb{Q}^{m'}$ and $\mathcal{P}'$ is the convex hull of $\mathcal{F}'$. Along with $\mathcal{P}'$ is associated a set of *side constraints* $[A'', b''] \in \mathbb{Q}^{m'' \times (n+1)}$ such that $\mathcal{Q} = \{x \in \mathbb{R}^n \mid A'x \geq b', A''x \geq b''\}$. We denote by $\mathcal{Q}'$ the polyhedron described by the inequalities $[A', b']$ and by $\mathcal{Q}''$ the polyhedron described by the inequalities $[A'', b'']$. Thus, $\mathcal{Q} = \mathcal{Q}' \cap \mathcal{Q}''$ And $\mathcal{F} = \{x \in \mathbb{Z}^n \mid x \in \mathcal{P}' \cap \mathcal{Q}''\}$. We often refer to $\mathcal{Q}'$ as the *relaxed polyhedron.* For the decomposition to be effective, we must have that $\mathcal{P}' \cap \mathcal{Q}'' \subset \mathcal{Q}$, so that the bound obtained by optimizing over $\mathcal{P}' \cap \mathcal{Q}''$ is at least as good as the LP bound and strictly better for some objective functions. The description of $\mathcal{Q}''$ must also be

compact so that we can construct it explicitly. Finally, we assume that there exists an *effective* algorithm for optimizing over $\mathcal{P}'$ and thereby, for separating arbitrary real vectors from $\mathcal{P}'$. We are deliberately using the term *effective* here to denote an algorithm that has an acceptable average-case running time, since this is more relevant than worst-case behavior in our computational framework. Note, throughout this research, we are assuming that the efficiency of the algorithm used for solving $\text{OPT}(\mathcal{P}', c)$ is not affected by the structure of the cost vector c.

Traditional decomposition methods can all be viewed as techniques for iteratively computing the bound

$$z_D = \min_{x \in \mathcal{P}'} \left\{ c^\top x \mid A'' x \geq b'' \right\} = \min_{x \in \mathcal{F}' \cap \mathcal{Q}''} \left\{ c^\top x \right\} = \min_{x \in \mathcal{P}' \cap \mathcal{Q}''} \left\{ c^\top x \right\}.$$

COMPUTATIONAL BACKGROUND FOR DECOMPOSITION METHODS

Sometime around the late 1980s, the recognition of mixed integer programming models as an important paradigm for solving real business problems had encouraged a number of commercial software vendors towards a large investment in tackling the solution of bigger and more difficult MILPs. The computational strides made in developing methods for solving generic MILPs throughout the 1990s were dramatic. Despite this, there are still many classes of important MILPs that are extremely difficult for today's best solvers. Exploiting the special structure of certain models has long been an active field of research.

In the early 1990s, several research groups recognized the potential of abstracting the general branch-and-cut framework in the form of a software framework with user *hooks* for adding problem-specific routines. This led to the development of several popular frameworks, for example, MINTO, MIPO , bc-opt , SIP , ABACUS , and SYMPHONY. The majority of these frameworks are focused on providing an infrastructure for implementing branch-and-bound algorithms in which the user could provide their own specific methods for customizing both the branching and the bounding operations. In the 1990s, most of the work using these frameworks focused on problem-specific cutting planes that were incorporated into the framework to produce a branch-and-cut algorithm.

At the same time, column-generation methods were also gaining popularity. Of the list above, the only frameworks that provided some facility for branch-and-price were MINTO. ABACUS, and SYMPHONY In all cases, the end-goal was to automate the most common elements of the branch- and-cut (or price) algorithm, allowing the user to focus on the problem-specific hooks. In addition, some of the frameworks (SYMPHONY, for example) were designed in a generic manner to allow- complete flexibility for the user to override just about every algorithmic function. This added to the wide array of problem types and methods that could be implemented within the frameworks. Much less common in these frameworks was support for integrated methods like branch-and-price- and-cut. Although there is some early mention of these ideas, there are very few implementations discussed in the literature that use any of these frameworks.

DECOMPOSITION METHODS

In this paper, we present two major categories of methods for generating bounds by iterative construction of polyhedral approximations of the convex hull of feasible solutions to some MILP.

Traditional Decomposition Methods - In this paper, we review three classical approaches that take advantage of implicit generation of a polyhedral approximation. By finding the common threads among each of these methods, we have generalized the overall approach into four steps. The first step is an initialization step where we define the initial polyhedral approximation of $\mathcal{P}'$. This is done using either valid inequalities, in the case of outer methods, or using extreme points, in the case of inner methods. In the second step, the *master problem* is solved which generates primal and or dual solution information over the current approximation. Then, in the third step, a *subproblem* is solved which will be used to improve the approximation. In the case of outer methods, the subproblem is a separation problem $\text{SEP}(\mathcal{P}', x)$, where we try to find valid inequalities (for $\mathcal{P}'$) that are violated by the current primal solution. In the case of inner methods, the subproblem is an optimization problem $\text{OPT}(\mathcal{P}', c)$ where we try to find extreme points of $\mathcal{P}'$ using the current dual solution. In the fourth and final step, we use the inequalities (or extreme points) found in step three and update the current approximation. By viewing all of these methods within the same conceptual framework, we are able to draw several connections among the methods.

Integrated Decomposition Methods - While traditional decomposition approaches build either an inner *or* an outer approximation, *integrated decomposition methods* build both an inner *and* an outer approximation. These methods follow the same basic logic as traditional decomposition methods, except that the master problem is required to generate both primal *and* dual solution information,

and the subproblem can be either a separation problem *or* an optimization problem.

Decompose-and-Cut - In this paper we extend this idea to the traditional cutting-plane method and examine a relatively unknown decomposition-based algorithm we refer to as *decompose-cind- cut.*

First, we review the well-known *template paradigm* for separation and introduce a concept called *structured separation.* Then, we describe a separation algorithm called *decompose-and-cut* that is closely related to the integrated decomposition methods.

STANDARD MILP CUTTING PLANES FOR INNER METHODS

Generic cutting planes have long been one of the most important factors in the success of branch- and-cut codes. In the past decade, there have been major advances in increasing the variety and strength of valid inequalities that can be generated and used to solve generic MILPs. Up until now, the use of cutting planes with inner methods such as price-and-cut has been limited to specific applications. When structured cuts are known and provably facet-defining, it is typical for them to dominate generic classes in terms of strength. In real-world applications, however, it is common for the user not to have any knowledge of the polyhedral structure of the model. EVen with knowledge of a specific class of facet-defining valid inequalities, an efficient separation algorithm may not be known, forcing one to rely on heuristics that give no guarantee of quality. In such cases, methods for generating generic classes of valid inequalities may be necessary to solve a given problem. This fact has long been understood with respect to branch-and-cut codes, and all solvers include at least a basic implementation of separation algorithms for the most common classes of valid inequalities. However, until now, there have been no branch-and-price-and-cut frameworks that have successfully allowed for direct integration of generic cutting planes.

CONCLUSION

The success of decomposition-based methods in real applications shows the potential for this area to make a positive impact on the field of mathematical programming. Until now, the study of these methods has been somewhat disconnected and application-specific. This thesis has provided a way to consider these various methods under one framework. The software framework DIP has the potential for opening the door to various new areas of research related to decomposition methods.

The ease-of-use and extensibility of the framework should allow users to compare and contrast these methods, as well as numerous extensions, under one uniform computational environment. From this, computational studies can be easily conducted which will allow practitioners to make intelligent choices between the many possible algorithmic variants. Along with this, we can also gain much insight into further areas of computational research. Specifically, the use of hybrid methods that combine different components of the numerous algorithms seems promising. DIP allows the user to easily experiment with these ideas which should generate several new avenues of research.

For every successful application we show in this study, there are twice as many applications we tried where the performance of the inner approximation methods (even with integrated cuts) performs quite poorly when compared to more standard approaches like branch-and-cut.

REFERENCES

- A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 2006.
- D.Applegate, R. Bixby, V. Chv´atal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Mathematical Programming*, 97:91–153, 2003.
- E.Balas, S. Ceria, and G. Cornu´ejols. Mixed 0-1 programming by lift and-project in a branchand- cut framework. *Management Science*, 42:1229–1246, 1996.
- G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.
- H.Crowder, E. L. Johnson, and M. W. Padberg. Solving large scale zero-one linear programming problems. *Operations Research*, 31:803–834, 2003.
- J.E. Beasley. Lagrangean relaxation. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Optimization*. Wiley, 1993.
- L.A. Wolsey. *Integer Programming*. Wiley, New York, 1998.
- M.L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 2001.
- R.Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice - closing the gap. In *Proceedings of the 19th IFIP TC7 Conference on System Modelling and Optimization*, pages 19–50, Deventer, The Netherlands, The Netherlands, 2000. Kluwer, B.V.

- T.K. Ralphs and M. Guzelsoy. The SYMPHONY callable library for mixed-integer linear programming. In *Proceedings of the Ninth INFORMS Computing Society Conference*, pages 61–76, 2005.