Symbolic Algorithms for Integer Programming: From Logic Specification to Solvers

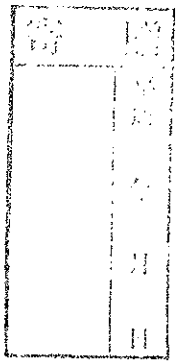
March 1999

Qiang Li

Symbolic Algorithms for Integer Programming: From Logic Specification to Solvers

March 1999

Qiang Li



Symbolic Algorithms for Integer Programming: From Logic Specification to Solvers

Doctoral Dissertation

Doctoral Program in Engineering
University of Tsukuba

Qiang Li

Promotor: Professor Dr. Tetsuo Ida

Referent: Professor Dr. Shigeru Igarashi

Professor Dr. Hisao Kameda

Professor Dr. Jiro Tanaka

Dr. Aart Middeldorp

Acknowledgments

I owe a great debt to my supervisor Professor Tetsuo Ida. He provided me a highly stimulating research environment and gave me his understanding, encouragement, support and tolerance. His guidance has been of vital importance to the successes of my research. He read my whole thesis and gave me many helpful comments.

I would like to thank Dr. Aart Middeldorp, Dr. Manuel M. T. Chakravarty and Dr. Gabriele Keller. They gave me many useful comments on my work and presentation for my research. Dr. Manuel M. T. Chakravarty also read an earlier draft of the dissertation and gave me many suggestions. I also would like to thank all past and present members of SCORE (Symbolic COmputation REsearch) group. They provided the friendship and great helps to me, a foreign student from China.

It is my great pleasure to thank Dr. Yike Guo. I am very lucky to have this friend. He introduced me to the Gröbner basis methods in integer programming. His insight and experience have been a great influence to my entire work. He repeatedly gave me detailed and insightful advice on my research. Without his helps, I could not complete this dissertation. I am also grateful to Professor John Darlington and all members of his group for their hospitality during my visit to Imperial College Parallel Computing Centre at London, which proved very important for my research.

Finally, I thank my parents, my wife and son for all their love and supports. During the three-year doctoral research, my wife, XueMei Gou, has been bearing with me through the ups and downs of my work. Without her support and self-sacrifice, this work would never have been possible.

Contents

Chapter 1

Introduction

Integer programming (IP) is an important mathematical optimisation technique that attempts to find an optimal solution under constraints so that the objective value is minimal or maximal. Optimisation problems that can be solved with IP are important in operations research and computer science, such as the knapsack problem, the traveling salesman problem, the job scheduling problem, the transportation problem and neural network optimisation problem. Since the late 1940s there have appeared a large number of articles that are concerned with IP problems. However, IP problems belong to the most difficult problems in operations research. Moreover for real world IP problems, such as stochastic IP problems, there did not exist an efficient solving procedure until now. This has limited the practical usefulness of IP techniques.

This thesis identifies two major deficiencies that have led to the situation: Firstly, the previous approach was limited by the lack of modelling techniques. Both the specification and the analysis of the solution of real world IP problems are highly complex. Hence, potential non-specialist users like managers are discouraged from formulating the problems or interpreting the results. Secondly, progress has been hampered by the lack of efficient algorithms to tackle real world IP problems and by the absence of good high-performance computing environments that compute solutions reasonably fast.

This thesis is concerned with solving real world IP problems in theory and practice. It introduces a logic modelling language for specifying real world IP problems and proposes an algorithm that translates the formulas of the modelling language into IP formulas. Furthermore, the thesis proposes a new algebraic algorithm for IP based on the Gröbner basis technique. The new algorithm, called the *Minimised Geometric Buchberger Algorithm* (MGBA), combines various newly developed symbolic IP solvers such as Hosten and Sturmfels' method GRIN and Thomas' truncated geometric Buchberger algorithm to achieve an efficient and general IP solver. MGBA has also successfully and efficiently solved a class of stochastic IP problem, called chance constrained integer programming

(CIP), where uncertainty of a decision problem is given by stating the constraints with probabilities. Chance constrained IP is a challenging research subject where traditional IP solving mechanisms fail to provide efficient solving methods. Overall, the thesis discusses a framework for solving real world complex IP problems: It includes a logic modelling language for specifying IP problems, a transformation tool for translating the formulas in the language into IP formulas, and two symbolic constraint solvers based on MGBA for IP and stochastic IP.

The contributions of this thesis are in three parts: It demonstrates how to model IP in logic, it introduces a minimised geometric Buchberger algorithm for IP, and it applies MGBA to stochastic IP. We discuss these three parts as follows.

Modelling IP with logic The classical algebraic modelling approach for integer programming (IP) is not suitable for some real world IP problems, since the algebraic formulations allow only for the description of mathematical relations, not logical relations. This thesis presents a language $\mathcal{L}^+$ for IP, in which we write logical specification of an IP problem. $\mathcal{L}^+$ is a language based on the predicate logic, but is extended with meta predicates such as at_least(m, S), where m is a non-negative integer, meaning that at least m predicates in the set S of formulas hold. The meta predicates are introduced to facilitate reasoning about a model of an IP problem rigorously and logically. $\mathcal{L}^+$ is executable in the sense that formulas in $\mathcal{L}^+$ are mechanically translated into a set of mathematical formulas, called IP formulas, which most of the existing IP solvers accept. We give a systematic method for translating formulas in $\mathcal{L}^+$ to IP formulas. The translation is rigorously defined, verified and implemented in Mathematica 3.0. By using Mathlink and CGI programming, we develop a Web-based interface to support the system with modelling language, transformation of IP, and IP solvers. This provides a Web-based client-server model, in which the power of high level IP modelling and high performance IP solving can be integrated and developed for a wide range of business users to solve large scale decision making problems. This work follows the approach of McKinnon and Williams, and elaborated the language in that (1) it is rigorously defined, (2) transformation to IP formulas is more optimised and verified, and (3) the transformation is completely given in Mathematica 3.0 and is integrated into IP solving environment as a client-server system for IP.

Minimised geometric Buchberger algorithm for IP Consider the following model of IP problems:

$$IP_{A,C}(b) = \min\{Cx : Ax = b, x \in \mathbb{N}^n\}$$

where C is an n-vector of real numbers, A is an $m \times n$ matrix of integers and b is an m-vector of integers. This model means that we solve variables x under the constraints $Ax = b$ so that the value of Cx is minimal. We use $IP_{A,C}$ to denote a generic IP problem.

IP problems are combinatorial optimisation problems where conventional numerical methods based on the *hill-climbing* technique can not be directly applied. Conventional methods for solving IP are based on searching algorithms where heuristics such as *branch-and-bound* can be applied to reduce the search space.

Recently, the tools of commutative algebra and algebraic geometry have brought new insights to IP via the theory of Gröbner bases. The key idea is to encode an IP problem into a special ideal associated with the constraint matrix A and the cost (objective) function Cx. An important property of such an encoding is that its Gröbner bases correspond directly to the test sets of the IP problem. Thus, by employing an algebraic package such as MACAULAY or MAPLE, the test sets of the IP problem can be directly computed. Using a proper test set (such as the minimal test set which corresponds directly to the reduced Gröbner basis of the encoded ideal), the optimal value of the cost function can be computed by constructing a monotonic path from the initial non-optimal solution of the problem to the optimal solution. Thus, an IP problem can be solved in a similar fashion as in the simplex method for linear programming without using intensive heuristic searching algorithms.

There are two strategies for encoding an IP problem into a special ideal.

(1) Indirect encoding: encoding with adding extra variables.

(2) Direct encoding: encoding without adding extra variables.

The first strategy was first invented by Conti and Traverso. The scheme involves two encoding mechanisms: Encoding the cost function of $IP_{A,C}$ into a linear order and encoding the coefficient matrix into a polynomial ideal. With this translation, IP problems are transformed into the problem of solving the sub-algebra membership problems.

In [84], Thomas proposed a geometric interpretation of Conti-Traverso method. The key idea of Thomas' Geometric Buchberger Algorithm (GBA) is to relate the Gröbner bases of the encoded binomial ideal of an IP to the notion of a test set for the IP. Each binomial is now directly interpreted as a directed line segment, i.e., a vector in a lattice of all feasible solutions of $IP_{A,C}$. The Buchberger algorithm is then directly applied to a directed graph, where nodes of the graph are lattice points corresponding to feasible solutions of $IP_{A,C}$ and the edges at the beginning correspond to the input basis of the binomial ideal. Finding the reduced Gröbner basis amounts to rebuilding the graph such that the edges correspond to the members of the reduced Gröbner basis, which can be geometrically understood as a test set of the IP problem. Thus, by this graph, an optimal solution of $IP_{A,C}$ can be found along the directed path in the graph from a feasible solution. Thomas' work provides not only a succinct understanding of an algebraic IP solver, but also a practical computational procedure for its implementation.

In the above strategy, the first problem is that the strategy is applied to an extended IP (EIP) with additional variables (y) of the form:

$$\min\{My + Cx\}$$

subject to $Iy + Ax = b$ where $(y, x) \in \mathbb{Z}^{m+n}$, I is the $m \times m$ identity matrix and $M \in \mathbb{R}^m$ is a vector whose components have large magnitude (it is assumed, without loss of generality, that all entries in A, C and b are nonnegative integers). In practice the additional variables will lead to a considerable increase in the space and time requirements of the algorithms considered.

The second problem is that the test set generated by both algorithms are generic in the sense that it is only determined by A and C for an IP system $IP_{A,C}$. Thus, the search space for computing the reduced Gröbner basis for such a generalised problem is quite large. In another paper [87], Thomas proposed the truncated Gröbner basis method by fixing b to reduce the cardinality of the reduced Gröbner basis, but the size of Gröbner basis computed by the algorithm is still not optimal since the basis is for the EIP w.r.t a $IP_{A,C}(b)$, not the $IP_{A,C}(b)$ itself. So, many vectors in the reduced Gröbner basis are needed to move from an initial solution of EIP to an initial solution of IP.

The second strategy was established by Hosten and Sturmfels. In [45], Hosten and Sturmfels proposed an algorithm in which a set of fundamental segments of $IP_{A,C}$ can be computed without going through EIP. This algorithm starts with a basis for the lattice $ker(A)$ and then proceeds to refine this to a set of fundamental segments for $IP_{A,C}$. But the algorithm is algebraic and generic. That is, the form of toric ideal I_A is:

$$x^{u^+} - x^{u^-}, u \in B \text{ (lattice basis for } ker(A))$$

The space constructed is not a truncated space since the vector b is not taken into account. Thus, the efficiency is still a problem when the method is applied to large scale IP problems due to the complexity of the Buchberger algorithm.

This thesis proposes a new algebraic algorithm for solving IP. The new algorithm, called the *Minimised Geometric Buchberger Algorithm* (MGBA), combines the Hosten and Sturmfels method (GRIN) and Thomas' truncated GBA to compute the fundamental segments of an IP problem $IP_{A,C}$ directly in its original space and also the truncated Gröbner basis for the fixed b. The new algorithm MGBA was implemented in C on a Sun Enterprise 3000 and experiments were carried out to compare this algorithm with GBA, the truncated GBA, and the GRIN algorithm. The experiments state that MGBA shows significant performance improvement.

Applying MGBA to stochastic IP An IP problem $IP_{A,C}(b)$ with deterministic coefficients c_i, a_{ij} and b_i is of limited use for modelling real world applications. For example, future productivities in a production problem, inflows into

a reservoir connected to a hydro power station, demands at various nodes in a transportation network, and so on, are often appropriately modeled as uncertain parameters, which are at best characterized by probability distributions. So, an IP model is no longer appropriate for describing the mentioned practical problems and we need a model with some uncertain parameters, that is, a stochastic IP model.

Consider the class of stochastic IP called chance constrained IP (CIP) P_0:

Min $h(x)$ subject to
$Prob\{Tx \leq \xi\} \geq \gamma$
$Ax = b$
$x \in \mathbb{N}^n$, ξ is a vector of random variables.

Model P_0 means that we solve variables x under the probabilistic constraint $Prob\{Tx \leq \xi\} \geq \gamma$ and the constraints $Ax = b$ so that the value of $h(x)$ is minimal. Model P_0 has two properties: (1) The probabilistic constraint is not separable and (2) integer variables are required to model setup decisions. The available techniques for stochastic programming do not provide a satisfactory solution for models that have integer variables and nonseparable chance (probabilistic) constraints arising from nonnormal distributions.

As shown before, MGBA provides a new method for solving IP problems. Furthermore, MGBA also provides a new promising solution for model P_0. The idea is to divide the model into two parts: One is composed of the probabilistic constraints and some complicated constraints, called membership oracle; and the other is a simple IP after removing the membership oracle from the problem, called reduced IP. We first compute the test set (reduced Gröbner basis of a toric ideal) for reduced IP by using MGBA. The test set provides a set of directions that can be used to trace paths from every nonoptimal solution to the optimal solution of the reduced IP. So, for any feasible solution of the reduced IP, we get an optimal solution by searching the set of directions. Simultaneously, we can also walk back from the optimal solution to every other feasible solution of the reduced IP by simply reversing these paths. So, with the same test set, we compute the optimal solution of the CIP by walking back from the optimal solution of the reduced IP to other feasible solutions and querying the membership oracle to check whether the reached point is feasible for the CIP. We have proved that the search terminates with either the optimal solution of CIP or all paths are searched, i.e., the CIP is infeasible.

The algorithm for CIP was implemented in C on a Sun Enterprise 3000 and applied to a job scheduling problem. The preliminary experiments indicate that this method provides a realistic solution for the challenge of stochastic IP.

Organisation of the Thesis

This thesis concentrates on modelling IP with logic and solving IP with Gröbner bases. In particular, it presents a modelling language and translation algorithm for the specification of IP problems; it proposes two new algorithms for solving IP problems and stochastic IP problems, respectively. The main contents of the thesis are presented in three chapters: Chapter 3, Chapter 5 and Chapter 6. A detailed summary of each chapter is as follows.

- **Chapter 2: Preliminaries**

  In this chapter, we introduce the basic notations, concepts and theorems which are related to integer programming (IP) and polynomial theory.

- **Chapter 3: Modelling Integer Programming with Logic**

  In this chapter, we present the language $\mathcal{L}^+$ for IP, in which we write logical specifications of IP problems. $\mathcal{L}^+$ is executable in the sense that formulas in $\mathcal{L}^+$ are mechanically translated into a set of mathematical formulas, called IP formulas, which most of the existing IP solvers accept. The translation is rigorously defined, verified and implemented in Mathematica 3.0.

- **Chapter 4: Gröbner Bases for Integer Programming**

  In this chapter, we first introduce the Gröbner basis theory including basic concepts and notations and the Buchberger algorithm for computing Gröbner bases. Then we describe the Gröbner basis technique for IP by introducing a test set for an IP problem.

- **Chapter 5: Minimised Geometric Buchberger Algorithm**

  In this chapter, we propose a new algebraic algorithm for solving IP problems. The new algorithm, called the *Minimised Geometric Buchberger Algorithm* (MGBA). We discuss the implementation of MGBA on a Sun Ultra Enterprise 3000 and conduct an experiment on comparing the new algorithm with other symbolic algorithm for IP such as the algorithm in GRIN, the geometric Buchberger algorithm and the truncated geometric Buchberger algorithm.

- **Chapter 6: Applying MGBA to Stochastic Integer Programming**

  In this chapter, we describe a method for solving a class of stochastic IP problems, the chance constrained IP problem, using the algorithm MGBA. We also apply the new method for chance constrained IP to solve a practical complex IP problem namely job scheduling, and give some experimental results.

- **Chapter 7: Conclusion**

  In this chapter, we draw conclusion on the research work of this thesis and summarise the main contributions of the research. We also give some suggestion for future research work.

Chapter 2

Preliminaries

2.1 Introduction

In this chapter we give the basic notations, concepts and theorems concerning with integer programming (IP) and polynomial theory. The remainder of this chapter is organized as follows. Section 2 introduces the definition of an IP problem, the conversion of a general IP model into the standard IP model and the solution methods for the IP models. In section 3, we introduce polynomial ideals and term orders.

2.2 Integer programming: problem definition

IP problems arise in situations where all decision variables of optimisation problems (with their objectives to be maximised or minimised) must assume integers. For example, we cannot build 1.37 schools, manufacture 11.74 aircraft, or award 10.48 (research, production, etc.) contracts. Many practical optimisation problems can be formulated as IP problems, such as the knapsack problem, the traveling salesman problem, the job scheduling problem, the transportation problem, the assignment problem, the minimum cost flow problem and so on [93][77][82][51]. The standard formulation of IP problems is given as

Minimise Cx subject to $Ax = b$
or more concisely
$\text{Min}\{Cx : \ Ax = b\}$
where $C \in \mathbb{R}^n, A \in \mathbb{Z}^{m \times n}$,
$b \in \mathbb{Z}^m$ and $x \in \mathbb{N}^n$

In the following we summarise the notations related to IP problems, which we will use in the whole thesis:

Cx	:	objective function
Value of Cx	:	objective
$Ax = b$	:	constraints
$\mathbb{R}$	:	the set of real numbers
$\mathbb{R}_+$	:	the set of nonnegative real numbers
$\mathbb{R}^n$	:	the set of n-vectors of real numbers
$\mathbb{Q}$	:	the set of rational numbers
$\mathbb{Z}$	:	the set of integers
$\mathbb{Z}_+$	:	the set of nonnagetive integers
$\mathbb{Z}^m$	:	the set of m-vectors of integers
$\mathbb{Z}^{m\times n}$	:	the set of $m \times n$-matrix of integers
$\mathbb{N}$	:	the set of natural numbers
$\mathbb{N}^n$	:	the set of n-vectors of natural numbers

Solving this IP problem is to find the values of the decision variables x that minimise the values of the objective function Cx while satisfying the constraints $Ax = b$.

IP problems can differ in terms of the objective function (the objective can be either to minimise or maximise a function), as well as in terms of the linear constraints (equality constraints, inequality constraints, smaller or equal, strictly smaller...). The IP problems can usually be converted into this standard form. Here we briefly discuss how to convert a general formulation of an IP problem into the standard formulation by distinguishing the following cases:

1. Conversion of "smaller than" inequalities into equalities

   Suppose that the i-th constraint is the following form:

   $$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

   It is equivalent to say that:

   $$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i - x_{n+1}$$

   or

   $$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + x_{n+1} = b_i$$

   with x_{n+1}, a positive slack variable.

   Now, we consider a new $(n+1) \times 1$ column-vector

   $$x = \begin{pmatrix} x_1 \\ \cdots \\ x_n \\ x_{n+1} \end{pmatrix}$$

   and change A and C as following:

- Add a column to matrix A with zeros everywhere except on the i-th line (where we put a 1, so as to include the x_{n+1} in the i-th constraint).

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{in} & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & 0 \end{pmatrix}$$

- Add a column to vector C (simply one zero, since the new slack variable, x_{n+1}, must not affect the value of the objective function).

$$C = (c_1\ c_2\ \ldots\ c_n\ 0).$$

So we have a new problem, with still m constraints, but $n+1$ variables. If we repeat this process with all the "smaller than" inequalities, and introduce one slack variable for each of them, then we obtain a new problem where all the "smaller than" inequalities are replaced by equalities.

2. Conversion of "bigger than" inequalities into equalities

   Suppose that the i-th constraint is the following form:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i$$

   It is equivalent to say that:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i + x_{n+1}$$

   or

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - x_{n+1} = b_i$$

   with x_{n+1}, a positive slack variable.

   Now, we consider a new $(n+1) \times 1$ column-vector

$$x = \begin{pmatrix} x_1 \\ \cdots \\ x_n \\ x_{n+1} \end{pmatrix}$$

   and change A and C as following:

- Add a column to matrix A with zeros everywhere except on the i-th line (where we put a -1, so as to include the x_{n+1} in the i-th constraint).

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{in} & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & 0 \end{pmatrix}$$

- Add a column to vector C (simply one zero, since the new slack variable, x_{n+1}, must not affect the value of the objective function).

$$C = (c_1, ..., c_n, 0).$$

Like the conversion of "smaller than" inequalities, we obtain a new problem where all the "bigger than" inequalities are replaced by equalities finally.

3. Nonpositive variables

The standard formulation of IP problems is quite restrictive, since it does only consider positive variables. In practice, this does not constitute a problem, since we can easily convert a problem where some variables are not required to be positive into a problem with only positive variables.

Suppose that the i-th variable x_i has no positive constraint. We introduce two "artificial variables" x_i' and x_i'' to replace x_i by

$$x_i = x_i' - x_i''$$

So, we can still have x_i either positive or negative, with x_i' and x_i'' positive. The new variable $(n+1) \times 1$ column-vector is:

$$x = \begin{pmatrix} x_1 \\ \cdots \\ x_i' \\ x_i'' \\ \cdots \\ x_n \\ x_{n+1} \end{pmatrix}$$

Correspondly, we change matrix A and vector C as follows:

- Add a new column in matrix A between columns i and $i+1$, equal to the opposite of column i.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1i} & -a_{1i} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2i} & -a_{2i} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ a_{m1} & \cdots & a_{mi} & -a_{mi} & \cdots & a_{mn} \end{pmatrix}$$

- Add a number c_i' in vector C between c_i and c_{i+1} with $c_i' = -c_i$.

  $C = (c_1 \ldots c_i \ -c_i \ldots c_n)$.

  So, the new problem after conversion of all the variables "nonpositive" into two positive variables can be solved by our algorithm. Once it is solved and we have obtained the optimal value of the objective function, we can find the optimal value of the variables of the original problem by doing the following:

  $x_i = x_i' - x_i''$

4. Conversion of maximisation problems into minimisation problems

   Suppose that a problem is:

   Max $c_1x_1 + c_2x_2 + \cdots + c_nx_n$.

   This is equivalent to:

   Min $-c_1x_1 - c_2x_2 - \cdots - c_nx_n$

   and take the opposite of the optimal value of the objective function for this minimisation problem as the optimal value for the original (maximisation) problem.

   So, what we only need to do is:

   - Take the opposite of vector C (i.e. $(-c_1 \ -c_2 \ldots \ -c_n)$).
   - Solve the standard IP problem "minimise $(-C)x$", subject to the same constraints as the original problem.
   - Take the opposite of the result as the optimal value of the objective function for the original problem, and the same values of the variables $x_1, x_2, \ldots, x_n$ as the optimal solution.

Now, we discuss the solution for the standard formulation of IP problems (simply called IP model afterward). First we introduce the following two notations.

Definition 2.2.1 Vector $x = (x_1, ..., x_n) \in \mathbb{N}^n$ is called a *feasible solution* for the IP model if $Ax = b$. A set $D = \{x : Ax = b\}$ is called a *feasible set* for the IP model.

Definition 2.2.2 Let D be a feasible set for the IP model, if $\exists x_o \in D$, for $\forall x \in D$, we have $Cx \geq Cx_o$, x_o is called an *optimal solution* for the IP model.

The solution for the IP model is classified into one of four cases:

1. $D = \emptyset$, i.e. the IP model has no feasible solution, we call it *infeasible*. For example:

   Min $x_1 + x_2 - x_3$
   subject to
   $x_1 + x_2 + x_3 = 1$,
   $x_1 + x_2 = 5$.

2. D is infinity and the objective is unbounded. That is, the IP model has infinity solutions and we call it *infeasible*. For example:

   Min $x_1 + x_2 - x_3$
   subject to
   $x_1 + x_2 = 5$.

3. The optimal solution x_o is *unique*. For example:

   Min $x_1 - x_2$
   subject to
   $x_1 + x_2 = 5$.

   objective=-5, x_1=0, x_2=5.

4. The optimal solution x_o is *not unique*. For example:

   Min $x_1 + x_2 - x_3$
   subject to
   $x_1 + x_2 + x_3 = 10$,
   $x_3 - x_1 - x_2 = 0$.

   objective=0, x_1=5, x_2=0, x_3=5, or x_1=0, x_2=5, x_3=5.

In case 3 and 4, we call the IP model *feasible.*

IP techniques are generally categorized into two broad types: (1) search methods and (2) cutting methods. The first type is motivated by the fact that the integer solution space can be regarded as consisting of a *finite* number of points. In its simplest form, search methods seek enumerating *all* such points. This would be equivalent to simple exhaustive enumeration. What makes search methods more promising than simple exhaustive enumeration, however, is that techniques can be developed to enumerate only a portion (hopefully small) of all the candidate solutions while *automatically* discarding the remaining points as nonpromising. Clearly, the efficiency of the resulting *search* algorithm depends on the power of the techniques that are developed to discard the nonpromising solution points.

Search methods primarily include implicit enumeration techniques and branch-and-bound techniques. The first type is mostly suited for the $0-1$ problem, and may actually be considered as a special case of the branch-and-bound methods.

Cutting methods are motivated by the fact that the simplex solution to a linear program must occur at an extreme point. The idea then is to add specially developed secondary constraints that are violated by the current noninteger solution but never by any feasible (integer) point. The successive application of such a procedure should eventually result in a new (convex) solution space with its optimum extreme point properly satisfying the integrality condition. The name *cutting* is suggested by the fact that the secondary constraints *cut* off infeasible parts of the continuous solution space.

Another method for the IP problem is inspired by the cutting techniques and calls for identifying the *convex hull* of all the feasible integer points by constructing a set of proper linear inequalities. Once this is done, the application of the regular simplex method clearly produces the desired result.

There are several specialised methods for solving structured problems such as the knapsack problem and the traveling salesman problem. These methods, although designed specifically to exploit the special structure of the problem, are actually in the spirit of either the search methods or the cutting methods (or a combination of both).

A common property that characterises almost all the cutting methods is that no feasible (integer) solution is available until the very end. In other words, there is no available information about the integer solution if the calculations are stopped prematurely. Such methods are usually know as dual techniques and this property represents a major disadvantage. There has been some effort to develop *primal* algorithms (that is, whose integer feasibility is maintained at all times), but from a computational viewpoint, these algorithms have been less satisfactory.

The search methods are sometimes referred to as *almost dual* techniques. Although primal feasibility is not guaranteed at all times, occasionally the algorithms may produce a good feasible solution. This is an important advantage compared to the cutting methods. However, the search methods (especially the

branch-and-bound type) usually result in heavy taxation of computer memory, a difficulty not present in the cutting methods. The reason for this will become evident after each method is explained.

Interestingly enough, the cutting and search methods possess mutually exclusive properties which, if combined, may result in a superior method of solution, namely, an algorithm that yields feasible solutions and in the meantime does not require a large computer memory. This idea is the basis for some interesting developments that will be presented in Chapter 4 and Chapter 5.

2.3 Polynomial theory

2.3.1 Polynomial ideal

Let k be a field and $k[x_1, ..., x_n]$ be the ring of polynomials in n variables with coefficients in k. A polynomial $f(x_1, ..., x_n)$ in n variables $x_1, ..., x_n$ with coefficients in a field k is a finite sum of terms of the form $ax_1^{\alpha_1} \cdots x_n^{\alpha_n}$, where $a \in k$, and all of the exponents $\alpha_i, i \in \{1, ..., n\}$ are nonnegative integers. We call $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ a *monomial* (or *power product*) and $ax_1^{\alpha_1} \cdots x_n^{\alpha_n}$ a *term*. If a polynomial $f(x_1, ..., x_n)$ has two terms, we call it *binomail*. For example, $f(x_1, x_2, x_3) = 3x_1^3x_2^2x_3^5 - 4x_1^3x_2^4 + x_1x_2 + 1$ is a polynomial in three variables and $g(x_1, x_2, x_3, x_4) = x_1^2x_2^3 - x_3^2x_4^3$ is a binomial in four variables, whereas $f(x_1, x_2) = x_1^2x_2^3 - 4x_1x_2^{-2}$ is not a polynomial and not a binomail. We use the vector notation for the monomials in $k[x_1, ..., x_n]$. If $\alpha = (\alpha_1, ..., \alpha_n)$ is a vector in $\mathbb{N}^n$ then $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. Hence the monomials in $k[x_1, ..., x_n]$ are in bijection with the vector in $\mathbb{N}^n$.

We give some concepts for polynomials in one variable. For any polynomial $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 \in k[x]$ with $a_0, a_1, ..., a_n \in k$ and $a_n \neq 0$, the degree of $f(x)$ is n, denoted by $\deg(f(x))=n$. The leading term of $f(x)$ is a_nx^n, denoted by $Lt(f(x))$, and the leading coefficient of $f(x)$ is a_n, denoted by $Lc(f(x))$.

Definition 2.3.1 A subset $I \subseteq k[x_1, ..., x_n]$ is a *polynomial ideal* if it satisfies the following three conditions:

1. $0 \in I$,
2. for all $f, g \in I$, $f + g \in I$,
3. for all $f \in I$ and $h \in k[x_1, ..., x_n]$, $fh \in I$.

Definition 2.3.2 Let $I \subseteq k[x_1, ..., x_n]$ be a polynomial ideal. I is said to be *binomial ideal* if for each $f \in I$, f has only two terms.

Definition 2.3.3 Given a polynomial $f \in k[x_1, ..., x_n]$, let f_k be the sum of all terms of f of total degree k. f_k is said to be the kth *homogeneous component* of f if f_k is homogeneous and $f = \sum_k f_k$.

Definition 2.3.4 An ideal $I \subseteq k[x_1, ..., x_n]$ is said to be *homogeneous* if for each $f \in I$, the homogeneous components f_i of f are in I as well.

Definition 2.3.5 Let $f_1, ..., f_s$ be polynomials in $k[x_1, ..., x_n]$, and let

$$\langle f_1, ..., f_s \rangle = \{\sum_{i=1}^{s} h_i f_i : h_1, ..., h_s \in k[x_1, ..., x_n]\},$$

we call $\langle f_1, ..., f_s \rangle$ the *ideal generated by* $f_1, ..., f_s$.

The ideal $\langle f_1, ..., f_s \rangle$ has a nice interpretation in terms of polynomial equations. Given $f_1, ..., f_s \in k[x_1, ..., x_n]$, we get the system of equations

$$f_1 = 0, \ \cdots, \ f_s = 0.$$

From these equations, one can derive others using algebra. For example, if we multiply the first equation by $h_1 \in k[x_1, ..., x_n]$, the second by $h_2 \in k[x_1, ..., x_n]$, etc., and add the resulting equations, we obtain

$$h_1 f_1 + h_2 f_2 + \cdots + h_s f_s = 0,$$

which is a consequence of our original system. Notice that the left-hand side of this equation is exactly an element of the ideal $\langle f_1, ..., f_s \rangle$. Thus, we can think of $\langle f_1, ..., f_s \rangle$ as consisting of all *polynomial consequences* of the equations $f_1 = f_2 = \cdots = f_s = 0$.

We say that an ideal I is *finitely generated* if there exist $f_1, ..., f_s \in k[x_1, ..., x_n]$ such that $I = \langle f_1, ..., f_s \rangle$, and we say that $f_1, ..., f_s$ are a *basis* of I. In Chapter 4, we prove that *every* ideal of $k[x_1, ..., x_n]$ is finitely generated (this is known as the Hilbert Basis Theorem).

2.3.2 Term orders

Term orders are important in the multi-variable division algorithm for polynomials and the computation of Gröbner bases. Here, we introduce some term orders.

Definition 2.3.6 Let $\succ$ be a total order on $\mathbb{N}^n$. We call $\succ$ a *term order* on $\mathbb{N}^n$ if it satisfies the following properties:

(i) $\forall \alpha, \beta, \gamma \in \mathbb{N}^n, \alpha \succ \beta \Longrightarrow \alpha + \gamma \succ \beta + \gamma$,

(ii) $\forall \alpha \in \mathbb{N}^n \backslash \{0\}, \alpha \succ 0$.

A term order on $\mathbb{N}^n$ gives a term order on the monomials of $k[x_1, ..., x_n]$ in the obvious way. For example, $X^\alpha \succ X^\beta$ if $\alpha \succ \beta$, where $X^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, $X^\beta = x_1^{\beta_1} \cdots x_n^{\beta_n}$, $\alpha = (\alpha_1, ..., \alpha_n)$ and $\beta = (\beta_1, ..., \beta_n)$.

A wide variety of term orders are available. Here we discuss the following four term orders that are often used.

Definition 2.3.7 (Lexicographic Order $\succ_{lex}$)
Let $\alpha, \beta \in \mathbb{N}^n$, $\alpha \succ_{lex} \beta$ if the first nonzero element of $\alpha - \beta$ from the left is positive. Obviously we have $X^\alpha \succ_{lex} X^\beta$ if $\alpha \succ_{lex} \beta$. We call this order *lex*.

Here are some examples:
(a) (1,2,0)$\succ_{lex}$(0,5,3) since $\alpha - \beta$=(1,-3,-3).
(b) (3,2,4)$\succ_{lex}$(3,2,3) since $\alpha - \beta$=(0,0,1).
(c) The variables $x_1, ..., x_n$ are ordered in the usual way by the lex ordering:

$$(1, 0, ..., 0) \succ_{lex} (0, 1, ..., 0) \succ_{lex} \cdots \succ_{lex} (0, ..., 0, 1)$$

$x_i = x_1^0 \cdots x_i^1 \cdots x_n^0$ for $i = 1, ..., n$, so $x_1 \succ_{lex} x_2 \succ_{lex} \cdots \succ_{lex} x_n$.

Definition 2.3.8 (Reverse Lexicographic Order $\succ_{revlex}$) Let $\alpha, \beta \in \mathbb{N}^n$, we say $\alpha \succ_{revlex} \beta$ if the first nonzero element of $\alpha - \beta$ from the right is negative. Obviously we have $X^\alpha \succ_{revlex} X^\beta$ if $\alpha \succ_{revlex} \beta$. We call this order *revlex*.

Definition 2.3.9 (Graded Lexicographic Order $\succ_{grlex}$)
Let $\alpha, \beta \in \mathbb{N}^n$, we say $\alpha \succ_{grlex} \beta$ if

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and } \alpha \succ_{lex} \beta.$$

We call this order *grlex*.

We see that grlex orders by total degree first, then breaks ties using lex order. Here are some examples:
(a) (1,2,5)$\succ_{grlex}$(3,2,0) since |(1,2,5)|=8 > |(3,2,0)|=5.
(b) (1,2,3)$\succ_{grlex}$(0,3,3) since |(1,2,3)|=|(0,3,3)| and (1,2,3)$\succ_{lex}$(0,3,3).
(c) We have $x_1 \succ_{revlex} x_2 \succ_{revlex} \cdots \succ_{revlex} x_n$, since

$$(1, 0, ..., 0) \succ_{grlex} (0, 1, ..., 0) \succ_{grlex} \cdots \succ_{grlex} (0, ..., 0, 1).$$

Definition 2.3.10 (Graded Reverse Lexicographic Order $\succ_{grevlex}$)
Let $\alpha, \beta \in N^n$, we say $\alpha \succ_{grevlex} \beta$ if

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and } \alpha \succ_{revlex} \beta.$$

We call this order *grvlex*.

Like grlex, grevlex orders by total degree first, then breaks ties in a different way. Here are some examples:
(a) (1,2,5)$\succ_{grevlex}$(3,2,0) since |(1,2,5)|=8 > |(3,2,0)|=5.

(b) (1,2,3)$\succ_{grevlex}$(0,0,6) since |(1,2,3)|=|(0,0,6)| and (1,2,3)$\succ_{revlex}$(0,3,3).
(c) We still have $x_1 \succ_{grevlex} x_2 \succ_{grevlex} \cdots \succ_{grevlex} x_n$.

From now on, when we fix a term order $\succ$ on $k[x_1, ..., x_n]$ and choose a nonzero $f \in k[x_1, ..., x_n]$, we will often write f with $X^{\alpha_1} \succ X^{\alpha_2} \succ \cdots \succ X^{\alpha_m}$ as

$$f = a_1 X^{\alpha_1} + a_2 X^{\alpha_2} + \cdots + a_m X^{\alpha_m}$$

where $a_i \in k, a_i \neq 0$ for all $i \in \{1, ..., m\}$. We call a_1, X^{α_1} and $a_1 X^{\alpha_1}$ leading coefficient $lc(f)$, leading power product $lp(f)$ and leading term $lt(f)$ of f respectively. For example, let $f = 4x_1x_2^2x_3 + 4x_3^2 - 5x_1^3 + 7x_2^2x_3^2$ and term order be $\succ_{lex}$. Then $lc(f) = -5$, $lp(f) = x_1^3$ and $lt(f) = -5x_1^3$.

2.3.3 Multi-variable division algorithm

In this subsection, we introduce a division algorithm for polynomials in $k[x_1, ..., x_n]$, which is called multi-variable division algorithm. The goal is to divide $f \in k[x_1, ..., x_n]$ by nonzero polynomials $f_1, ..., f_m \in k[x_1, ..., x_n]$ such that

$$f = u_1 f_1 + \cdots + u_m f_m + r$$

where $u_i \in k[x_1, ..., x_n]$ for all $i \in \{1, ..., m\}$ and $r \in k[x_1, ..., x_n]$ with $lp(r) \leq lp(f)$. The basic idea behind the algorithm is the same as for linear and one variable polynomials: when dividing f by $f_1, ..., f_m$, we want to cancel terms of f using the leading terms of the f_i's (so the new terms which are introduced are smaller than the canceled terms) and continue this process until we cannot do anymore. First we look at the special case of the division of f by g, where $f, g \in k[x_1, ..., x_n]$.

Definition 2.3.11 Given a term order on $k[x_1, ..., x_n]$, $f, g, h \in k[x_1, ..., x_n]$ with $g \neq 0$, we say that f is reduced to h modulo g, denoted by $f \xrightarrow{g} h$, if and only if there exists a nonzero term X in f which is divisible by $lp(g)$ and $h = f - \frac{X}{lt(g)}g$.

Note that in this definition the term X in f is replaced by terms strictly smaller than X. For example, let $f = x^2y + 3xy^4 + 2y^2 - 1$ and $g = 2xy^3 + 4xy + 1$ be polynomials in $\mathbb{Q}[x, y]$. If the term order is lex with $x > y$, then we have $lt(g) = 2xy^3$, $X = 3xy^4$ and $f \xrightarrow{g} h$ with $h = x^2y - 6xy^2 + 2y^2 - \frac{3}{2}y - 1$.

Definition 2.3.12 Given a term order on $k[x_1, ..., x_n]$, let $f, f_1, ..., f_m, h \in k[x_1, ..., x_n]$ with $f_i \neq 0 (1 \leq i \leq m)$, we say that f is reduced to h modulo $F = \{f_1, ..., f_m\}$, denoted by $f \xrightarrow{F} h$, if and only if there exist a set $\{i_1, ..., i_t\} \subseteq \{1, ..., m\}$ and a set of polynomials $\{h_{i_1}, ..., h_{i_{t-1}}\} \subseteq k[x_1, ..., x_n]$ such that

$$f \xrightarrow{f_{i_1}} h_{i_1} \xrightarrow{f_{i_2}} \cdots \xrightarrow{f_{i_{t-1}}} h_{i_{t-1}} \xrightarrow{f_{i_t}} h.$$

For example, let $f = y^2x, f_1 = yx - y, f_2 = y^2 - x \in \mathbb{Q}[x, y]$. Let $F = \{f_1, f_2\}$ and the term order be grlex with $y > x$. Then we have

$$f \xrightarrow{f_1} y^2 \xrightarrow{f_2} x.$$

Thus, $f \xrightarrow{F} x$.

Definition 2.3.13 A polynomial $r \in k[x_1, ..., x_n]$ is irreducible modulo a set of nonzero polynomials $F = \{f_1, ..., f_m\} \subseteq k[x_1, ..., x_n]$, if r cannot be reduced modulo F.

We also say that r is irreducible if either $r = 0$ or there exists no term in r which is divisible by any one of the $lp(f_i), i = 1, ..., m$. If $f \xrightarrow{F} r$ and r is irreducible, then we call r the reminder of f with respect to F and write as $f \xrightarrow{F}_* r$.

Now we can describe the multi-variable division (MVD) algorithm as follows.

Algorithm 2.3.1 The MVD Algorithm
Input: $f_1, ..., f_m, f$
Output: $u_1, ..., u_m, r$
Set $u_1 = 0, u_2 = 0, ..., u_m = 0, r = 0, h = f$
While $h \neq 0$ **Do**
  IF $\exists i \wedge lp(f_i)$ divides $lp(h)$ **Then**
    choose i least such that $lp(f_i)$ divides $lp(h)$
    $u_i = u_i + \frac{lt(h)}{lt(f_i)}$
    $h = h - \frac{lt(h)}{lt(f_i)} f_i$
  Else
    $r = r + lt(h)$
    $h = h - lt(h)$

Let us illustrate the algorithm by an example. In this example, let $f = x^3y^2 + x^2y^4 + 3xy + 1, f_1 = x + y^2 - 1, f_2 = y^6 + 1 \in \mathbb{Q}[x, y]$ and $F = \{f_1, f_2\}$. We use lex as term order with $x > y$. By above algorithm, we finally get $u_1 = x^2y^2 + xy^2 - y^4 + y^2 + 3y, u_1 = 1$ and $r = -2y^4 - 3y^3 + y^2 + 3y$ such that $f = u_1f_1 + u_2f_2 + r$ with $lp(r) < lp(f) = x^3y^2, lp(f_1)lp(u_1) = lp(f)$, and $lp(f_2)lp(u_2) < lp(f)$.

Chapter 3

Modelling Integer Programming with Logic

3.1 Introduction

Quantitative modelling is now recognized as the key technology underlying decision support in financial, commercial and industrial organizations. However, despite important advances, the practical adoption of this technology has been limited. We identify two major deficiencies that have led to this. Firstly the limitation of the classical approach has been one of modelling. Both the specification and analysis of the solution of real world problems are highly complex. Hence, non-specialist potential users, such as managers, are discouraged from formulating the problem or interpreting the results. Secondly, progress has been limited by the lack of effective algorithms to address some of these problems and the absence of good high performance computing environment to compute solutions at reasonably speed.

We have been working on parallel methods for IP and stochastic IP for decision support systems [56][55][54]. In order to exploit the power of these new algorithms in real world applications, a declarative modelling system becomes necessary, that enables us to specify complex decision support procedures. Logically specified decision problems will be transformed into admissible constraints which will then be solved using constraint solvers on a parallel computer.

In this Chapter, we describe a modelling language for IP based on the predicate calculus and present a systematic algorithm for transforming logic formulas into IP formulas. We discuss the implementation for this algorithm. The rest of the chapter is organized as follows. Section 2 contains the brief review of approaches for logic modelling IP. In section 3, we describe a language for modeling IP in the framework of the first-order logic. In section 4, we present an algorithm for transforming logical formulas into IP formulas. In section 5 we discuss the implementation of the algorithm in Mathematica.

3.2 Review of logical modelling frameworks for IP

Since the early 50's many discrete optimization techniques have been developed and have made a significant contribution to solving decision problems using a quantitative (model) representation. During the 80's some logic-based modelling frameworks for IP have been proposed. These frameworks can be divided into two classes: constructing IP models by the predicate calculus and constructing IP models by the propositional calculus. The first class may be represented by the work of McKinnon & Williams [62]. The second may be represented by the works of Hadjiconstantinou & Mitra[40] and Raman & Grossmann[69].

Hadjiconstantinou & Mitra propose a modelling tool which employs propositional calculus as a basis for formalizing the modelling of IP. They transform the tree-represented logical formulas directly into IP forms.

Raman & Grossmann propose a modelling framework based on propositional calculus and a specialized algorithm for discrete linear programming problems. Their approach is to convert the logical formulas in the propositional calculus model into the conjunctive normal form and then to obtain the equivalent IP form.

Mckinnon & Williams propose a logical modelling framework based on the predicate calculus and a series of transformation rules, which automatically converts all the input predicates into an intermediate form called ge-form, from which an IP model can be generated.

The predicate calculus offers a more flexible modelling tool than the propositional calculus. In the predicate calculus one can declare predicates (n-ary relations) between objects. But it is difficult to transform the predicate into IP forms directly. By introducing ge-form, Mckinnon & Willams transform the predicate into ge-forms first and then transform ge-forms into IP forms. Mckinnon & Willams' method based on the predicate calculus has, thus, the obvious advantages over the previous other works. However, their method lacks rigorous definition of modelling language and systematic transformation algorithm. In this paper, we extend McKinnon & Williams' idea and propose a modelling language $\mathcal{L}^+$ based on the predicate logic and present rules for transforming formulas of the language into IP formulas with rigorous proofs of the soundness of the transformation. Our language for modelling IP is presented in the next section.

3.3 Modelling language $\mathcal{L}^+$

In this section we describe a modelling language $\mathcal{L}^+$ for describing IP problems. Our aim is to dispense with the commonly used algebraic model by a logical

model in problem specification. An algebraic model is an optimisation-oriented analytical model used in mathematical programming. Real world problems are described in mathematical terms. The main components of the formulation are decision variables, constraints and objective functions. The solver tries to find values of the decision variables that maximise or minimise the values of the objective functions while satisfying the constraints.

We assume the language $\mathcal{L}$ of the first-order predicate logic in this paper. Our modelling language $\mathcal{L}^+$ extends $\mathcal{L}$ by introducing meta predicates at_least(m, S), at_most(m, S), and none(S) where m is an integer and S is a set of $\mathcal{L}^+$ formulas. The relation at_least(m, S) (at_most(m, S)) expresses that at least (at most) m formulas in S must hold, while none(S) expresses that no formula in S holds.

By the use of $\mathcal{L}^+$, we can describe IP problems more naturally and logically, yet keeping track of the algebraic properties of the original IP problems.

Now, we define the language $\mathcal{L}^+$. $\mathcal{L}^+$ is a language of quantifier-free formulas built over the following alphabet.

(a) improper symbols: (,), {, } and logical connectives $\vee, \wedge, \rightarrow, \leftrightarrow, \neg$

(b) integer variables: $d, x, y, z, \ldots$

(c) propositional variables: $p, q, r, \ldots$

(d) integer constants: ...,-2,-1,0,1,2,...

(e) function symbols: $+, -, \times$

(f) predicate symbols: $=, \geq, \leq, >, <$

The function symbol $-$ is used as both unary and binary operators. In the former case it is prefix, in the latter it is infix. All the other function and predicate symbols are binary and infix. Function symbols are used to form arithmetic terms, and predicate symbols to form constraints over arithmetic terms.

Following are some examples of arithmetic predicates. Let x, y, z be integer variables. Expressions $2x$, $3y$, $4z$, x, 10, $6z$, $2x + 3y - 4z$, $x - y$, $10 + 6z$ are arithmetic terms. For brevity we omit $\times$ in $2x$, $3y$, $4z$, and $6z$. Expressions $2x + 3y - 4z \geq 0$ and $x - y = 10 + 6z$ are constraints specified by arithmetic relations between the arithmetic terms $2x + 3y - 4z$ and 0 in the former, and $x - y$ and $10 + 6z$ in the latter.

In addition to these standard arithmetic terms, we will use, as a meta notation, a summation of arithmetic terms $\sum_i a_i x_i$, where a_i is an integer coefficient of integer variable x_i. Constraints and propositional variables are combined to form logical formulas. The formation rules for well formed logical formulas (hereafter we simply call formulas) are as follows:

1. A propositional variable or a constraint is an atomic formula.

2. An atomic formula is a formula.
3. $P_1 \Xi P_2$ is a formula if P_1 and P_2 are formulas, and Ξ is one of the logical connectives $\vee$, $\wedge$, $\rightarrow$ and $\leftrightarrow$.
4. $\neg P$ is a formula if P is a formula.
5. at_least(m, S) is a formula if $m(\geq 1)$ is an integer and S is a set of formulas.
6. at_most(m, S) is a formula if $m(\geq 1)$ is an integer and S is a set of formulas.
7. none(S) is a formula if S is a set of formulas.

A set is represented a set of formulas in the usual way. For example, $\{P_1, ..., P_n\}$ is a set of formulas if $P_1, ..., P_n$ are formulas.

The first four definition rules (1.to 4.) are standard definitions of propositional formulas (see, e.g. standard text book[30]), while the last three rules (5. to 7.) are newly introduced. As we see in Rule $\mathcal{T}$ in section 4.1, at_most(m, S) and none(S) are expressed by at_least(m, S), hence they are unnecessary theoretically. They are provided as a primitive for convenience in describing IP problems. We assume the standard precedence rules for logical connectives and the use of parentheses for changing the precedence. For example,

$$\neg p \vee x \leq 2 \leftrightarrow y \leq 8 \wedge z \geq 7$$

is understood as

$$((\neg p) \vee (x \leq 2)) \leftrightarrow ((y \leq 8) \wedge (z \geq 7))$$

Example 3.3.1 If there are at most 2 analysts or at most 3 administrators in a research group for a project then there must be at most 8 programmers or at least 7 engineers and vice versa.

Let integer variables x_1, x_2, x_3 and x_4 represent the number of analysts, administrators, programmers and engineers, respectively. We model the above problem in $\mathcal{L}^+$:

$$x_1 \leq 2 \vee x_2 \leq 3 \leftrightarrow x_3 \leq 8 \vee x_4 \geq 7$$

3.4 Algorithm for transformation on $\mathcal{L}^+$

A specification of an IP-problem expressed as an $\mathcal{L}^+$-formulas, although rigorous, will not be very helpful from computational point of view unless the specification can be executed in some way and it delivers a solution of the IP-problem. There would be a method for directly executing $\mathcal{L}^+$-formulas, but this is not what we aim for since we know already that in the domain of our interest we have

efficient methods for solving IP-problems. Therefore we transform $\mathcal{L}^+$-formulas to expressions that IP-solvers will accept, i.e. a system of linear constraints, which we call formulas of IP-form, or simply IP-formulas. In this section, we present an algorithm that transforms $\mathcal{L}^+$-formulas to IP-formulas.

Our algorithm will use an intermediate form to be called Γ-form like ge-form in the method proposed by Mckinnon & Williams. A formula of Γ-form or Γ-formula in short, is either an atomic $\mathcal{L}^+$-formula or an expression of the form $\Gamma_m S$, meaning that m or more formulas in S are true. An IP-formula is a set of constraints $\{C_1, \ldots, C_n\}$, where each constraint C_i is a linear equality or a linear inequality involving with integer variables and constants.

Any $\mathcal{L}^+$-formula can be transformed to a Γ-formula since we have the following logical equivalence:

$$\Gamma_1 \{P_1, \ldots P_n\} \equiv P_1 \vee \cdots \vee P_n \tag{3.4.1}$$

and

$$\Gamma_n \{P_1, \ldots, P_n\} \equiv P_1 \wedge \cdots \wedge P_n \tag{3.4.2}$$

and any $\mathcal{L}^+$-formulas are transformed to a logically equivalent normal forms of either a DNF (disjunctive normal form) or a CNF (conjunctive normal form). Here we say that P and Q are logically equivalent, written as $P \equiv Q$, iff for any model and valuation the value of P and Q are the same.

With this observation one would conclude that CNF or DNF would be used instead of Γ-form. However, this would lead to formulations of large size, resulting in an unnecessarily large set of IP-formulas.

In summary our approach for modelling IP-problems consists in the following steps:

1. to describe an IP-problem in $\mathcal{L}^+$-formulas,
2. to transform $\mathcal{L}^+$-formulas to Γ-formulas,
3. to transform Γ-formulas to IP-formulas, and
4. solving IP-formulas by a specialised IP-solver.

3.4.1 Transformation of $\mathcal{L}^+$-formulas into Γ-formulas

First we introduce an equivalence rule $\mathcal{E}$. By this rule, all $\mathcal{L}^+$-subformulas of the form of the left-hand side of the rules are eliminated. Then resulting $\mathcal{L}^+$-formulas are transformed by rule $\mathcal{T}$ to Γ-formulas.

Equivalence rule $\mathcal{E}$:

Let P, Q be $\mathcal{L}^+$-formulas.

$(\mathcal{E}_1)$: $\mathcal{E}[\neg\neg P] = \mathcal{E}[P]$
$(\mathcal{E}_2)$: $\mathcal{E}[P \rightarrow Q] = \mathcal{E}[\neg P] \vee \mathcal{E}[Q]$
$(\mathcal{E}_3)$: $\mathcal{E}[P \leftrightarrow Q] = \mathcal{E}[P \rightarrow Q] \wedge \mathcal{E}[Q \rightarrow P)$
$(\mathcal{E}_4)$: $\mathcal{E}[P \vee Q] = \mathcal{E}[P] \vee \mathcal{E}[Q]$
$(\mathcal{E}_5)$: $\mathcal{E}[P \wedge Q] = \mathcal{E}[P] \wedge \mathcal{E}[Q]$
$(\mathcal{E}_6)$: $\mathcal{E}[A] = A$ if A is an atomic formula

Rule $\mathcal{T}$: transformation of $\mathcal{L}^+$-formula into Γ-formula

Let P, Q be $\mathcal{L}^+$-formulas, S be a set of $\mathcal{L}^+$-formulas, $\overline{S} = \{\neg P | P \in S\}$, and $|S|$ be the cardinality of S.

$(\mathcal{T}_1)$: $\mathcal{T}[P \vee Q] = \Gamma_1\{\mathcal{T}[P], \mathcal{T}[Q]\}$
$(\mathcal{T}_2)$: $\mathcal{T}[P \wedge Q] = \Gamma_2\{\mathcal{T}[P], \mathcal{T}[Q]\}$
$(\mathcal{T}_3)$: $\mathcal{T}[\text{at\_least}(m, S)] = \Gamma_m \mathcal{T}'[S]$
$(\mathcal{T}_4)$: $\mathcal{T}[\text{at\_most}(m, S)] = \Gamma_{|S|-m} \mathcal{T}'[\overline{S}]$
$(\mathcal{T}_5)$: $\mathcal{T}[\text{none}(S)] = \Gamma_{|S|} \mathcal{T}'[\overline{S}]$
$(\mathcal{T}_6)$: $\mathcal{T}[\neg P] = \neg \mathcal{T}[P]$
$(\mathcal{T}_7)$: $\mathcal{T}[A] = A$ if A is an atomic formula
where $\mathcal{T}'[\{P_1, \ldots, P_n\}] = \{\mathcal{T}[P_1], \ldots, \mathcal{T}[P_n]\}$

After applying $\mathcal{T}$-rule, we eliminate negated Γ-formulas by the following rule.

Rule $\mathcal{N}$: elimination of negated Γ-formulas

$(\mathcal{N}_1)$: $\mathcal{N}[\neg \Gamma_m S] = \Gamma_{|S|-m+1} \mathcal{N}'[\overline{S}]$
$(\mathcal{N}_2)$: $\mathcal{N}[\neg(\sum_{j=1}^n a_j x_j \Theta b)] = \sum_{j=1}^n a_j x_j \overline{\Theta} b$
$(\mathcal{N}_3)$: $\mathcal{N}[\neg(\sum_{j=1}^n a_j x_j = b)] =$
$\Gamma_1\{\sum_{j=1}^n a_j x_j < b, \sum_{j=1}^n a_j x_j > b\}$
$(\mathcal{N}_4)$: $\mathcal{N}[\neg p] = \neg p$ if p is a propositional variable
$(\mathcal{N}_5)$: $\mathcal{N}[\Gamma_m S] = \Gamma_m \mathcal{N}'[S]$
$(\mathcal{N}_6)$: $\mathcal{N}[A] = A$ if A is an atomic formula

where $\mathcal{N}'[\{P_1, ..., P_n\}] = \{\mathcal{N}[P_1], ..., \mathcal{N}[P_n]\}$,
Θ is one of $\leq, \geq, <, >$, and
$\overline{\Theta}$ is one of $>, <, \geq, \leq$, respectively.

Repeated application of the following flattening rules may further simplify nested Γ-formulas.

Rule $\mathcal{F}$: Flattening of Γ-formulas

$(\mathcal{F}_1)$: $\mathcal{F}[\Gamma_1(\{\Gamma_1 S_1\} \cup S_2)] = \Gamma_1(S_1 \cup S_2)$
$(\mathcal{F}_2)$: $\mathcal{F}[\Gamma_{|S_2|+1}(\{\Gamma_{|S_1|} S_1\} \cup S_2)] = \Gamma_{|S_1 \cup S_2|}(S_1 \cup S_2)$

Before we proceed further, we make sure that the above rules $\mathcal{E}$, $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{F}$ are correct. Obviously rule $\mathcal{E}$ is correct since it is a standard logical equivalence. We consider transformation rule $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{F}$. We rely on certain algebraic properties that hold for domains of integers and operators that are applied to integer values. Hence, it suffices to define the correctness of the transformations in a specialised domain.

Let $\mathcal{M}$ be a model whose domain of interpretation is the set of integers. Arithmetic operators used in the language of $\mathcal{L}^+$ are interpreted as usual; e.g. $+$ is an addition of integers. Then we define the correctness of the transformation as follows.

Definition 3.4.1 We call a transformation rule $\mathcal{H}$ $\mathcal{M}$-correct if

$$\mathcal{M}[\![\mathcal{H}[P]]\!]_\phi = \mathcal{M}[\![P]\!]_\phi$$

for any valuation ϕ associated with $\mathcal{M}$.

By (3.4.1) and (3.4.2), it is easy to see that the transformation rule $\mathcal{T}$ is $\mathcal{M}$-correct for any model $\mathcal{M}$. The flattening rule $\mathcal{F}$ is a consequence of the associativity of connective $\vee$. So, $\mathcal{F}$ is $\mathcal{M}$-correct for any model $\mathcal{M}$. It is easy to prove rule $\mathcal{N}$ is $\mathcal{M}$-correct by analysis of the values of $\mathcal{M}[\![\mathcal{N}[P]]\!]_\phi$ and $\mathcal{M}[\![P]\!]_\phi$.

By the transformation rules $\mathcal{E}$, $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{F}$, we can translate all $\mathcal{L}^+$-formulas into Γ-formulas.

Example 3.4.1 We translate the following $\mathcal{L}^+$-formula into a Γ-formula.

$$p_1 \wedge p_2 \rightarrow p_3 \vee p_4$$

We first translate the above formula into

$$\neg(p_1 \wedge p_2) \vee (p_3 \vee p_4)$$

by rule $(\mathcal{E}_2)$.
By rule $(\mathcal{T}_1)$, we get a Γ-form:

$$\Gamma_1\{\neg\, \Gamma_2\{p_1, p_2\}, \Gamma_1\{p_3, p_4\}\}.$$

We eliminate negated Γ-formulas by $(\mathcal{N}_1)$:

$$\Gamma_1\{\Gamma_1\{\neg\, p_1, \neg\, p_2\}, \Gamma_1\{p_3, p_4\}\}.$$

Finally, we flatten the Γ-formula by $(\mathcal{F}_1)$:

$$\Gamma_1\{\neg\, p_1, \neg\, p_2, p_3, p_4\}.$$

Example 3.4.2 We transform the $\mathcal{L}^+$-formulas in Example 3.3.1 into a Γ-formula.

$x_1 \leq 2 \vee x_2 \leq 3 \leftrightarrow x_3 \leq 8 \vee x_4 \geq 7$

By rule $(\mathcal{E}_3)$, we translate the above formula into the following formula:

$$(\neg(x_1 \leq 2 \vee x_2 \leq 3) \vee (x_3 \leq 8 \vee x_4 \geq 7)) \wedge$$
$$((\neg(x_3 \leq 8 \vee x_4 \geq) \vee (x_1 \leq 2 \vee x_2 \leq 3))$$

By $(\mathcal{T}_1)$ and $(\mathcal{T}_2)$ we get a Γ-formula:

$$\Gamma_2\{\Gamma_1\{\neg\, \Gamma_1\{x_1 \leq 2, x_2 \leq 3\}, \Gamma_1\{x_3 \leq 8, x_4 \geq 7\}\},$$
$$\Gamma_1\{\neg\, \Gamma_1\{x_3 \leq 8, x_4 \geq 7\}, \Gamma_1\{x_1 \leq 2, x_2 \leq 3\}\}\}.$$

We apply rule $(\mathcal{N}_1)$ to eliminate negated Γ-formulas.

$$\Gamma_2\{\Gamma_1\{\Gamma_2\{x_1 > 2, x_2 > 3\}, \Gamma_1\{x_3 \leq 8, x_4 \geq 7\}\},$$
$$\Gamma_1\{\Gamma_2\{x_3 > 8, x_4 < 7\}, \Gamma_1\{x_1 \leq 2, x_2 \leq 3\}\}\}.$$

Then by $(\mathcal{F}_1)$ we obtain the following Γ-formula:

$$\Gamma_2\{\Gamma_1\{\Gamma_2\{x_1 > 2, x_2 > 3\}, x_3 \leq 8, x_4 \geq 7\},$$
$$\Gamma_1\{\Gamma_2\{x_3 > 8, x_4 < 7\}, x_1 \leq 2, x_2 \leq 3\}\}.$$

3.4.2 Transformation of Γ-formulas into IP-formulas

We give the transformation rule $\mathcal{R}$ of Γ-formulas into IP-formulas in Figure 3.1. The key idea of the transformation is that (1) we introduce decision variables that take the values of either 1 (implying true) or 0 (implying false) and that (2) propositional variables are treated in the same way as in decision variables. Suppose that p denotes a propositional variable that represents a proposition describing an action, option or decision. With a propositional variable p, we associate a decision variable δ_p with the property:

$$\delta_p = 1 \quad \text{iff} \quad \mathcal{M}[\![p]\!]_\phi = \mathrm{T}, \text{ and}$$
$$\delta_p = 0 \quad \text{iff} \quad \mathcal{M}[\![p]\!]_\phi = \mathrm{F}$$

for any valuation ϕ.

By introducing new variable d_i $(i = 1, ..., n)$, we translate a Γ-formula into a conditional Γ-formula of the form $d_i \rightarrow P$, which then is translated into an IP-formula.

In order to translate a conditional Γ-formula that contains a constraint of the form $\sum_{j=1}^{n} a_j x_j \Theta b$, where Θ is one of the predicate symbols $\geq$, $\leq$, $>$ and $<$, into an IP-formula, we use the upper and lower bounds U and L of $\sum_{j=1}^{n} a_j x_j - b$ for given ranges of $x_j, j = 1, ..., n$. With L and U, we translate the conditional constraints into IP-formulas (see $\mathcal{R}_{8.1}$, $\mathcal{R}_{8.2}$, $\mathcal{R}_{8.3}$ and $\mathcal{R}_{8.4}$). For a conditional Γ-formula that contains the constraint $\sum_{j=1}^{n} a_j x_j = b$, we replace the constraint $\sum_{j=1}^{n} a_j x_j = b$ with an equivalent form $\sum_{j=1}^{n} a_j x_j \geq b \wedge \sum_{j=1}^{n} a_j x_j \leq b$ and translate it into two conditional Γ-formulas (see $\mathcal{R}_{8.5}$).

After the above transformation, we compute the bounds U and L for each $\sum_{j=1}^{n} a_j x_j - b$. These bounds are computed using the bounds of $x_j, j = 1, ..., n$

$$
\begin{aligned}
(\mathcal{R}_1): &\quad \mathcal{R}[p] = \delta_p \geq 1 \\
(\mathcal{R}_2): &\quad \mathcal{R}[\neg p] = \delta_p \leq 0 \\
(\mathcal{R}_3): &\quad \mathcal{R}[A] = A \text{ if } A \text{ is a constraint} \\
(\mathcal{R}_4): &\quad \mathcal{R}[\Gamma_m\{P_1,...,P_n\}] = \exists\bar{d}\{d_1+\cdots+d_n \geq m, \\
&\qquad\qquad \mathcal{R}[d_1>0\rightarrow P_1],...,\mathcal{R}[d_n>0\rightarrow P_n]\} \\
(\mathcal{R}_5): &\quad \mathcal{R}[d>0\rightarrow p] = \delta_p \geq d \\
(\mathcal{R}_6): &\quad \mathcal{R}[d>0\rightarrow \neg p] = 1-\delta_p \geq d \\
(\mathcal{R}_7): &\quad \mathcal{R}[d>0\rightarrow \Gamma_m\{P_1,...,P_n\}] = \exists\bar{d}\{d_1+\cdots+d_n \geq md, \\
&\qquad\qquad \mathcal{R}[d_1>0\rightarrow P_1],...,\mathcal{R}[d_n>0\rightarrow P_n]\} \\
(\mathcal{R}_{8.1}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j \leq b] = \sum_{j=1}^n a_jx_j - b \leq U(1-d) \\
(\mathcal{R}_{8.2}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j < b] = \sum_{j=1}^n a_jx_j - b < (U+1)(1-d) \\
(\mathcal{R}_{8.3}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j \geq b] = \sum_{j=1}^n a_jx_j - b \geq L(1-d) \\
(\mathcal{R}_{8.4}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j > b] = \sum_{j=1}^n a_jx_j - b > (L-1)(1-d) \\
(\mathcal{R}_{8.5}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j = b] = \{\mathcal{R}[d>0\rightarrow \sum_{j=1}^n a_jx_j \geq b], \\
&\qquad\qquad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j \leq b]\}
\end{aligned}
$$

where $\bar{d}$ is an abbreviation of $d_1, ..., d_n$.

Figure 3.1: Rule $\mathcal{R}$

that are usually specified separately or inferred from characteristics of the problems. We compute U and L as follows.

Let $l_j \leq x_j \leq u_j$ $(j = 1, ..., n)$,

$$U = \sum_{i\in\alpha} a_iu_i + \sum_{i\in\beta} a_il_i - b,$$

$$L = \sum_{i\in\alpha} a_il_i + \sum_{i\in\beta} a_iu_i - b,$$

where $\alpha = \{i \mid a_i > 0\}$, $\beta = \{i \mid a_i < 0\}$ and
$\alpha \cup \beta = \{1, ..., n\}$.

Bounds l_j and u_j for x_j, $j = 1, ..., n$, are natural numbers and u_j can be allowed to be infinite in this computation. With the computation, we eliminate some redundant IP-formulas by checking the value of U or L. For example, consider the following rule:

$$
\begin{aligned}
(\mathcal{R}_{8.1}): &\quad \mathcal{R}[d>0\rightarrow \textstyle\sum_{j=1}^n a_jx_j \leq b] = \\
&\qquad \textstyle\sum_{j=1}^n a_jx_j - b \leq U(1-d).
\end{aligned}
$$

The idea behind $(\mathcal{R}_{8.1})$ is that we use

$$\sum_{j=1}^{n} a_j x_j - b \leq U$$

as an expression whose value is always true.
If $\neg(U > 0)$,

$$\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j - b \leq 0]\!]_\phi = \mathrm{T}$$

for any valuation ϕ. Hence, IP formula $\sum_{j=1}^{n} a_j x_j - b \leq U(1-d)$ is redundant in this case and will not be generated. Otherwise we generate this formula.

Some IP-solvers such as CPLEX [21] assume all constraint boundaries are closed. In such a case, we remove strict inequalities $>$ and $<$ before giving the IP-formulas to these IP-solvers. It can be done easily by converting $>$ and $<$ into $\geq$ and $\leq$ respectively using the following equivalence:

$\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j > b]\!]_\phi = \mathcal{M}[\![\sum_{j=1}^{n} a_j x_j \geq b+1]\!]_\phi$, and
$\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j < b]\!]_\phi = \mathcal{M}[\![\sum_{j=1}^{n} a_j x_j \leq b-1]\!]_\phi$

for any valuation ϕ.

$(\mathcal{R}_9)$: $\mathcal{R}[\Gamma_n\{P_1,...,P_n\}] = \{\mathcal{R}[P_1],...,\mathcal{R}[P_n]\}$
$(\mathcal{R}_{10})$: $\mathcal{R}[d>0 \rightarrow \Gamma_n\{P_1,...,P_n\}] = \{\mathcal{R}[d>0 \rightarrow P_1],...,\mathcal{R}[d>0 \rightarrow P_n]\}$
$(\mathcal{R}_{11})$: $\mathcal{R}[\Gamma_m\{P_1,...,P_r,p_1,...,p_s,\neg p_{s+1},...,\neg p_t\}] =$
$\exists d\{\mathcal{R}[d_1>0 \rightarrow P_1],...,\mathcal{R}[d_r>0 \rightarrow P_r],$
$\sum_{i=1}^{r} d_i + \sum_{i=1}^{s} \delta_{p_i} + \sum_{i=s+1}^{t}(1-\delta_{p_i}) \geq m\}$
$(\mathcal{R}_{12})$: $\mathcal{R}[d>0 \rightarrow \Gamma_m\{P_1,...,P_r,p_1,...,p_s,\neg p_{s+1},...,\neg p_t\})] =$
$\exists d\{\mathcal{R}[d_1>0 \rightarrow P_1],...,\mathcal{R}[d_r>0 \rightarrow P_r],$
$\sum_{i=1}^{r} d_i + \sum_{i=1}^{s} \delta_{p_i} + \sum_{i=s+1}^{t}(1-\delta_{p_i}) \geq md\}$
where P_i $(i=1,...,r)$ is a Γ-formula and p_i $(i=1,...,t)$ is a propositional variable.

Figure 3.2: Optimising Rule $\mathcal{R}_o$

The output of rule $\mathcal{R}$ is a set of IP-formulas. A set of IP-formulas $\{C_1,\ldots,C_n\}$ represents a conjunction of formulas $C_1 \wedge \cdots \wedge C_n$. Due to the associativity of $\wedge$, we can flatten a set of formulas. For example, we can flatten a nested set $\{q,\{C_1,\ldots,C_n\},\{D_1,\ldots,D_m\}\}$ to a logically equivalent set $\{q,C_1,\ldots,C_n,D_1,\ldots,D_m\}$. This allows us to use nested sets in Figure 3.1 to represent IP-formulas. We will, however, assume that nested sets of IP-formulas are flattened implicitly whenever necessary.

The output of rule $\mathcal{R}$ may be an existentially quantified IP-formula. The quantification is dropped in IP-formulas. The original logical specification is implicitly existentially quantified formulas, and hence resulting IP-formulas are implicitly existentially quantified. Note, however, all the transformation rules are applicable to universally quantified formulas.

We have the following soundness result of rule $\mathcal{R}$.

Theorem 3.4.1 Rule $\mathcal{R}$ is $\mathcal{M}$-correct.

Proof: We prove $\mathcal{M}[\![\mathcal{R}[P]]\!] = \mathcal{M}[\![P]\!]$ by induction on the number of Γ symbols occurring in a Γ-formula P. We prove rules $(\mathcal{R}_5) \sim (\mathcal{R}_{8.1})$. Rules $(\mathcal{R}_1)$, $(\mathcal{R}_2)$ and $(\mathcal{R}_4)$ can be proved as special cases of rules $(\mathcal{R}_5) \sim (\mathcal{R}_7)$, respectively. Rules $(\mathcal{R}_{8.2}) \sim (\mathcal{R}_{8.5})$ are proved similarly to rule $(\mathcal{R}_{8.1})$.

Let P be a formula of the form $d > 0 \rightarrow Q$. We distinguish the following cases depending on the structure of Q.

(1) Q is an equation of the form p or $\neg p$. We take an arbitrary but fixed valuation ϕ. By easy case analysis of the values of $\mathcal{M}[\![d > 0]\!]_\phi$ and $\mathcal{M}[\![p]\!]_\phi$ with definition of δ_p, we have

$$\mathcal{M}[\![d > 0 \rightarrow p]\!]_\phi = \mathcal{M}[\![\delta_p \geq d]\!]_\phi$$
$$\mathcal{M}[\![d > 0 \rightarrow \neg p]\!]_\phi = \mathcal{M}[\![1 - \delta_p \geq d]\!]_\phi.$$

(2) Q is a Γ-formula.
We prove that

$$\mathcal{M}[\![d > 0 \rightarrow \Gamma_m\{P_1, \ldots, P_n\}]\!]_\phi = \mathcal{M}[\![\exists \bar{d}\{d_1 + \ldots + d_n \geq md, \mathcal{R}[d_1 > 0 \rightarrow P_1], \ldots, \mathcal{R}[d_n > 0 \rightarrow P_n]\}]\!]_\phi \quad (A.1)$$

for an arbitrary but fixed valuation ϕ.

We further distinguish two cases depending on the values of $\phi(d)$. Since d is a decision variable, d is assigned the value of either 1 or 0. Suppose $\phi(d) = 0$. Then $\mathcal{M}[\![d > 0]\!]_\phi = \mathrm{F}$, it is easy to see that by taking $\phi(d_1) = 0, ..., \phi(d_n) = 0$, both side of Eq.(A.1) reduce to T.

Next consider the case that $\phi(d) = 1$. Let $S_1, \ldots, S_k$ be all the m-subsets of $\{P_1, \ldots, P_n\}$, where $k =_n C_m$. Suppose that the left-hand side of Eq. (A.1)= T, *i.e.* $\mathcal{M}[\![d > 0 \rightarrow S_1 \vee \ldots \vee S_k]\!]_\phi = \mathrm{T}$. This implies that there exists a set $S_j = \{P_{j_1}, \ldots, P_{j_m}\} \in \{S_1, \ldots, S_k\}$ such that $\mathcal{M}[\![P_{j_1}]\!]_\phi = \mathrm{T}, \ldots, \mathcal{M}[\![P_{j_m}]\!]_\phi = \mathrm{T}$. We define a new valuation ϕ' that modifies ϕ as follows.

$$\phi'(x) = \phi(x), \quad \text{if } x \notin \{d_1, \ldots, d_n\}$$
$$\phi'(d_i) = 1, \quad \text{if } i \in \{j_1, \ldots j_m\}$$
$$\phi'(d_i) = 0, \quad \text{if } i \in \{1, \ldots, n\} - \{j_1, \ldots j_m\}$$

We have to prove

$$\mathcal{M}[\![d > 0 \rightarrow \Gamma_m \{P_1, \ldots, P_n\}]\!]_{\phi'} =$$
$$\mathcal{M}[\![\{d_1 + \ldots + d_n \geq md,$$
$$\mathcal{R}[d_1 > 0 \rightarrow P_1], \ldots, \mathcal{R}[d_n > 0 \rightarrow P_n]\}]\!]_{\phi'}$$

With this valuation ϕ', we have

$$\mathcal{M}[\![d_1 + \ldots + d_n \geq md]\!]_{\phi'} = \mathrm{T}, \text{since } \phi'(d) = 1$$
$$\mathcal{M}[\![d_i > 0 \rightarrow P_i]\!]_{\phi'} = \mathrm{T}, \text{ for } i \in \{j_1, \ldots j_m\}$$
$$\mathcal{M}[\![d_i > 0 \rightarrow P_i]\!]_{\phi'} = \mathrm{T},$$
$$\text{for } i \in \{1, \ldots, n\} - \{j_1, \ldots j_m\}.$$

By induction hypothesis, we have

$$\mathcal{M}[\![\mathcal{R}[d_i > 0 \rightarrow P_i]]\!]_\rho = \mathcal{M}[\![d_i > 0 \rightarrow P_i]\!]_\rho,$$

for arbitrary (but associated with $\mathcal{M}$) valuation ρ.
In particular,

$$\mathcal{M}[\![\mathcal{R}[d_i > 0 \rightarrow P_i]]\!]_{\phi'} = \mathcal{M}[\![d_i > 0 \rightarrow P_i]\!]_{\phi'} = \mathrm{T}.$$

Hence, Eq. (A.1) holds.

Suppose next that the left-hand side of Eq.(A.1) is F, *i.e.* $\mathcal{M}[\![d > 0 \rightarrow S_1 \vee \ldots \vee S_k]\!]_\phi = \mathrm{F}$. Then for all S_j $(j = 1, ..., k)$, $\mathcal{M}[\![S_j]\!]_\phi = \mathrm{F}$. This implies that there exists $i \in \{1, \ldots, n\}$ such that $\mathcal{M}[\![P_i]\!]_\phi = \mathrm{F}$. For this case, we define a valuation ϕ' as follows:

$$\phi'(d_i) = 1$$
$$\phi'(x) = \phi(x), \text{ otherwise}$$

Then $\mathcal{M}[\![d_i > 0 \rightarrow P_i]\!]_{\phi'} = \mathrm{F}$. The same reasoning is applied to this case as in the case of T and we have $\mathcal{M}[\![\mathcal{R}[d_i > 0 \rightarrow P_i]]\!]_{\phi'} = \mathrm{F}$. Hence, the right-hand side of Eq.(A.1) is F.

(3) Q is a constraint.
We only prove that

$$\mathcal{M}[\![d > 0 \rightarrow \textstyle\sum_{j=1}^{n} a_j x_j \leq b]\!]_\phi =$$
$$\mathcal{M}[\![\textstyle\sum_{j=1}^{n} a_j x_j - b \leq U(1 - d)]\!]_\phi \qquad (A.2)$$

We distinguish two cases depending on the value assignment of d.

Case $\phi(d) = 0$.
The left-hand side of Eq. (A.2) is equal T. The right-hand side is equal to $\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j - b \leq U]\!]_\phi$, and hence is equal to T by definition of U.

Case $\phi(d) = 1$.
If $U > 0$, both sides of Eq.(A.2) are equal to $\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j \leq b]\!]_\phi$. If $U \leq 0$, the right-hand side of Eq.$(A.2)$ is equal to $\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j - b \leq U]\!]_\phi$, and the left-hand side of Eq.$(A.2)$ is equal to $\mathcal{M}[\![\sum_{j=1}^{n} a_j x_j - b \leq 0]\!]_\phi$. Since U is the upper bound of $\sum_{j=1}^{n} a_j x_j - b$, both are equal to T.

□

We give the optimising rules of $\mathcal{R}$ in Figure 3.2. It is easy to prove that the optimising rules are correct in the same way as in the proof of Theorem 3.4.1. By the rules $\mathcal{R}$ and $\mathcal{R}_o$, we can translate all Γ-formulas into IP-formulas.

Example 3.4.3 We transform the Γ-formula in Example 3.4.1 to an IP-formula.

(1) $\quad \Gamma_1\{\neg p_1, \neg p_2, p_3, p_4\}$

Applying $(\mathcal{R}_{11})$ to (1), we obtain:
$(1 - \delta_{p_1}) + (1 - \delta_{p_2}) + \delta_{p_3} + \delta_{p_4} \geq 1$, that is, $\delta_{p_1} + \delta_{p_2} - \delta_{p_3} - \delta_{p_4} \leq 1$

Example 3.4.4 We transform the Γ-formula in Example 3.4.2 to IP-formulas.

$\Gamma_2\{\Gamma_1\{\Gamma_2\{x_1 > 2, x_2 > 3\}, x_3 \leq 8, x_4 \geq 7\},$
$\quad \Gamma_1\{\Gamma_2\{x_3 > 8, x_4 < 7\}, x_1 \leq 2, x_2 \leq 3\}\}$

By $(\mathcal{R}_9)$, we have
(1) $\Gamma_1\{\Gamma_2\{x_1 > 2, x_2 > 3\},\ x_3 \leq 8,\ x_4 \geq 7\}$,
(2) $\Gamma_1\{\Gamma_2\{x_3 > 8, x_4 < 7\},\ x_1 \leq 2,\ x_2 \leq 3\}$.
$(\mathcal{R}_4)$ transforms (1) to give
(1.1) $d_1 > 0 \rightarrow \Gamma_2\{x_1 > 2, x_2 > 3\}$,
(1.2) $d_2 > 0 \rightarrow x_3 \leq 8,\ d_3 > 0 \rightarrow x_4 \geq 7$,
(1.3) $d_1 + d_2 + d_3 \geq 1$.
$(\mathcal{R}_{10})$ transforms (1.1) to give
(1.1.1) $d_1 > 0 \rightarrow x_1 > 2,\ d_1 > 0 \rightarrow x_2 > 3$.
Let U_{x-b} and L_{x-b} represent the upper bound of $x - b$ and the lower bound of $x - b$ respectively. By $(\mathcal{R}_{8.4})$ we transform (1.1.1) into the following IP-formulas:

(1.1.1.1) $\{x_1 - 2 > (L_{x_1-2} - 1)(1 - d_1) : L_{x_1-2} \leq 0\}$,
$\{x_2 - 3 > (L_{x_2-3} - 1)(1 - d_1) : L_{x_2-3} \leq 0\}$.

$(\mathcal{R}_{8.1})$ and $(\mathcal{R}_{8.3})$ transform (1.2) to give

(1.2.1) $\{x_3 - 8 \leq U_{x_3-8}(1-d_2) : U_{x_3-8} > 0\}$,
$\{x_4 - 7 \geq L_{x_4-7}(1-d_3) : L_{x_4-7} < 0\}$.

($\mathcal{R}_4$) transforms (2) to give
(2.1) $d_4 > 0 \rightarrow \Gamma_2\{x_3 > 8, x_4 < 7\}$,
(2.2) $d_5 > 0 \rightarrow x_1 \leq 2$, $d_6 > 0 \rightarrow x_2 \leq 3$,
(2.3) $d_4 + d_5 + d_6 \geq 1$.
($\mathcal{R}_{10}$) transforms (2.1) to give
(2.1.1) $d_4 > 0 \rightarrow x_3 > 8$, $d_4 > 0 \rightarrow x_4 < 7$.
($\mathcal{R}_{8.4}$) and ($\mathcal{R}_{8.2}$) transform (2.1.1) to give

(2.1.1.1) $\{x_3 - 8 > (L_{x_3-8} - 1)(1-d_4) : L_{x_3-8} \leq 0\}$,
$\{x_4 - 7 < (U_{x_4-7} + 1)(1-d_4) : U_{x_4-7} \geq 0\}$.

Two applications of ($\mathcal{R}_{8.1}$) transform (2.2) to give

(2.2.1) $\{x_1 - 2 \leq U_{x_1-1}(1-d_5) : U_{x_1-1} > 0\}$,
$\{x_2 - 3 \leq U_{x_2-3}(1-d_6) : U_{x_2-3} > 0\}$.

Let $1 \leq x_i \leq 10$ $(i = 1,..,4)$, we compute U_{x_i-b} and L_{x_i-b} and get the following IP-formulas:

$x_1 - 2d_1 > 0$,
$x_2 - 3d_1 > 0$,
$x_3 + 2d_2 \leq 10$,
$x_4 - 6d_3 \geq 1$,
$d_1 + d_2 + d_3 \geq 1$,
$x_3 - 8d_4 > 0$,
$x_4 + 4d_4 < 11$,
$x_1 + 8d_5 \leq 10$,
$x_2 + 7d_6 \leq 10$,
$d_4 + d_5 + d_6 \geq 1$.

3.5 Implementation in Mathematica

We have implemented the transformation discussed in this paper in Mathematica 3.0. The implemented system, to be called TIP (Transformation of Integer Programming) is used as a front-end to existing solvers. By using Mathlink and CGI, we connect TIP to (possibly remote) IP-solvers such as CPLEX [21] or solvers that we have developed [56][55][54] via Web page. When TIP is called in Web page, with $\mathcal{L}^+$-formulas, TIP will generate IP-formulas, which then are sent to the solvers together with an objective function via Mathlink and CGI. The answer, an optimal solution for the IP problem, is sent back to the Web

page. TIP consists of the functions GenIP[spec_,decl_,opts___], RRC[spec_], RuleE[spec_], RuleT[spec_], RuleN[spec_], RuleF[spec_] and RuleR[spec_]. RRC[spec_] (Remove Redundant Constraints) checks whether a given constraint is redundant and outputs empty set {} if redundant or else a simplified constraint after substituting the value of the bounds. RuleE[spec_] to RuleR[spec_] define the transformation rules $\mathcal{E}$ to $\mathcal{R}$ respectively. GenIP[spec_,decl_,opts___] is the main function which is called with a logical specification spec, a list of propositional variables and integer variables with their bounds decl and an option opts. By option opts, users specify whether the generated IP-formulas will be simplified or not with IPFormSimplify→True (default) or IPFormSimplify→False. TIP first checks each variable in the logic specification spec whether it is declared in decl and outputs an error message if some variables are not declared or else outputs the generated IP-formulas. We give the whole program of TIP in the appendix.

Example 3.5.1 We buy nine types of stocks numbered by 1 to 9. If three or more types of stocks {1 to 5} are bought, or less than four types of stocks {3 to 6,8,9} are bought then at most two types of stocks {6 to 9} will be bought unless none of stocks {5 to 7} are bought.
Question: Which stocks can we buy to get as many types as possible?

We can formulate the above problem in $\mathcal{L}^+$ by using propositional variable s_i for $i = 1, ..., 9$ to represent i-th type of the stock. $s_i = \mathrm{T}$ means we buy the i-th type of the stock and $s_i = \mathrm{F}$ means we do not buy the i-th type of the stock. The constraint of the problem can be formulated as follows.

$$\begin{aligned}&(\text{at\_least}(3, \{s_1, s_2, s_3, s_4, s_5\}) \vee \\ &\ \text{at\_most}(3, \{s_3, s_4, s_5, s_6, s_8, s_9\})) \wedge \\ &\ \neg\ \text{none}(s_5, s_6, s_7) \rightarrow \text{at\_most}(2, \{s_6, s_7, s_8, s_9\})\end{aligned}$$

We transform the $\mathcal{L}^+$-formula into the IP-formulas by TIP.

$$\begin{aligned}&\texttt{GenIP[(atleast[3,}\{s_1, s_2, s_3, s_4, s_5\}\texttt{]}\vee \\ &\qquad \texttt{atmost[3,}\{s_3, s_4, s_5, s_6, s_8, s_9\}\texttt{])}\wedge \\ &\qquad \neg\ \texttt{none[}\{s_5, s_6, s_7\}\texttt{]} \\ &\qquad \rightarrow\ \texttt{atmost[2,}\{s_6, s_7, s_8, s_9\}\texttt{]}, \\ &\qquad \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}\texttt{]}\end{aligned}$$

The IP-formulas generated are:

$$\begin{aligned}&d_1 + d_2 + d_3 \geq 1, \\ &1 - \delta_{s_5} \geq d_2, \\ &1 - \delta_{s_6} \geq d_2, \\ &1 - \delta_{s_7} \geq d_2,\end{aligned}$$

$$4 - \delta_{s_6} - \delta_{s_7} - \delta_{s_8} - \delta_{s_9} \geq 2d_1,$$
$$5 - \delta_{s_1} - \delta_{s_2} - \delta_{s_3} - \delta_{s_4} - \delta_{s_5} \geq 3d_3,$$
$$\delta_{s_3} + \delta_{s_4} + \delta_{s_5} + \delta_{s_6} + \delta_{s_8} + \delta_{s_9} \geq 4d_3,$$
$$0 \leq d_1 \leq 1, 0 \leq d_2 \leq 1, 0 \leq d_3 \leq 1,$$
$$0 \leq \delta_{s_1} \leq 1, 0 \leq \delta_{s_2} \leq 1, 0 \leq \delta_{s_3} \leq 1,$$
$$0 \leq \delta_{s_4} \leq 1, 0 \leq \delta_{s_5} \leq 1, 0 \leq \delta_{s_6} \leq 1,$$
$$0 \leq \delta_{s_7} \leq 1, 0 \leq \delta_{s_8} \leq 1, 0 \leq \delta_{s_9} \leq 1.$$

Then we send the above IP-formulas and an objective function `Max` $\sum_{i=1}^{9} \delta_{s_i}$ to an IP-solver and finally get the optimal solution:

objective=7,
$\delta_{s_1} = 1, \delta_{s_2} = 1, \delta_{s_3} = 1, \delta_{s_4} = 1, \delta_{s_5} = 1, \delta_{s_6} = 1,$
$\delta_{s_7} = 1, \delta_{s_8} = 0, \delta_{s_9} = 0, d_1 = 1, d_2 = 0, d_3 = 0.$

Chapter 4

Gröbner bases for Integer Programming

4.1 Introduction

In this chapter, we first introduce the Gröbner basis theory including basic concepts, notation and the famous Buchberger algorithm for computing Gröbner bases. Then we describe the Gröbner basis technique for IP. We also give a rigorous proof of the equivalence between the test set of the IP problem and the reduced Gröbner basis of the encoded toric ideal from this IP problem.

4.2 Gröbner bases

Before we introduce Gröbner bases, let us see the Hilbert Basis Theorem.

Theorem 4.2.1 (Hilbert Basis Theorem) In the ring $k[x_1, ..., x_n]$, we have the following:

(i) For any ideal I of $k[x_1, ..., x_n]$, there exist polynomials $f_1, ..., f_s \in k[x_1, ..., x_n]$ such that $I = \langle f_1, ..., f_s \rangle$.
(ii) If $I_1 \subseteq I_2 \subseteq \cdots \subseteq I_n \subseteq \cdots$ is an ascending chain of ideals of $k[x_1, ..., x_n]$, then there exists N such that $I_N = I_{N+1} = I_{N+2} = \cdots$.

An ideal I in any ring satisfying (i) of the above theorem is said to be finitely generated or to have a finite set of generators. Any commutative ring satisfying (ii) is called a Noetherian ring. It turns out that if either of the two conditions in above theorem holds, then the other also holds.

For every ideal of $k[x_1, ..., x_n]$, there exist different generating sets. Among them, the best one is the Gröbner basis [14][20][1]. Here we introduce the concept of Gröbner basis and discuss some of its properties.

Definition 4.2.1 Given a term order, a finite subset of nonzero polynomials $G = \{g_1, ..., g_m\}$ of an ideal I of $k[x_1, ..., x_n]$ is said to be a *Gröbner basis* for I if and only if for every nonzero polynomial $f \in I$, there exist $i \in \{1, ..., m\}$ such that $lp(g_i)$ divides $lp(f)$.

From the definition we can see that an ideal I of $k[x_1, ..., x_n]$ may have a different Gröbner basis if we use a different order. We discuss some properties of Gröbner bases as follows.

Lemma 4.2.1 Given a nonzero ideal I of $k[x_1, ..., x_n]$, a set of nonzero polynomials $G = \{g_1, ..., g_m\}$ in I is a Gröbner basis for I if and only if $f \xrightarrow{G}_* 0$ for every nonzero $f \in I$.

It follows immediately from the above lemma that $G = \{g_1, ..., g_m\}$ is a Gröbner basis for I if and only if for every polynomial $f \in I, f = \sum_{i=1}^{m} u_i g_i$ with $lp(f) = \max\{lp(u_1)lp(g_1), ..., lp(u_m)lp(f_m)\}$. Moreover, it is easy to verify that if $G = \{g_1, ..., g_m\}$ is a Gröbner basis for I, then $I = \langle g_1, ..., g_m \rangle$.

Before discussing the other properties of Gröbner bases, we introduce the following definition. For a subset I in $k[x_1, ..., x_n]$, $Lt(I)$ denotes the ideal generated by leading terms of elements of I, that is

$$Lt(I) = \langle lt(f) : f \in I \rangle.$$

We also call $Lt(I)$ the *leading term ideal* of I. Note that in general $Lt(I) \neq I$. Now we have the following lemma.

Lemma 4.2.2 Given a nonzero ideal $I \in k[x_1, ..., x_n]$, a set of nonzero polynomials $G = \{g_1, ..., g_m\}$ is a Gröbner basis for I if and only if $Lt(I) = Lt(G)$.

By Lemma 4.2.1 and Lemma 4.2.2, we have the following theorem.

Theorem 4.2.2 There exists a Gröbner basis for every ideal I in $k[x_1, ..., x_n]$.

In the next section we introduce an algorithm for finding a Gröbner basis of an ideal I of $k[x_1, ..., x_n]$.

4.3 Buchberger algorithm

Before describing the Buchberger algorithm, we first introduce a basic concept which will be used in the algorithm.

Definition 4.3.1 The S-polynomial of nonzero polynomials $f, g \in k[x_1, ..., x_n]$ is

$$SP(f, g) = \text{LCM}(lp(f), lp(g))\left(\frac{f}{lt(f)} - \frac{g}{lt(g)}\right)$$

where $\text{LCM}(lp(f), lp(g))$ is the least common multiple of $lp(f)$ and $lp(f)$. In other words, if $lp(f) = X^{\alpha}$ and $lp(g) = X^{\beta}$, then $\text{LCM}(lp(f), lp(g)) = X^{\gamma}$ with $\gamma = (\gamma_1, ..., \gamma_n)$ and $\gamma_i = \max\{\alpha_i, \beta_i\}$ for each $i \in \{1, ..., n\}$.

For example, let $f = 2x^3y^2+xy^3+1$ and $g = xy^3+3xy+4$ be in $\mathbb{Q}[x,y]$. We use lex with $x > y$. Then we have $lt(f) = 2x^3y^2$, $lt(g) = xy^3$, $\mathrm{LCM}(lp(f), lp(g)) = x^3y^3$, and

$$SP(f,g) = x^3y^3(\frac{2x^3y^2+xy^3+1}{2x^3y^2} - \frac{xy^3+3xy+4}{xy^3}) = -3x^3y - 4x^2 + \frac{1}{2}xy^4 + \frac{1}{2}y$$

Note that the $SP(f,g)$ is in fact designed to cancel leading terms. We have the following lemma and theorem.

Lemma 4.3.1 Let $f_1, .., f_m \in k[x_1, ..., x_n]$ be given such that $lp(f_i) = X^\alpha \neq 0$ for all $i \in \{1, ..., m\}$. If $f = \sum_{i=1}^m c_i f_i$ with $c_i \in k, i = 1, ..., m$ and $lp(f) < X^\alpha$, then f is a linear combination of $SP(f_i, f_j), 1 \leq i < j \leq m$ with coefficients in k. Moreover, $lp(SP(f_i, f_j)) < X^\alpha, 1 \leq i < j \leq m$.

Theorem 4.3.1 (Buchberger Theorem)
Let $G = \{g_1, ..., g_m\}$ be a set of nonzero polynomials in $k[x_1, ..., x_n]$. Then G is a Gröbner basis for the ideal $I = \langle g_1, ..., g_m \rangle$ if and only if $SP(g_i, g_j) \xrightarrow{G}_* 0, 1 \leq i < j \leq m$.

This theorem gives an easy criterion for checking whether a set of nonzero polynomials is a Gröbner basis or not. We give an example. Let $f_1 = x + y^2 - 1$ and $f_2 = y^6 + 1$ be in $\mathbb{Q}[x,y]$. Let $F = \{f_1, f_2\}$. We use lex with $x > y$. Then we have that $SP(f_1, f_2) = -x + y^8 - y^6$. Moreover,

$$SP(f_1, f_2) \xrightarrow{f_1} y^8 - y^6 + y^2 - 1 \xrightarrow{f_2} -y^6 - 1 \xrightarrow{f_2} 0.$$

Hence F is a Gröbner basis for $I = \langle f_1, f_2 \rangle$.

Now we introduce the Buchberger algorithm as follows.

Algorithm 4.3.1 Buchberger Algorithm
Input: $F = \{f_1, ..., f_m\} \subseteq k[x_1, ..., x_n], f_i \neq 0, 1 \leq i \leq m$.
Output: $G = \{g_1, ..., g_t\}$, a Gröbner basis for $\langle f_1, ..., f_m \rangle$
Set G=F, $\mathcal{G} = \{\{f_i, f_j\} : f_i \neq f_j \in G\}$
While $\mathcal{G} \neq \emptyset$ **Do**
  Choose any $\{p, q\} \in \mathcal{G}$
  $SP(p,q) \xrightarrow{G}_* h$
  $\mathcal{G} = \mathcal{G} \backslash \{p, q\}$
  IF $h \neq 0$ **Then**
    $\mathcal{G} = \mathcal{G} \cup \{\{f, h\} :$ for all $f \in G\}$
    $G = G \cup \{h\}$

We demonstrate the algorithm by an example. Let $f_1 = x^2y+1$ and $f_2 = y^2+1$ be in $\mathbb{Q}[x, y]$. Let $F = \{f_1, f_2\}$. We use grlex with $x > y$. We first show that F is not a Gröbner basis for $I = \langle f_1, f_2\rangle$. Take for example $f = -x^2 + y$ from I. Note that $f = yf_1 - x^2f_2$. Since $lp(f) = x^2, lp(f_1) = x^2y$, and $lp(f_2) = y^2$, it is clear that $lp(f)$ is not divisible by $lp(f_1)$ or $lp(f_2)$. By above algorithm, we get $G = \{g_1, g_2, g_3\}$ with $g_1 = f_1, g_2 = f_2$ and $g_3 = f$ which is a Gröbner basis for $I = \langle f_1, f_2\rangle$.

Theorem 4.3.2 Given $F = \{f_1, ..., f_m\}$ with $f_i \neq 0, 1 \leq i \leq m$, Buchberger algorithm (Algorithm 4.3.1) will terminate with a Gröbner basis for the ideal $I = \langle f_1, ..., f_m\rangle$.

From the Buchberger algorithm we can find that for the same ideal I there exist several Gröbner bases. In order to get a unique Gröbner basis for any ideal, we need to slightly modify the definition of Gröbner bases.

Definition 4.3.2 A Gröbner basis G for an ideal I is a minimal Gröbner basis for I if for every $g \in G$, $lc(g) = 1$ and $lp(g)$ is not divisible by $lp(p)$ of any $g \neq p \in G$.

Definition 4.3.3 A Gröbner basis G for an ideal I is a reduced Gröbner basis for I if for every $g \in G$, $lc(g) = 1$ and g is a reminder of g with respect to $G\backslash\{g\}$.

This definition says that G is a reduced Gröbner basis if the leading coefficient of every element g of G is 1 and no nonzero term of every element g of G is divisible by the leading product of any other elements of G. The following theorem states that every nonzero ideal has a unique reduced Gröbner basis.

Theorem 4.3.3 Let I be a nonzero ideal of $k[x_1, ..., x_n]$. Then for a given term order, I has a unique reduced Gröbner basis.

Consider the previous example where $f_1 = x^2y + 1$ and $f_2 = y^2 + 1$ are in $\mathbb{Q}[x, y]$. We use grlex with $x > y$. We have shown that $G = \{g_1, g_2, g_3\}$ with $g_1 = f_1, g_2 = f_2, g_3 = -x^2 + y$ is a Gröbner basis for $I = \langle f_1, f_2\rangle$. But G is not a reduced Gröbner basis since $lp(g_1) = x^2y$ is divisible by $lp(g_3) = x^2$. However, we can easily see that $H = \{g_2, g_3\}$ is a reduced Gröbner basis for I.

4.4 Gröbner bases as test sets in IP

Test set for IP was first introduced by Conti et.al[18] in algebra and by Thomas[84] in geometry. Using the test set, the optimal value of the cost function can be computed by constructing a monotone path from the initial non-optimal solution of the problem to the optimal solution. By the theory of Gröbner bases[14],

the test set corresponds directly to the reduced Gröbner basis of a special ideal associated with the constraint matrix A and the cost (objective) function cx. So, the test set can be obtained by computing the reduced Gröbner basis of the special ideal. In this section, we overview the properties of the test set and give a rigorous proof of equivalence between a minimal test set and a reduced Gröbner basis.

4.4.1 A test set for IP problems

Consider IP problems of the form

$$\text{Min}\{cx:\ Ax = b,\ x \in \mathbb{N}^n\}$$

where $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{R}^n$. Let $IP_{A,c}$ denote the family of IP problems of the above form obtained by varying the right-hand side vector b, but keeping A and c fixed. Let $IP_{A,c}(b)$ denote the specific member of the family $IP_{A,c}$ with fixed right hand side vector b. We assume that each $IP_{A,c}(b)$ is bounded with respect to the cost function cx. In the rest of the thesis, we will use this form for IP problems if we have no special statement.

Consider the map $\pi : \mathbb{N}^n \to \mathbb{Z}^m$ such as $\pi(x) = Ax$. Given a vector $b \in \mathbb{Z}^m$, the set of feasible solutions to $IP_{A,c}(b)$ constitute $\pi^{-1}(b)$, the pre-image of b under this map. We identify $\pi^{-1}(b)$ with its convex hull and call it the *b-fiber* of IP. Therefore the b-fiber of $IP_{A,c}$ is just the convex hull of all feasible solutions to $IP_{A,c}(b)$.

Each point in $\mathbb{N}^n$ is a feasible solution in some fiber of $IP_{A,c}$, i.e. α in $\mathbb{N}^n$ is a feasible solution in the $A\alpha$-fiber of $IP_{A,c}$. Now we encode the cost function cx of $IP_{A,c}$ into a linear order on $\mathbb{N}^n$ by which we group points in $\mathbb{N}^n$. This can be done by using term order $\succ$ as a "tie breaker" on the points that have the same value under function cx; That is, we define $>_c$ to encode c as a linear order on $\mathbb{N}^n$:

$$x >_c y \iff \begin{cases} cx > cy \\ cx = cy \quad x \succ y \end{cases}$$

We can see $>_c$ is a total order on $\mathbb{N}^n$ and it ensures a unique optimum in every fiber of $IP_{A,c}$. Let $\mathcal{G}$ denote the set of all points in $\mathbb{N}^n$ that are nonoptimal with respect to $>_c$. The structure of $\mathcal{G}$ is characterized in the following Lemma. We first state Lemma 4.4.1 which is a geometric version of the algebraic Grodan-Dickson Lemma (Theorem 5, section 2.4 in [20]) and is used to prove Lemma 4.4.2.

Lemma 4.4.1 If $P \subseteq \mathbb{N}^n$, $P \neq \phi$, then there exists a minimal subset $\{p_1, ..., p_m\} \subseteq P$ that is finite and unique such that $p \in P$ implies $p_j \leq p$ (component-wise) for at least one j=1,...,m.

Lemma 4.4.2 There exists a unique, minimal, finite set of vectors $\alpha(1),...,\alpha(s) \in \mathbb{N}^n$ such that the set of all nonoptimal points with respect to $>_c$ in all fibers of $IP_{A,c}$ is a subset of $\mathbb{N}^n$ of the form

$$\mathcal{G} = \bigcup_{i=1}^{s} (\alpha(i) + \mathbb{N}^n)$$

where $\alpha(i) + \mathbb{N}^n = \alpha(i) + v, v \in \mathbb{N}^n$.

We now define a test set for $IP_{A,c}$ dependent on the matrix A and order $>_c$.

Definition 4.4.1 A set $\mathcal{G}_{>_c} \subseteq \mathbb{Z}^n$ is a test set for the family of IP (with respect to the matrix A and the order $>_c$) if
(1) for each nonoptimal point α in each fiber of $IP_{A,c}$, there exists $g \in \mathcal{G}_{>_c}$ such that $\alpha - g$ is a feasible solution in this fiber and $\alpha >_c \alpha - g$, and
(2) for the optimal point β in a fiber of $IP_{A,c}$, $\beta - g$ is infeasible for every $g \in \mathcal{G}_{>_c}$.

A test set for $IP_{A,c}$ provides an obvious algorithm to find the unique optimal solution of a feasible problem $IP_{A,c}(b)$. That is, starting an initial feasible solution α, a better solution can be found by subtracting α with g in the test set until the optimal solution β is reached.

We construct a set $\mathcal{G}_{>_c}$ for $IP_{A,c}$ as follows. Let $\mathcal{G}_{>_c} = \{g_i = (\alpha(i) - \beta(i)), i = 1,..., s\}$, where $\alpha(1),..., \alpha(s)$ are the unique minimal elements of $\mathcal{G}$ and $\beta(i)$ is the unique optimum with respect to $>_c$ in the $A\alpha(i)$-fiber of $IP_{A,c}$. We show below that $\mathcal{G}_{>_c}$ is indeed a test set for $IP_{A,c}$.

Geometrically we can think of $(\alpha(i) - \beta(i))$ as the directed line segment $\vec{g_i} = [\alpha(i), \beta(i)]$ in the $A\alpha(i)$-fiber of $IP_{A,c}$. The vector is directed from the nonoptimal point $\alpha(i)$ to the optimal $\beta(i)$ due to the minimization requirement in the problem. If we add a vector $v \in \mathbb{N}^n$ to both end points of $\vec{g_i}$, $\vec{g_i}$ will be translated from the $A\alpha(i)$-fiber of $IP_{A,c}$ to the $A(\alpha(i)+v)$-fiber of $IP_{A,c}$. Now we consider an arbitrary fiber of $IP_{A,c}$ and a feasible lattice point θ in this fiber. For each vector $\vec{g_i}$ in $\mathcal{G}_{>_c}$, check if it can be translated through some vector $v \in \mathbb{N}^n$ such that the tail of the translated vector is incident at θ. At θ, we draw all such possible translations of vectors from $\mathcal{G}_{>_c}$. Notice that the head of a translated vector is also incident at a feasible lattice points in the same fiber as θ. We do this construction for all feasible lattice points in all fibers of $IP_{A,c}$. From Lemma 4.4.2 and the definition of $\mathcal{G}_{>_c}$, it follows that no vector in $\mathcal{G}_{>_c}$ can be translated by $v \in \mathbb{N}^n$ such that its tail meets the optimum on a fiber.

Theorem 4.4.1 The above construction builds a connected, directed graph in every fiber of $IP_{A,c}$. The nodes of the graph are all the lattice points in the fiber and the edges are the translations of elements in $\mathcal{G}_{>_c}$ by nonnegative integral vectors. The graph in a fiber has a unique sink at the optimum in the fiber with respect to $>_c$ and the outdegree of every nonoptimal lattice point in the fiber is at least one.

Proof: Pick a fiber of IP and at each feasible lattice point construct all possible translations of the vectors $\vec{g_i}$ from the set $\mathcal{G}_{>_c}$ as described earlier. This results in a possibly disconnected directed graph in this fiber where nodes are the lattice points and edges the translations of elements in $\mathcal{G}_{>_c}$. Let α be a nonoptimal lattice point in this fiber. By Lemma 4.2.2, $\alpha = \alpha(i) + v$ for some $i \in \{1, ..., s\}$ and $v \in \mathbb{N}^n$. Now $\alpha' = \beta(i) + v$ also lies in this fiber and $\alpha >_c \alpha'$ since $\alpha(i) >_c \beta(i)$. Therefore $\vec{g_i}$ translated by $v \in \mathbb{N}^n$ is an edge in this graph and we can move along it from α to an improved point α' in the same fiber. This proves that from every nonoptimal lattice point in the fiber we can reach an improved point in the same fiber by moving along an edge of the graph. The structure of $\mathcal{G}$ and the construction of the set $\mathcal{G}_{>_c}$ imply that the outdegree of the optimal solution is zero. Therefore, a directed path from a nonoptimal point terminates precisely at the unique optimum of the fiber. This is turn implies connectedness of the graph. □

We call the graph in the b-fiber of $IP_{A,c}$ the $>_c$-skeleton of that fiber. We have two corollaries which follow from Theorem 4.4.1 as follows.

Corollary 4.4.1 In the $>_c$-skeleton of a fiber, there exists a directed path from every nonoptimal point α to the unique optimum β. The objective function value (with respect to c) of successive points in the decreases monotonically from α to β.

Corollary 4.4.2 The set $\mathcal{G}_{>_c}$ is a uniquely defined minimal test set for $IP_{A,c}$.

We illustrate the construction of $>_c$-skeleton by the following example.

Example 4.4.1 $\mathcal{G}_{>_c} = \{\vec{g_i} : i = 1, 2, 3, 4\}$,
$\vec{g_1}$=[(2,0,0),(0,0,1)], $\vec{g_2}$=[(1,0,1),(0,1,0)], $\vec{g_3}$=[(1,1,0),(0,0,2)], $\vec{g_4}$=[(0,0,3),(0,2,0)].
A feasible point θ=(6,0,0) in 6-fiber. The graph $>_c$-skeleton constructed is as Fig 4.1.

4.4.2 Gröbner bases and test sets

Now we know for any IP problem there is a minimal test set and by the test set we can derive an optimal solution from any feasible solution of the IP problem. But we do not know how to compute this test set. Fortunately by Gröbner basis theory, we found the test set corresponds directly to the reduced Gröbner basis of a special ideal associated with the constraint matrix A and the cost (objective) function cx. So, we can get the test set by computing the reduced Gröbner basis. In this subsection we discuss the equivalence between them.

We first introduce a special ideal "toric ideal" I_A associated with A.

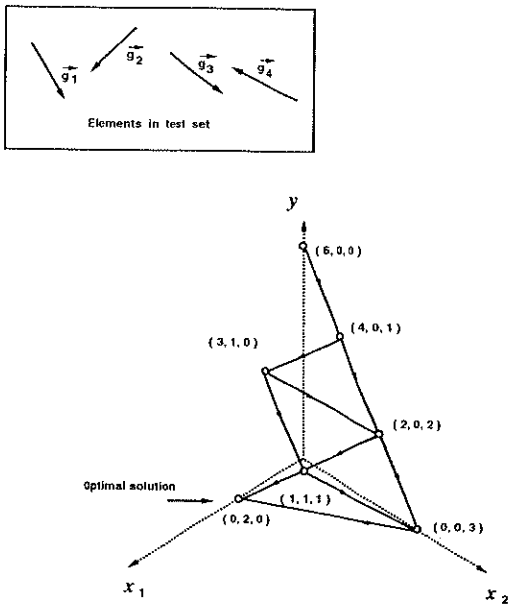


Figure 4.1: $>_c$-skeleton

Definition 4.4.2 The toric ideal I_A is a binomial ideal constructed from matrix A:

$$I_A = \langle x^\alpha - x^\beta : \alpha, \beta \in \mathbb{N}^n, \alpha - \beta \in ker(A) \rangle$$

where $ker(A) = \{v : v \in \mathbb{N}^n, Av = 0\}$

$\alpha - \beta \in ker(A)$ means $A(\alpha - \beta) = 0$, i.e. $A\alpha = A\beta$. Given an integral vector $u \in \mathbb{Z}^n$, we can write it uniquely as $u = u^+ - u^-$ where u^+ is the vector with $u_i^+ = u_i$ if $u_i \geq 0$ and $u_i^+ = 0$, otherwise; accordingly, u^- is the vector with $u_i^- = -u_i$ if $u_i < 0$ and $u_i^- = 0$, otherwise. So, we also write I_A as the following form:

$$I_A = \langle x^{u^+} - x^{u^-} : u \in ker(A) \rangle$$

Observation 4.4.1 Let α, β be vectors in $\mathbb{N}^n$. Then the binomial $x^\alpha - x^\beta$ is in I_A if and only if $A\alpha = A\beta$.

Proposition 4.4.1 Any reduced Gröbner basis of the toric ideal I_A consists of binomials.

Proof: By Hilbert basis theorem (Theorem 2.4.1) we can find a finite binomial generating set for I_A. If the input to the Buchberger algorithm (Algorithm 4.3.1)

is a set of binomials, then all intermediate polynomials created, as well as the final output, consists of binomials. □

Theorem 4.4.2 The test set $\mathcal{G}_{>_c}$ is the reduced Gröbner basis of I_A with respect to $>_c$.

Proof: We first show $\mathcal{G}_{>_c}$ is a Gröbner basis of I_A. By Lemma 4.2.2, we need to prove $Lt(\mathcal{G}_{>_c}) = Lt(I_A)$. Recall the test set $\mathcal{G}_{>_c} = \{(\alpha(i)-\beta(i)), i = 1, ..., s\}$, with respect to order $>_c$, which sorted points by cost c and then broke ties using a fixed term order $\succ$. As in the identification of a monomial with its exponent vector, we identify the vector $(\alpha(i)-\beta(i))$ with the binomial $x^{\alpha(i)}-x^{\beta(i)}$. For each g_i in $\mathcal{G}_{>_c}$, both $\alpha(i)$ and $\beta(i)$ were feasible solutions from the same fiber of IP. This implies that $A\alpha(i) = A\beta(i), \alpha(i), \beta(i) \in \mathbb{N}^n$ for all $i = 1, ..., s$. Therefore by Observation 4.4.1, $\langle x^{\alpha(i)}-x^{\beta(i)}, i = 1, ..., s\rangle \subseteq I_A$. Further, since $\alpha(i) >_c \beta(i)$ for all $i = 1, ..., s$, we have $lt(x^{\alpha(i)} - x^{\beta(i)}) = x^{\alpha(i)}$ for all i. Therefore, $\langle x^{\alpha(i)}, i = 1, ..., s\rangle \subseteq Lt(I_A)$, i.e., $Lt(\mathcal{G}_{>_c}) \subseteq Lt(I_A)$.

Now we show $Lt(I_A) \subseteq Lt(\mathcal{G}_{>_c})$, i.e., $lt(f) \in Lt(\mathcal{G}_{>_c})$ for each binomial f in I_A. We may assume that $lt(x^\alpha - x^\beta) = x^\alpha$ for each binomial in I_A. It is enough to show that $x^\alpha \in \langle x^{\alpha(i)}, i = 1, ..., s\rangle$ for each binomial $x^\alpha - x^\beta \in I_A$. Now $lt(x^\alpha - x^\beta) = x^\alpha$ implies that $\alpha >_c \beta$. Therefore α is a nonoptimal point with respect to $>_c$ in the $A\alpha$-fiber of $IP_{A,c}$. Therefore α is in $\mathcal{G}$, the set of all nonoptimal points from all fibers of $IP_{A,c}$. By Lemma 4.4.2, $\mathcal{G} = \cup_{i=1}^{s}(\alpha(i) + \mathbb{N}^n)$. This implies that $\alpha = \alpha(i) + v$ for some $i \in \{1, ..., s\}$ and $v \in \mathbb{N}^n$. Therefore $x^{\alpha(i)}$ divides x^α which in turn implies that $x^\alpha \in \langle x^{\alpha(i)}, i = 1, ..., s\rangle$. Thus $Lt(I_A) \subseteq Lt(\mathcal{G}_{>_c})$. Therefore $Lt(\mathcal{G}_{>_c}) = Lt(I_A)$, by Lemma 4.2.2, the test set $\mathcal{G}_{>_c}$ is a Gröbner basis for I_A with respect to $>_c$.

Then we show that $\mathcal{G}_{>_c}$ is in fact the reduced Gröbner basis of I_A respect to $>_c$. For $x^{\alpha(i)} - x^{\beta(i)} \in \mathcal{G}_{>_c}$, we first note that $x^{\alpha(i)}$ is not in $Lt(\mathcal{G}_{>_c} - \{g_i\})$ since $\alpha(1), ..., \alpha(s)$ were a unique minimal set of generators for $\mathcal{G}$. Also $x^{\beta(i)}$ is not in $Lt(\mathcal{G}_{>_c} - \{g_i\})$ since $\beta(i)$ was the unique optimum with respect to $>_c$ on its fiber and so is not in $\mathcal{G}$. Therefore $\mathcal{G}_{>_c}$ is the reduced Gröbner basis for I_A with respect to $>_c$. □

The Gröbner basis algorithm for $IP_{A,c}$ was first introduced by Conti and Traverso[18] and consists of the following steps:

1. Compute the reduced Gröbner basis $\mathcal{G}_{>_c}$ for I_A.

2. For a solution α to the program $IP_{A,c}(b)$, compute the normal form x^β of the x^α. Then β is the unique optimal solution of $IP_{A,c}(b)$.

Two important issues that are bypassed in the above version of the Conti-Traverso algorithm are those of (1) finding a generating set for the toric ideal I_A and (2)

finding an initial solution α for $IP_{A,c}(b)$. The original algorithm of Conti and Traverso[18] meets both of these requirements by computing the reduced Gröbner basis of a toric ideal in a larger polynomial ring with respect to an elimination order that depends on $>_c$. We will discuss the algorithm in detail in the next chapter.

The connections between Gröbner bases and IP point toward many exciting future directions of research. An important practical issue is the computability of Gröbner bases for IP. Devising both theoretical and programming tricks to speed up the Buchberger algorithm is an obvious step in this direction. Like in the classical techniques for IP, computations can often be fine tuned by exploiting the structure of the problems. If one is interested in solving $IP_{A,c}(b)$ for a fixed b, then often a small subset of the Gröbner basis $\mathcal{G}_{>_c}$ will suffice as a test set. Therefore, methods that incorporate b into the Buchberger algorithm (permitting shortcuts in computations) to produce a sufficient test set for $IP_{A,c}(b)$ are very worthwhile in this respect. An idea in this direction can be found in our new algorithm *Minimised Geometric Buchberger Algorithm* (MGBA) and will be described in the next chapter.

Chapter 5

Minimised Geometric Buchberger Algorithm (MGBA)

5.1 Introduction

As show in Chapter 4, The key idea of Gröber basis technique for IP is to encode an IP problem into a toric ideal associated with the constraint matrix A and the cost (objective) function cx. An important property of such encoding is that the reduced Gröbner basis of the toric ideal corresponds directly to the test set for the IP problem. The main difficulty of the technique is large size of the generated Gröbner bases. It depends on the structure of the IP problem and how to encode the IP problem into a toric ideal. In all methods of applying Gröbner basis technique to IP, there are two strategies for this encoding:

1. Indirect encoding: encoding with adding extra variables.

2. Direct encoding: encoding without adding extra variables.

The first strategy was firstly established by Conti and Traverso [18]. The scheme involves two encoding mechanisms: encoding the cost function of $IP_{A,c}$ into a linear order and encoding the coefficient matrix into a toric ideal. With this translation, solving IP problems is transformed into solving the sub-algebra membership problems (We describe this in section 5.2).

In [84], Thomas proposed a geometric interpretation of Conti-Traverso method. The key idea of Thomas' *Geometric Buchberger Algorithm* (GBA) is to relate the Gröbner bases of the encoded toric ideal of an IP to the notion of a test set for the IP. Each binomial is now directly interpreted as a directed line segment, i.e., a vector, in a lattice of all feasible solutions of $IP_{A,c}$. The Buchberger algorithm is then directly applied to a directed graph, where nodes of the graph are lattice points corresponding to feasible solutions of $IP_{A,c}$ and the edges at the beginning correspond to the input basis of the binomial ideal

I. Finding the reduced Gröbner basis amounts to rebuilding the graph such that the edges correspond to the members of the reduced Gröbner basis of I, which can be geometrically understood as a test set of the IP problem. Thus, by this graph, an optimal solution of $IP_{A,c}$ can be found along the directed path in the graph from a feasible solution. Thomas' work provides not only a succinct understanding of an algebraic IP solver but also a practical computational procedure for its implementation. In particular, this "generate and test" approach provides great inherent parallelism. In [55], we presented a parallel implementation of GBA on a 32 node Fujitsu AP1000+ MPP machine. The experiment showed that the algebraic approach towards IP provides a very promising mechanism. It also showed that the new method can be improved in various ways.

In the above strategy, the first problem is that the strategy is applied to an "extended" integer programming (EIP) with additional variables y, of the form:

$$\text{Min}\{My + cx : \ Iy + Ax = b\}$$

where $(y, x) \in \mathbb{N}^{m+n}$, I is the $m \times m$ identity matrix and $M \in \mathbb{R}^m$ is a vector whose components have large magnitude. In practice the additional variables will lead to a considerable increase in the space and time requirements of the algorithms considered.

The second problem is that the test set generated by both algorithms are generic in the sense that it is only determined by A and c for an IP problem $IP_{A,c}$. Thus, the search space for computing the reduced Gröbner basis for such a generalised problem is quite large. In [87], Thomas proposed the "Truncated Gröbner basis" method by fixing b to reduce the cardinality of the reduced Gröbner basis, but the size of Gröbner basis computed by the algorithm is still not optimal since the basis is for the EIP w.r.t a $IP_{A,c}(b)$, not the $IP_{A,c}(b)$ itself. So, many vectors in the reduced Gröbner basis are needed to move from an initial solution of EIP to an initial solution of IP.

The second strategy was established by Hosten and Sturmfels. In [45], Hosten and Sturmfels proposed an algorithm in which a generating set of $IP_{A,c}$ can be computed without going through EIP. This algorithm starts with a basis for the lattice $ker(A)$ and then proceeds to refine this basis to a generating set of toric ideal I_A for $IP_{A,c}$. But the algorithm is algebraic and generic. That is, the form of toric ideal I_A is:

$$\langle x^{u^+} - x^{u^-}, u \in B \text{ (lattice basis for } ker(A))\rangle$$

The space constructed is not a truncated space since the vector b is not taken into account. Thus, the efficiency is still a problem when the method is applied to large scale IP problems due to the complexity of the Buchberger algorithm.

In this chapter, we propose a new algebraic algorithm for solving IP. The new algorithm, called the *Minimised Geometric Buchberger Algorithm* (MGBA), combines the Hosten and Sturmfels method (GRIN) and Thomas' truncated GBA

to compute the generating set of an IP problem $IP_{A,c}$ directly in its original space and also the truncated Gröbner basis for the fixed b.

This chapter is organised as follows. In section 5.2 and section 5.3, we give a brief sketch of the approaches of strategy 1 and strategy 2, respectively. Section 5.4 introduces the idea of the Truncated Gröbner basis [87]. In section 5.5, we present a new algorithm to compute test set for IP. We present the implementation of this new algorithm and also show some computational result to compare MGBA with other algorithms such as the algorithm in GRIN, the geometric Buchberger algorithm (GBA) and the truncated geometric Buchberger algorithm (TGBA) in section 5.6. Finally we give a concluding remark in section 5.7.

5.2 Strategy 1: indirect encoding

In this strategy, we first translate an IP problem into an "extended" integer programming (EIP) problem by introducing additional variables y:

$$\text{EIP(b)} = \text{Min}\{My + cx : \ Iy + Ax = b, \ (y, x) \in \mathbb{N}^{m+n}\}$$

where I is the identity matrix, $M \in \mathbb{N}^m$ is a vector whose components have large magnitude. We use $\text{EIP}_{A,c}$ to denote the whole family of these IP problems, with fixed A and c, but varying right-hand side b.

From EIP(b), we can see all of the programs in it are feasible: They have the obvious solution $x = 0, y = b$. An optimal solution will satisfy $y = 0, x = x_o$ if the IP(b) is feasible, because the components of M are sufficiently large. If IP(b) is infeasible, then EIP(b) has an optimal solution with $y > 0$.

There are two approaches as follows to solve the EIP problems.

5.2.1 Algebraic approach

The algebraic approach was proposed by Conti and Traverso [18]. The scheme involves two encoding mechanisms:

1. Encoding the cost function $(M\ c)$ into a linear order on $\mathbb{Z}^{m+n}$. This can be done by choosing an arbitrary term order $>_o$ (here we let $>_o$ be the lexicographic order) and use it as a "tie breaker" on the points that have the same value under function $(M\ c)$; That is, we define $>_{M,c}$ to encode $(M\ c)$ as a linear order on $\mathbb{Z}^{m+n}$:

$$x_1 >_{M,c} x_2 \iff \begin{cases} (M\ c)x_1 < (M\ c)x_2 \\ (M\ c)x_1 = (M\ c)x_2 \ \text{ and } \ x_1 >_o x_2 \end{cases}$$

2. Encoding $(I\ A)$ into a toric ideal:

$$I = \langle y^{Ae_j} - x_j : j = 1, \ldots, n \rangle$$

where e_j is the j-th unit vector on $\mathbb{Z}^n$.

Actually, Ae_j is a projection of j-th column of A. So,

$$I = \langle y^{a^1} - x_1, \ldots, y^{a^n} - x_n \rangle$$

where $a^1, \ldots, a^n$ are the columns of A.

Let $\phi : k[x_1, ..., x_n] \longrightarrow k[y_1, ..., y_n]$ be the image defined by $x_j \longmapsto y_1^{a_{1j}} \cdots y_m^{a_{mj}}$. With this translation, an IP problem becomes a subalgebra membership problem of determining whether y^b is in the image of ϕ.

Let $\mathcal{G}$ be the reduced Gröbner basis of I with respect to $>_{M,c}$. According to Shannon and Sweedler subalgebra membership theorem [1], $g \in k[y_1, ..., y_m]$ is in the image of ϕ if and only if the remainder $r_{\mathcal{G}}(g)$ of g on division by $\mathcal{G}$ is in $k[x_1, ..., x_n]$. Thus, first we compute the reduced Gröbner basis of I. Then we divide y^b by $\mathcal{G}$. If $r_{\mathcal{G}}(y^b) \in k[x_1, ..., x_n]$ then the remainder is a monomial whose exponent vector is an optimal solution for the IP problem, otherwise the IP problem has no feasible solution.

Example 5.2.1 $\text{Min}\{x_1 + x_2 : 3x_1 + 2x_2 = 6,\ x_1, x_2 \in \mathbb{N}\}$
Firstly, we translate it into EIP:

$$\text{EIP} : \text{Min}\{100y + x_1 + x_2 :\ y + 3x_1 + 2x_2 = 6,\ (y, x_1, x_2 \in \mathbb{N})\}$$

Encoding $(I\ A)$ into a toric ideal:

$$\text{I} = \langle y^{(3\,2)(1\,0)^T} - x_1,\ y^{(3\,2)(0\,1)^T} - x_2 \rangle = \langle y^3 - x_1,\ y^2 - x_2 \rangle$$

Computing reduced Gröbner basis:

$$\mathcal{G} = \{y^2 - x_2,\ yx_2 - x_1,\ yx_1 - x_2^2,\ x_2^3 - x_1^2\}$$

Computing optimal solution of IP(b):

$$r_{\mathcal{G}}(y^6) = x_1^2$$

So, optimal solution is: $x_1 = 2,\ x_2 = 0$

5.2.2 Geometric approach

The Geometric Buchberger Algorithm (GBA), proposed by Thomas [84], is based on an geometric interpretation of the above algebraic approach. The key idea is to relate the Gröbner basis of encoded toric ideal to the test set for IP. A test set for an IP problem $IP_{A,c}$ is a set of vectors in $\mathbb{Z}^n$ such that for each non-optimal solution u to a problem in this family, there is at least one element g in this set such that $u - g$ has an improved cost value as compared to u.

By the theorem in [84], the unique minimal test set of EIP is the reduced Gröbner basis $\mathcal{G}_{>_{M,c}}$ of the toric ideal $I = \langle y^{Ae_j} - x_j :\ j = 1, ..., n \rangle$. Thus, the optimal solution of $IP_{A,c}(b)$ can be therefore computed by using $\mathcal{G}_{>_{M,c}}$ to improve the obvious feasible solution $(b, 0)$ to optimum $(0, v)$ of EIP(b). Here, we are really dealing with a geometric algorithm operating on lattice vectors.

Segment vector : Geometric Polynomial Each binomial in the ideal can be interpreted as a directed line segment, i.e., a vector by reading off its exponents. For example, we translate $x_1^2x_3x_5^3 - x_2^3x_4x_6^2$ directly into the vector $[(2,0,1,0,3,0),(0,3,0,1,0,2)]$. For a vector $d = [\alpha, \beta]$, $\alpha, \beta \in \mathbb{N}^{m+n}$, the tail $d^t = \alpha$ of the vector is more expensive than the head $d^h = \beta$ according to the order $>_{M,c}$ defined as follow:

$$\beta >_{M,c} \alpha \iff \begin{cases} \beta(M\ c)^T < \alpha(M\ c)^T \\ \beta(M\ c)^T = \alpha(M\ c)^T \ \text{ and } \ \beta >_o \alpha \end{cases}$$

where $>_o$ is the lexicographic order.

The generating set of the ideal I, called set of fundamental segments, can be easily constructed by translating each binomial $y_1^{a_{1j}} \cdots y_m^{a_{mj}} - x_j$ of I into a vector d_j.

S-Vector : Geometric S-Polynomial The geometric correspondence of S-Polynomial is called S-vector of two segment vectors d_1, d_2, which is computed as the difference of the heads of two vectors yielded by translated d_1, d_2 into a fiber on which their tails meet. This procedure is shown in Fig.5.1.

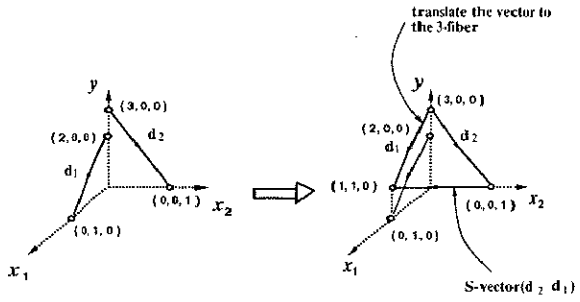


Figure 5.1: Computing S-Vector

Geometric Reduction: Reducing a vector f by a set of vectors $\{f_i\}$. if the tail of f can be reduced by f_i ($f_i^t >_{M,c} f^t$), then translate f_i so that f_i^t meets f^t and replace f by $\bar{f}$ joining f^h and f_i^h of translated f_i directed from the expensive to cheap end; if the head of f can be reduced by f_i ($f_i^t >_{M,c} f^h$), then translate f_i so that f_i^t meets f^h and replace f by $\bar{f}$ joining f^t and f_i^h of translated f_i directed from the expensive to cheap end. Repeat this procedure until there is no such f_i existing. Actually, the geometric interpretation of reducing a vector f by a set of vectors $\{f_i\}$ is constructing a path from the tail of f to the head of $\bar{f}$, as Fig.5.2.

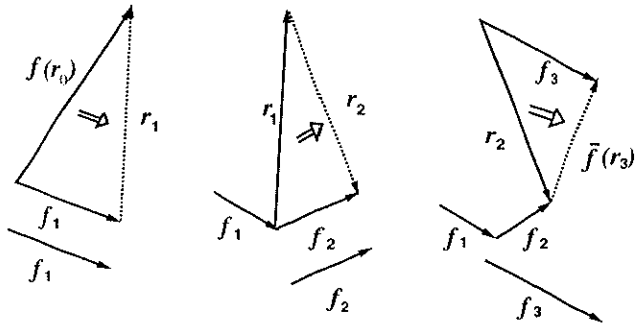


Figure 5.2: Geometric Reduction

Algorithm 5.2.1 Geometric Buchberger Algorithm:
The algorithm computes the reduced Gröbner basis of the ideal I with the term order $>_{M,c}$.

Step 1: Construct a Gröbner basis

INPUT $F = \{d_1, ..., d_n\}$, the fundamental segments of EIP(b) directed according to $>_{M,c}$

SET $\mathcal{G}_{old} := \emptyset$, $\mathcal{G} := F$

REPEAT While $\mathcal{G}_{old} \neq \mathcal{G}$, repeat the following steps

$\mathcal{G}_{old} := \mathcal{G}$

(S-vector) construct the pairs $g := d_i - d_j$

(reduction) reduce the vectors g by the vectors in $\mathcal{G}_{old}$. If $\bar{g} \neq 0$, set $\mathcal{G} := \mathcal{G} \cup \{g\}$.

Step 2: Construct a minimal Gröbner basis

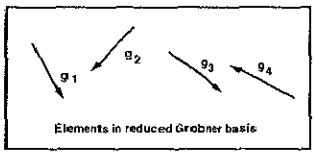
REPEAT If for some $g \in \mathcal{G}$ the tail g^t can be reduced by some $g' \in \mathcal{G} \backslash g$, then delete g from $\mathcal{G}$.

Step 3: Construct the reduced Gröbner basis

REPEAT If for some $g \in \mathcal{G}$ the head g^h can be reduced by some $g' \in \mathcal{G} \backslash g$, then replace g by the corresponding reduced vector: $\mathcal{G} := \mathcal{G} \backslash g \cup \{\bar{g}\}$.

OUTPUT $\mathcal{G}_{red} := \mathcal{G}$.

Example 5.2.2 (Same data as example 5.2.1)
EIP : $\{min\ 100y + x_1 + x_2 :\ y + 3x_1 + 2x_2 = 6,\ (y, x_1, x_2 \in \mathbb{N})\}$
Computing fundamental segments by interpreting binomials of I geometrically.
$I = \langle y^3 - x_1, y^2 - x_2 \rangle = \langle y^3 x_1^0 x_2^0 - y^0 x_1^1 x_2^0, y^2 x_1^0 x_2^0 - y^0 x_1^0 x_2^1 \rangle$
So, the fundamental segments are:
$d_1 = [(3,0,0),(0,1,0)], d_2 = [(2,0,0),(0,0,1)]$
Computing reduced Gröbner basis $\mathcal{G}_{>_{M,c}}$:
$g_1 = [(2,0,0),(0,0,1)],\ g_2 = [(1,0,1),(0,1,0)],$
$g_3 = [(1,1,0),(0,0,2)],\ g_4 = [(0,0,3),(0,2,0)].$
Deriving the optimal solution of $IP_{A,c}(b)$ from the feasible solution (6,0,0) by using $\mathcal{G}_{>_{M,c}}$, we get the optimal solution: $x_1 = 2$, $x_2 = 0$, as Fig.5.3.



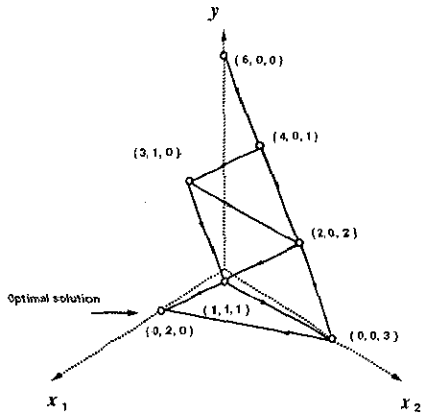


Figure 5.3: Example 5.2.2

5.3 Strategy 2: direct encoding

A typical approach of this strategy is the GRIN method. GRIN (GRöbner basis for INteger programming) is an experimental software system developed by Serkan Hosten and Bernd Sturmfels for computing the Gröbner basis of toric ideal, in particular, for solving IP problems using Gröbner bases. The algorithm coded in GRIN introduces a new method for computing the reduced Gröbner basis of the toric ideal which operates entirely in $k[x_1, ..., x_n]$ rather than in the

auxiliary polynomial ring $k[y_1, ..., y_m, x_1, ..., x_n]$. In GRIN, two algorithms are implemented. Here we discuss only one.

The algorithm in GRIN works in three stages: Stage 1 encodes an IP problem $IP_{A,c}$ into a subideal of the toric ideal I_A; stage 2 computes the toric ideal I_A from the subideal; stage 3 computes the reduced Gröbner basis $\mathcal{G}_{>_c}$ of I_A with respect to cost function cx.

Stage 1 and 3 are easy. We can encode A into a subideal of the toric ideal I_A by finding arbitrary lattice basis for $ker(A)$ with some methods such as Hermit normal form algorithm [65] or Smith normal form algorithm [74]. After we get the toric ideal I_A, we can use Buchberger algorithm to compute the reduced Gröbner basis for I_A. Here we focus on stage 2 computing the toric ideal I_A from a subideal.

Definition 5.3.1 If f is a polynomial in $k[x_1, ..., x_n]$ and $J \subseteq k[x_1, ..., x_n]$ is an ideal, then we call the following two subsets of $k[x_1, ..., x_n]$ *ideal quotients* and they are again ideals:

$$(J : f) = \{g \in k[x_1, ..., x_n] : fg \in J\},$$

$$(J : f^\infty) = \{g \in k[x_1, ..., x_n] : f^r g \in J \text{ for some } r \in \mathbb{N}\}$$

A basic formula involving ideal quotients is $(I : fg) = ((I : f) : g)$. A general method for computing Gröbner bases of the ideals from generators of J can be found in [20]. If J is an homogeneous ideal and f is one of the variables, say, $f = x_n$, then the algorithm for computing the Gröbner basis of the ideal from J is provided by the following lemma.

Lemma 5.3.1 Fix the graded reverse lexicographic term order induced by $x_1 > ... > x_n$, and let $\mathcal{G}$ be the reduced Gröbner basis of a homogeneous ideal $J \subset k[x_1, ..., x_n]$. Then the set

$$\mathcal{G}' = \{f \in \mathcal{G} : x_n \text{ does not divide } f\} \cup \{f/x_n : f \in \mathcal{G} \text{ and } x_n \text{ divides } f\}$$

is a Gröbner basis of $(J : x_n)$. A Gröbner basis of $(J : f^\infty)$ is obtained by dividing each element $f \in \mathcal{G}$ by the highest power of x_n that divides f.

Proof: We show that $\mathcal{G}'$ is a Gröbner basis for $(J : x_n)$. The proof of the second assertion about $(J : f^\infty)$ is analogous. Let $g \in (J : x_n)$. Then $lp(x_n \cdot g) = x_n \cdot lp(g)$ is divisible by $lp(f)$ for some $f \in \mathcal{G}$. Our choice of term order guarantees that x_n divides f if and only if x_n divides $lp(f)$. If this is the case, then f/x_n lies in $\mathcal{G}'$ and $lp(f/x_n)$ divides $lp(g)$. If x_n does not divide f, then f lies in $\mathcal{G}'$ and $lp(f)$ divides $lp(g)$. In either case $lp(g)$ is divisible by $lp(f')$ for some $f' \in \mathcal{G}'$. Thus by the definition of Gröbner bases (Definition 4.2.1), $\mathcal{G}'$ is a Gröbner basis for $(J : x_n)$. □

The term order in Lemma 5.3.1 makes sense whenever the ideal J is homogeneous with respect to some positive grading $deg(x_i) = d_i > 0$. By iterating the Gröbner basis computation n times with respect to different reverse lexicographic term orders, that is, by applying Lemma 5.3.1 one variable at a time, we can compute the ideal quotient

$$(J : (x_1 x_2 \cdots x_n)^\infty) = ((\cdots (J : {x_1}^\infty) : {x_2}^\infty) \cdots) : {x_n}^\infty)$$

So, if we find the relationship between the toric ideal I_A and ideal quotient, we can compute I_A and further the reduced Gröbner basis for I_A. Let $B \subseteq ker(A)$, we associate a subideal of I_A:

$$J_B := \langle x^{v^+} - x^{v^-} : v \in B \rangle$$

where $v = v^+ - v^-$ is the usual decomposition into positive and negative part.

We have the following lemma whose proof is in [79].

Lemma 5.3.2 A subset B spans the lattice $ker(A)$ if and only if

$$(J_B : (x_1 \cdots x_n)^\infty) = I_A$$

From Lemma 5.3.1 and Lemma 5.3.2, we can proof the following Proposition.

Proposition 5.3.1 Let $J_0 = \langle x^{v^+} - x^{v^-} : v \in B \rangle$ and $J_i = (J_{i-1} : {x_i}^\infty)$ $(i = 1, ..n)$ with the graded reverse lexicographic term order by making x_i the reverse lexicographically cheapest variable. Then J_n is the toric ideal I_A.

The lemmas and proposition stated above give the following algorithm which computes the reduced Gröbner basis of a toric ideal.

Algorithm 5.3.1 Algorithm in GRIN

- 1. Find any lattice basis B for $ker(A)$.
- 2. (Optional) Replace B by a reduced lattice basis B_{red}.
- 3. Let $J_0 := \langle x^{u^+} - x^{u^-} : u \in B_{red} \rangle$.
- 4. For i=1,...,n: Compute $J_i := (J_{i-1} : {x_i}^\infty)$ by making x_i the reverse lexicographically cheapest variable.
- 5. Compute the reduced Gröbner basis of $J_n = I_A$ for the desired term order. If the term order is obtained from an objective function cx, then the computed reduced Gröbner basis is the minimal test set for $IP_{A,c}$.

Example 5.3.1 Let m=4, n=8 and consider the matrix A

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 0 & 1 & 4 & 5 \\ 2 & 3 & 4 & 1 & 1 & 4 & 5 & 0 \\ 3 & 4 & 1 & 2 & 4 & 5 & 0 & 1 \\ 4 & 1 & 2 & 3 & 5 & 0 & 1 & 4 \end{pmatrix}$$

STEP 1 and STEP 2 :

Compute basis for the lattice $ker(A)$. In this case we get the reduced basis B_{red} as follow.

$$B_{red} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -3 & 0 & 0 & 0 & 2 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & -1 & 0 & 1 & 0 \end{pmatrix}$$

Here the basis for $ker(A)$ was expressed as a matrix whose every row is in $ker(A)$. In the new algorithm *Minimised Geometric Buchberger Algorithm* in section 5.5, we will give another expression of a basis for $ker(A)$ which is better than this one.

STEP 3:

By splitting each row of B_{red} into positive and negative parts and interpreting it as a binomial, we get the binomial ideal J_0 associated with B_{red} :

$$J_0 = \langle x_2x_4 - x_6x_8,\ x_2x_8^2 - x_4^3,\ x_1x_3 - x_5x_7,\ x_1^2x_7 - x_3^2x_5 \rangle$$

STEP 4:

In this step, We need to make eight Gröbner basis computations with respect to certain reverse lexicographic orders, starting with J_0. After each Gröbner basis computation we need to divide out certain variables. What we get after the computations of eight Gröbner bases is a generating set of I_A.

Entering the loop in this step, we first compute the reduced Gröbner basis for J_0 with respect to the reverse lexicographic order that makes x_1 the cheapest variable. Here is $x_1 < x_2 < x_3 < x_4 < x_5 < x_6 < x_7 < x_8$. The result is

$$G_0 = \{x_3^3x_1 - x_7^2x_1^2,\ x_4^3 - x_8^2x_2,\ x_5x_3^2 - x_7x_1^2, x_7x_5 - x_3x_1,\ x_8x_6 - x_4x_2\}$$

Next we divide each binomial in G_0 by x_1 whenever possible. So for example $x_3^3x_1 - x_7^2x_1^2$ when divided by x_1 gives $x_3^3 - x_7^2x_1$ and none of the other can be divided by x_1 (the cheapest variable in the above order).

Then we get a new set J_1 which consists of all the binomials in G_0 divided by x_1 whenever possible.

$$J_1 = \langle x_3^3 - x_7^2x_1,\ x_4^3 - x_8^2x_2,\ x_5x_3^2 - x_7x_1^2,\ x_7x_5 - x_3x_1,\ x_8x_6 - x_4x_2 \rangle$$

Now we compute the reduced Gröbner basis G_1 for J_1 by using the reverse lexicographic order that makes x_2 the cheapest variable. For example we use the order $x_1 > x_8 > x_7 > x_6 > x_5 > x_4 > x_3 > x_2$. The result is

$$G_1 = \{x_4^3 - x_8^2x_2,\ x_7x_5 - x_1x_3, x_8x_6 - x_4x_2,\ x_1x_7^2 - x_3^3,\ x_1^2x_7 - x_5x_3^2,\ x_1^3x_3 - x_5^2x_3^2\}$$

Dividing each binomial in G_1 by x_2 whenever possible, we get J_2:

$$J_2 = \langle x_4^3 - x_8^2x_2,\ x_7x_5 - x_1x_3, x_8x_6 - x_4x_2,\ x_1x_7^2 - x_3^3,\ x_1^2x_7 - x_5x_3^2,\ x_1^3x_3 - x_5^2x_3^2\rangle$$

Then we can repeat the process, every time computing the reduced Gröbner basis G_i for J_i by using the reverse lexicographic order that makes x_{i+1} the cheapest variable and then dividing each binomial in G_i by x_{i+1} to get J_{i+1}. Finally we get J_8, that is the generating set of the toric ideal I_A.

$$\begin{aligned} J_8 = \langle & x_2^4 - x_6^3x_8,\ x_3^3 - x_7^2x_1,\ x_4x_2 - x_6x_8,\ x_4^3 - x_2x_8^2, \\ & x_5x_3^2 - x_7x_1^2,\ x_5^2x_3 - x_1^3,\ x_6x_4^2 - x_2^2x_8,\ x_6^2x_4 - x_2^3, \\ & x_7x_5 - x_3x_1\rangle \end{aligned}$$

STEP 5:

Now, we can use J_8 as a generating set to compute the reduced Gröbner basis of I_A with a fixed term order. Here, the reduced Gröbner basis of I_A with respect to the lexicographic term order given by $x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > x_7 > x_8$ equals

$$\begin{aligned} \mathcal{G} = \{ & x_1^3 - x_3x_5^2,\ x_1^2x_7 - x_3^2x_5,\ x_1x_3 - x_5x_7,\ x_1x_7^2 - x_3^3, \\ & x_2^3 - x_4x_6^2,\ x_2^2x_8 - x_4^2x_6,\ x_2x_4 - x_6x_8,\ x_2x_8^2 - x_4^3, \\ & x_3^4 - x_5x_7^3,\ x_4^4 - x_6x_8^3\} \end{aligned}$$

5.4 Truncated Gröbner bases

The computation of the entire reduced Gröbner basis associated with the family of programs $IP_{A,c}$ is often expensive or impossible. In practice, we are often interested in solving $IP_{A,c}(b)$ for a fixed right hand side vector b, which typically requires only a subset of the entire Gröbner basis. In [87], Thomas proposed a truncated Buchberger algorithm called b-Buchberger algorithm for toric ideals that finds a sufficient test set for $IP_{A,c}(b)$. This set is a proper subset of the reduced Gröbner basis of I_A. So, by the algorithm, we can produce a minimal test set for the family of integer programs whose right hand side vector is smaller than or equal to b in a specific sense, which greatly improves the computation.

Let $C_{\mathbb{N}}(A) = \{\sum_{i=1}^n m_ia_i : m_i \in \mathbb{N}\}$, where a_i is the i-th column of the matrix A $(i = 1, ..., n)$. Then $C_{\mathbb{N}}(A)$ is a monoid and the $IP_{A,c}(b)$ is feasible if and only if b lies in $C_{\mathbb{N}}(A)$. We have the following lemma:

Lemma 5.4.1 The toric ideal $I_A = \bigoplus_{\beta \in C_{\mathbb{N}}(A)} I_A(\beta)$ where $I_A(\beta)$ is the vector space spanned by the binomials $\{x^u - x^v : Au = Av = \beta, u, v \in \mathbb{N}^n\}$.

Proof: The ideal I_A is spanned as a vector space by the binomials $\{x^u - x^v : Au = Av = \beta, u, v \in \mathbb{N}^n\}$. The above decomposition is the obvious grading of I_A indexed by elements of $C_{\mathbb{N}}(A)$, where the component $I_A(\beta)$ is the vector space spanned by $\{x^u - x^v : Au = Av = \beta \in C_{\mathbb{N}}(A)\}$. □

Let M denote the set of all monomials in $k[x_1, ..., x_n]$. The monoids M and $\mathbb{N}^n$ are isomorphic via the usual identification of a monomial x^u with its exponent vector. Under this identification, the monoid homomorphism π_A induces a multivariate grading of M, where the π_A-degree of x^u denoted $\pi_A(x^u) = \pi_A(u) = Au \in C_{\mathbb{N}}(A)$. Let $\mathrm{M}(f)$ denote the monomials in a polynomial $f \in k[x_1, ..., x_n]$.

Definition 5.4.1 A polynomial $0 \neq f \in k[x_1, ..., x_n]$ is side to be π_A-homogeneous if $\pi_A(s) = \pi_A(t)$ for all monomials $s, t \in \mathrm{M}(f)$. The π_A-degree of a homogeneous polynomial f, denoted $\pi_A(f)$, equals the π_A-degree of any monomial in $\mathrm{M}(f)$.

With Lemma 5.4.1 and Definition 5.4.1, we have the following lemma:

Lemma 5.4.2 The toric ideal I_A is homogeneous with respect to the grading induced by π_A.

Proof: A binomial $x^u - x^v$ lies in I_A if and only if $u, v \in \mathbb{N}^n$ and $Au = Av$. Binomials of this form are clearly π_A-homogeneous. Let f_β denote the sum of all monomials of π_A-degree β, in a non-zero polynomial $f \in I_A$. By Lemma 5.4.1, f is a linear combination of homogeneous binomials with $f_\beta \in I_A(\beta) \subset I_A$ for all $\beta \in C_{\mathbb{N}}(A)$. Hence I_A is homogeneous with respect to π_A. □

Associated with the monoid $C_{\mathbb{N}}(A)$ there is a "natural" partial order $\succeq$ such that for $b_1, b_2 \in C_{\mathbb{N}}(A)$, $b_1 \succeq b_2$ if and only if $b_1 - b_2 \in C_{\mathbb{N}}(A)$. Notice that when $C_{\mathbb{N}}(A) = \mathbb{N}^m$, the partial order $\succeq$ coincides with the componentwise partial order $\geq$, where $b_1 \geq b_2$ if and only if $b_1 - b_2 \geq 0$.

Now we are ready to describe the b-Buchberger algorithm. Let $\mathrm{NF}_{\{G, >_c\}}(g)$ denote the normal form of a binomial g, modulo a set of binomials G with term order $>_c$ and $S\text{-}bin(g_1, g_2)$ denote the S-binomial of two binomials g_1 and g_2.

Algorithm 5.4.1 b-Buchberger algorithm for toric ideals:

Input: A finite homogeneous binomial basis F of I_A and the term order $>_c$.

Output: A truncated (with respect to b) Gröbner basis of I_A.

$i = -1, G_0 = F$

Repeat

$$i = i + 1$$
$$G_{i+1} = G_i \cup (\{\mathrm{NF}_{\{G_i,>_c\}}(S\text{-}bin(g_1,g_2)) : g_1, g_2 \in G_i, \pi_A(S\text{-}bin(g_1,g_2)) \preceq b\} \backslash \{0\})$$

Until $G_{i+1} = G_i$

Reduce G_{i+1} modulo the leading monomials of its elements.

The algorithm 5.4.1 considers a S-binomial $g = x^u - x^v$ for reduction if and only if $\pi_A(g) = Au = Av \preceq b$. This amounts to checking feasibility if the system $\{x \in \mathbb{N}^n : Ax = b - Au\}$ which is as hard as solving the original IP problem $IP_{A,c}(b)$. Therefore, in order to implement the algorithm in practice, Thomas proposed two relaxations of the above check. Consider the S-binomial $g = x^u - x^v \in I_A$ for reduction if:

- $b - Au \in C(A)$ where $C(A) = \{Ax : x \in \mathbb{R}^n_+\}$. i.e., check feasibility of the linear programming relaxation of the original check.
- $b - Au \in C(A) \cap ZA$ where $ZA = \{Az : z \in \mathbb{Z}^n\}$. This is a relaxation of the original check since in general, $C_{\mathbb{N}}(A)$ is strictly contained in $C(A) \cap ZA$.

In our implementation, we use the second check by introducing the Hermite normal form, see the *Minimised Buchberger Geometric Algorithm* in section 5.5. When $C_{\mathbb{N}}(A) = \mathbb{N}^m$, we just use $b - Au \geq 0$, as the following example.

Example 5.4.1 (continue with example 5.3.1)
Consider the example 5.3.1 in section 5.3. Suppose we are interested only in right hand side vectors $b = (n_1, n_2, n_3, n_4)$ which satisfy $n_1 \leq 8$ and $n_2 \leq 8$. In this case, the truncated Gröbner basis equals $\{x_1^3 - x_3x_5^2,\ x_1x_3 - x_5x_7,\ x_2x_4 - x_6x_8\}$. Because here $C_{\mathbb{N}}(A) = \mathbb{N}^4$, the degree Au of these three binomials are (3,6,9,12), (4,6,4,6) and (6,4,6,4), which satisfy $b - Au \geq 0$, while for the other seven binomials, their degree Au are so large that $b - Au < 0$. Thus, they lie outside of $C_{\mathbb{N}}(A)$.

5.5 Minimised Geometric Buchberger Algorithm

Combining the GRIN method, the truncated Gröbner bases and geometric Buchberger algorithm together, we propose an new Buchberger algorithm, called *Minimised Geometric Buchberger Algorithm* (MGBA) for IP problems. The idea behind the new algorithm is that for a special IP problem $IP_{A,c}(b)$ with fixed b, its minimal test set corresponds to a truncated reduced Gröbner basis of the toric ideal I_A. We encode $IP_{A,c}(b)$ into a subideal of I_A first, and then compute I_A

using GRIN method, finally compute the truncated reduced Gröbner basis of I_A with b-Buchberger algorithm. We formulate the algorithm in the original space of $IP_{A,c}$ without introducing any additional variables and interpret all steps of the algorithm geometrically. This truncated reduced Gröbner basis, i.e., the minimal test set for $IP_{A,c}(b)$ we get is a subset of the test set for $IP_{A,c}$. So, with the new algorithm, we can obtain considerable improvements in the efficiency and applicability of the Gröbner basis technique for IP.

Before giving the description of the algorithm, we need to introduce the Hermite normal form from which we compute the lattice basis of $ker(A)$. For a fuller exposition of Hermite normal form see [65] and [74].

Definition 5.5.1 An $m \times m$ nonsingular integer matrix H is said to be in *Hermite normal form* if:

(a). H is lower triangular and $h_{ij} = 0$ for $i < j$,

(b). $h_{ij} > 0$ for $i = 1, ..., m$, and

(c). $h_{ij} \leq 0$ and $|h_{ij}| < h_{ii}$ for $i > j$.

Definition 5.5.2 Let C be a nonsingular integer matrix. Then C is called unimodular if C has determinant ± 1.

Theorem 5.5.1 If A is an $m \times n$ integer matrix with rank(A)=m, then there exists an $n \times n$ unimodular matrix C such that:

(a). $AC = (H, 0)$ and H is in *Hermite normal form*, and

(b). $H^{-1}A$ is an integer matrix.

$(H, 0)$ is called the *Hermite normal form* of A. In [65], there is a polynomial-time algorithm, called Hermite Normal Form Algorithm for finding C and H which serve as a constructive proof of Theorem 5.5.1. It also can be shown that H is unique.

Theorem 5.5.2 Let $S = \{x \in \mathbb{Z}^n : Ax = b\}$ and let H and $C = (C_1, C_2)$ be as in Theorem 5.5.1, with C_1 an $n \times m$ matrix and C_2 an $n \times (n - m)$ matrix.

(a). $S \neq \emptyset$ if and only if $H^{-1}b \in \mathbb{Z}^m$.

(b). If $S \neq \emptyset$, every solution of S is of the form

$$x = C_1 H^{-1} b + C_2 z, \quad z \in \mathbb{Z}^{n-m}$$

Proof:

$$\begin{aligned} S &= \{x \in \mathbb{Z}^n : Ax = b\} \\ &= \{x : x = Cw,\ ACw = b,\ w \in \mathbb{Z}^n\} \text{ (since } C \text{ is unimodular)} \\ &= \{x : x = Cw,\ (H,0)w = b,\ w \in \mathbb{Z}^n\} \\ &= \{x : x = C_1w_1 + C_2w_2,\ Hw_1 = b,\ w_1 \in \mathbb{Z}^m,\ w_2 \in \mathbb{Z}^{n-m}\} \\ &= \{x : x = C_1H^{-1}b + C_2w_2,\ H^{-1}b \in \mathbb{Z}^m,\ w_2 \in \mathbb{Z}^{n-m}\}. \end{aligned}$$

□

From Theorem 5.5.2, we have a computation of a basis for $ker(A)$ stated in the following theorem:

Theorem 5.5.3 Let B be a basis for $ker(A)$ and let H and $C = (C_1, C_2)$ be as in Theorem 5.5.2. Then $B = \{c_i : c_i \text{ is the } i\text{-th column of } C_2 \text{ and } i = 1, .., n-m\}$.

Proof: Suppose $x \in ker(A)$. Then $Ax = 0$. From Theorem 5.5.2, we have

$$x = C_1H^{-1}0 + C_2w_2 = C_2w_2$$

where w_2 is an arbitrary $(n-m)$-vector of integers.
Thus,

$$x = \sum_{i=1}^{n-m} \nu_i c_i$$

where c_i is the i-th column of C_2 and $\nu_i \in \mathbb{Z}$, $i = 1, ..., n-m$.
Therefore, $B = \{c_i : c_i \text{ is the } i\text{-th column of } C_2 \text{ and } i = 1, .., n-m\}$. □

So, by the Hermite Normal Form Algorithm [65], we can compute H and $C = (C_1, C_2)$ for a matrix A, as well as a basis for $ker(A)$.

Now we are ready to describe our new algorithm MGBA. In this algorithm, the segment vector is slightly different from one in geometric Buchberger algorithm. For a vector $d = [\alpha, \beta]$ in our algorithm, $\alpha, \beta \in \mathbb{N}^n$ and α is more expensive than β according to the term order $>_c$ defined as follow:

$$\beta >_c \alpha \iff \begin{cases} \beta c^T < \alpha c^T \\ \beta c^T = \alpha c^T \text{ and } \beta >_o \alpha \end{cases}$$

where $>_c$ is encoded from the objective function cx of $IP_{A,c}$ and $>_o$ is the lexicographic order. The fundamental segments in MGBA are constructed by interpreting the binomials of toric ideal $I_A = \langle x^{u^+} - x^{u^-} : u \in ker(A)\rangle$ geometrically.

Algorithm 5.5.1 Minimised Buchberger Geometric Algorithm

1. Compute lattice basis B for ker(A)

(1.1) Use the Hermite normal form algorithm to compute H and $C = (C_1, C_2)$.

(1.2) Compute the basis B:
$B = \{c_i : c_i \text{ is the } i\text{-th column of } C_2 \text{ and } i = 1, .., n-m\}$

2. Reduce B into reduced lattice basis B_{red}

Here we use *Reduced Basis Algorithm* in [65] to compute the reduced lattice basis B_{red}. The purpose of this step is to make some big numbers in B smaller so that we can speed up the following computation by using the simplified (reduced) basis.

3. Compute toric ideal I_A

(3.1) Compute a subideal of I_A based on B_{red}

$$J_0 := \langle x^{u^+} - x^{u^-} : u \in B_{red} \rangle$$

Then interpret each binomial in J_0 as a vector by reading off its exponents. Here we can directly translate an element of B_{red} into a vector of J_0. For example, let $u \in B_{red}$ and $u = (1, 2, -1, -2)$, then the translated vector $v = [(1, 2, 0, 0), (0, 0, 1, 2)]$.

(3.2) For i=1,2,...,n: Compute $J_i := (J_{i-1} : x_i^{\infty})$ geometrically by making x_i the reverse lexicographically cheapest variable. Here each J_i is in geometric term, i.e., its elements are all vectors. So, we use *Geometric Buchberger Algorithm* (Algorithm 5.2.1) to compute the reduced Gröbner basis for each J_i, $i = 1, ..., n$.

4. Compute the truncated Gröbner basis $\mathcal{G}_{>_c}(b)$ of I_A with order $>_c$

Input: generating set J_n of toric ideal I_A and term order $>_c$
Output: truncated reduced Gröber basis $\mathcal{G}_{>_c}(b)$

(4.1) Construct a Gröbner basis
In the first step of Algorithm 5.2.1 (*Geometric Buchberger algorithm*) we add a related check of the truncated Gröbner basis into the computation of S-vector :

$$b - Au \in C(A) \cap ZA \text{ where } ZA = \{Az : z \in \mathbb{Z}^n\}$$

we check whether there is feasible solution for $S = \{x \in \mathbb{Z}^n : Ax = b - Au\}$ by (a) of Theorem 5.5.2 stated as follow:

S is not empty if and only if $H^{-1}(b - Au) \in \mathbb{Z}^m$

(4.2) Construct a minimal Gröbner basis
This step is same as the second step of Algorithm 5.2.1.

(4.3) Construct the reduced Gröbner basis
This step is same as the third step of Algorithm 5.2.1.

We illustrate the whole procedure of the above algorithm by the following example.

Example 5.5.1 We consider the following IP problem $IP_{A,c}(b)$:

Min $x_1+8x_2+8x_3+16x_4+2x_5+2x_6+2x_7+2x_8$ subject to

$$x_1+2x_2+3x_3+4x_4+x_6+4x_7+5x_8=7,$$
$$2x_1+3x_2+4x_3+x_4+x_5+4x_6+5x_7=7,$$
$$3x_1+4x_2+x_3+2x_4+4x_5+5x_6+x_8=13,$$
$$5x_1+2x_2+3x_3+4x_4+6x_5+x_6+2x_7+5x_8=17.$$

From the IP problem, we have the coefficient matrix A, vector c and b as follows.

$$A=\begin{pmatrix} 1 & 2 & 3 & 4 & 0 & 1 & 4 & 5 \\ 2 & 3 & 4 & 1 & 1 & 4 & 5 & 0 \\ 3 & 4 & 1 & 2 & 4 & 5 & 0 & 1 \\ 5 & 2 & 3 & 4 & 6 & 1 & 2 & 5 \end{pmatrix}$$

$c=(1,8,8,16,2,2,2,2)$ $b=(7,7,13,17)$

STEP 1.1: We use the Hermite normal form algorithm to compute H and C for A and get a basis B for $ker(A)$ as follows.

$$H=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ -11 & -12 & -2 & 22 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C=\begin{pmatrix} -3 & -3 & -2 & 7 & -3 & 0 & -4 & 0 \\ 2 & 1 & 1 & -3 & 0 & -3 & 0 & -4 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 2 & 0 & 3 & 0 \\ 0 & 1 & 0 & -1 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Then,

$$C_2=\begin{pmatrix} -3 & 0 & -4 & 0 \\ 0 & -3 & 0 & -4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 0 & 2 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

From G_2 we can get the basis B for $ker(A)$:

$B = \{(-3,0,1,0,2,0,0,0),\ (0,-3,0,1,0,2,0,0),$
$(-4,0,0,0,3,0,1,0),\ (0,-4,0,0,0,3,0,1)\}$

STEP 1.2: We compute the reduced basis B_{red} for $ker(A)$ and get:

$B_{red} = \{(-1,0,-1,0,1,0,1,0),\ (0,-1,0,-1,0,1,0,1),$
$(-2,0,2,0,1,0,-1,0),\ (0,-2,0,2,0,1,0,-1)\}$

STEP 3.1: We interpret each element of B_{red} as a vector. For example, (-1,0,-1,0,1,0,1,0) is translated to the vector $[(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)]$. So, we get a subideal of I_A as follow.

$J_0 = \langle [(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)], [(0,0,0,0,0,1,0,1),(0,1,0,1,0,0,0,0)],$
$[(0,0,2,0,1,0,0,0),(2,0,0,0,0,0,1,0)], [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)]\rangle$

STEP 3.2: We compute the toric ideal I_A. we first use Algorithm 5.2.1 to compute the reduced Gröbner basis for J_0 with respect to reverse lexicographic order that makes x_1 the cheapest variable. Here is $x_1 < x_2 < x_3 < x_4 < x_5 < x_6 < x_7 < x_8$. The result is

$G_0 = \{[(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)], [(0,0,0,0,0,1,0,1),(0,1,0,1,0,0,0,0)],$
$[(0,0,2,0,1,0,0,0),(2,0,0,0,0,0,1,0)], [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(1,0,3,0,0,0,0,0),(2,0,0,0,0,0,2,0)], [(0,1,0,3,0,0,0,0),(0,2,0,0,0,0,0,2)]\}$

Next we divide each vector in G_0 by x_1 whenever possible, just as removing the common factor x_1 from two monomials. For example $[(1,0,3,0,0,0,0,0),(2,0,0,0,0,0,2,0)]$ when divided by x_1 gives $[(0,0,3,0,0,0,0,0),(1,0,0,0,0,0,2,0)]$ and none of the other can be divided by x_1 (the cheapest variable in the above order).

Then we get a new set J_1 which consists of all the vectors in G_0 divided by x_1 whenever possible.

$J_1 = \langle [(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)], [(0,0,0,0,0,1,0,1),(0,1,0,1,0,0,0,0)],$
$[(0,0,2,0,1,0,0,0),(2,0,0,0,0,0,1,0)], [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(0,0,3,0,0,0,0,0),(1,0,0,0,0,0,2,0)], [(0,1,0,3,0,0,0,0),(0,2,0,0,0,0,0,2)]\rangle$

Now we compute reduced Gröbner basis G_1 for J_1 by using the reverse lexicographic order that makes x_2 the cheapest variable. The order is $x_1 > x_8 > x_7 > x_6 > x_5 > x_4 > x_3 > x_2$. The result is

$G_1 = \{[(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)], [(0,0,0,0,0,1,0,1),(0,1,0,1,0,0,0,0)],$
$[(2,0,0,0,0,0,1,0),(0,0,2,0,1,0,0,0)], [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(1,0,0,0,0,0,2,0),(0,0,3,0,0,0,0,0)], [(0,1,0,3,0,0,0,0),(0,2,0,0,0,0,0,2)],$
$[(3,0,1,0,0,0,0,0),(0,0,2,0,2,0,0,0)]\}$

Dividing each binomial in G_1 by x_2 whenever possible, we get J_2:

$J_2 = \langle[(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)],[(0,0,0,0,0,1,0,1),(0,1,0,1,0,0,0,0)],$
$[(2,0,0,0,0,0,1,0),(0,0,2,0,1,0,0,0)],[(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(1,0,0,0,0,0,2,0),(0,0,3,0,0,0,0,0)],[(0,0,0,3,0,0,0,0),(0,1,0,0,0,0,0,2)],$
$[(3,0,1,0,0,0,0,0),(0,0,2,0,2,0,0,0)]\rangle$

Then we can repeat the process, every time computing reduced Gröbner basis G_i for J_i by using reverse lexicographic order that makes x_{i+1} the cheapest variable and then dividing each vector in G_i by x_{i+1} to get J_{i+1}. Finally we get J_8, that is the the toric ideal I_A.

$J_8 = \langle[(0,0,0,0,1,0,1,0),(1,0,1,0,0,0,0,0)],\ [(0,1,0,1,0,0,0,0),(0,0,0,0,0,1,0,1)],$
$[(0,0,2,0,1,0,0,0),(2,0,0,0,0,0,1,0)],\ [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(0,0,3,0,0,0,0,0),(1,0,0,0,0,0,2,0)],\ [(0,0,0,3,0,0,0,0),(0,1,0,0,0,0,0,2)],$
$[(0,0,1,0,2,0,0,0),(3,0,0,0,0,0,0,0)],\ [(0,0,0,1,0,2,0,0),(0,3,0,0,0,0,0,0)],$
$[(0,4,0,0,0,0,0,0),(0,0,0,0,0,3,0,1)]\rangle$

STEP 4: In this step, we can use J_8 as the fundamental segments to compute truncated reduced Gröbner basis of I_A with fixed right hand side b and cost function cx. The test set for $IP_{A,c}(b)$, i.e., the truncated reduced Gröbner basis is:

$\mathcal{G}_{>_c}(b) = \{[(1,0,1,0,0,0,0,0),(0,0,0,0,1,0,1,0)],\ [(0,1,0,1,0,0,0,0),(0,0,0,0,0,1,0,1)],$
$[(0,0,2,0,1,0,0,0),(2,0,0,0,0,0,1,0)],\ [(0,0,0,2,0,1,0,0),(0,2,0,0,0,0,0,1)],$
$[(0,0,3,0,0,0,0,0),(1,0,0,0,0,0,2,0)],\ [(0,0,0,3,0,0,0,0),(0,1,0,0,0,0,0,2)],$
$[(0,0,1,0,2,0,0,0),(3,0,0,0,0,0,0,0)],\ [(0,3,0,0,0,0,0,0),(0,0,0,1,0,2,0,0)]\}$

For any feasible solution of the problem $IP_{A,c}(b)$, we can derive an optimal solution by using the above test set to reduce this feasible solution. For example, we have a feasible solution:

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1,\ x_5 = 1,\ x_6 = 0,\ x_7 = 0,\ x_8 = 0.$$

Then we can get an optimal solution:

$$x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 0,\ x_5 = 1,\ x_6 = 1,\ x_7 = 0,\ x_8 = 1.$$

Theorem 5.5.4 The algorithm MGBA terminates after a finite number of steps and its output is the unique minimal test set for $IP_{A,c}(b)$.

Proof: Finiteness of the algorithm is clear since the Hermite normal form algorithm, the Buchberger algorithm and b-Buchberger algorithm all terminate in finitely many steps.

By Proposition 5.3.1, we obtain the toric ideal I_A in step 3. Because the generating set of the toric ideal I_A is a set of fundamental segments for $IP_{A,c}$,

The geometric Buchberger algorithm and b-Buchberger algorithm guarantee that we can obtain the truncated reduced Gröbner basis $\mathcal{G}_{>_c}(b)$ for $IP_{A,c}(b)$ with the term order $>_c$ in step 4. Now, we study $\mathcal{G}_{>_c}(b)$ from a completely geometric point of view. With $\mathcal{G}_{>_c}(b)$, we can build a connected, directed graph for only one fiber (b-fiber) of $IP_{A,c}(b)$. The nodes of the graph are all the lattice points in the fiber and the edges are the translations of elements in $\mathcal{G}_c(b)$ by nonnegative integral vectors. By Theorem 4.4.1 in Chapter 4, the graph has a unique sink at the unique optimum in this fiber. In this graph, there exists a directed path from every nonoptimal point to the unique optimum. So, the reduced Gröbner basis $\mathcal{G}_c(b)$ is a test set for $IP_{A,c}(b)$. By Corollary 4.4.2 in Chapter 4, we can proof $\mathcal{G}_c(b)$ is the unique minimal test set for $IP_{A,c}(b)$, depends on A, $>_c$ and b. □

5.6 Implementation, Experiment and Comparison

We have implemented the Minimised Geometric Buchberger Algorithm (MGBA) by language C and develop a solver, called `MGBS` (Minimised Geometric Buchberger Solver) for IP on a Sun Ultra Enterprise 3000. `MGBS` works on two stages: the first stage is to compute a test set (reduced Gröbner basis) $\mathcal{G}_c(b)$ for $IP_{A,c}(b)$ based on MGBA, the second is to find a feasible solution and derive the optimal solution for $IP_{A,c}(b)$ by using $\mathcal{G}_c(b)$ to reduce the feasible solution. `MGBS` is connected via MathLink and CGI to the modelling IP system `TIP` described in Chapter 3. When `MGBS` is called in Web page with an IP model (an objective function and a set of constraints), `MGBS` will solve the IP model and send the result (an optimal solution or a message "no feasible solution") to the Web page via CGI.

`MGBS` is an experimental software developed for solving IP problems using Gröbner bases. The key difficult is the computation of Gröbner bases. `MGBS` uses conventional Gröbner basis techniques to speed up this computation whenever this is suitable. For example, we make effective use of criteria to cut down the number of S-binomials, which is a bottleneck during the computations. It is common strategy to keep the set of binomials throughout the entire Gröbner basis computation as reduced as possible. We implemented this idea by doing global reductions periodically (whenever new elements of the size of a fixed percentage of the current basis are created) as opposed to doing it every time a new binomial is added. Another important strategy is the extraction of common monomial factors in every newly created S-binomial. This extraction is justified by the fact that the toric ideal I_A is a prime ideal not containing any common monomials. The above idea proved to be very effective, leading to reductions as much as 40-50% in running times.

We are particularly interested in having practitioners of IP problems test our program, and to provide us with feedback and suggestions. It is our belief that there is still much room for further improvements in the efficiency and applicability of the Gröbner basis approach.

We also have implemented the Geometric Buchberger Algorithm (GBA), the Truncated Geometric Buchberger Algorithm (TGBA) and the algorithm in GRIN, simply called GRIN by language C respectively on the Sun Ultra Enterprise 3000. We have carried out an experiment by running `MGBS` on randomly generated matrices A of various sizes (ranging from 3×7 to 8×16) with nonnegative entries in a range between 0 and 20. We generate random right hand sides b to compute truncated Gröbner basis $\mathcal{G}_c(b)$. For each test instance (A,c,b) three comparisons are made with GBA, TGBA and GRIN.

Problems	Entries	MGBA		GRIN		TGBA		GBA	
		Size	Time	Size	Time	Size	Time	Size	Time
A3x7.1	0-20	17	0.34	31	0.55	212	68.84	245	420.90
A3x7.2	0-20	20	2.29	69	4.64	236	78.68	345	1129.26
A3x7.3	0-20	26	3.88	72	5.42	237	79.88	723	4746.13
A4x8.1	0-20	21	4.30	95	40.29	823	8476.86	3390	35118.24
A4x8.2	0-20	25	17.72	103	116.40	847	9913.19	3831	39426.20
A4x8.3	0-20	33	92.37	124	217.30	949	12118.89	4814	42042.16
A5x10.1	0-4	55	65.72	85	70.57	1143	826.27	1766	11887.24
A5x10.2	0-4	57	65.89	92	73.18	1171	896.76	1890	11995.97
A5x10.3	0-4	65	66.42	102	73.38	1284	930.23	2014	12207.70
A6x12.1	0-3	100	112.29	181	256.54	1300	1569.63	2353	18600.83
A6x12.2	0-3	153	189.87	418	1348.34	3304	3671.52	5026	24189.12
A6x12.3	0-3	267	358.39	709	3119.38	7872	8593.21	9590	35870.38
A8x16.1	0-1	19	212.60	30	215.77	167	313.45	928	8250.13
A8x16.2	0-1	18	210.78	26	215.72	136	267.36	702	6385.31
A8x16.3	0-1	11	6.76	20	7.46	48	13.89	63	15.67

Table 5.1: Experimental Result

The result of the comparisons is as table 5.1. The first column of the table represents IP problems with their coefficient matrix A. For example, A3x7.1 represents No.1 IP problem with 3×7 integer matrix A. The range of the entries used in the problems are given in the second column. The third and fourth columns give the size of the truncated reduced Gröbner basis and the execution time for computing it for each problem with MGBA. The fifth and sixth columns

give the two kinds of data (size and execution time) for GRIN. The seventh and eighth columns are for TGBA. The last two columns are for GBA. The timings are in CPU seconds on the Sun Ultra Enterprise 3000.

From table 5.1, we can see the reduced Gröbner basis in GBA is the biggest one among the all algorithms because of the introduction of additional variables. Also the execution time is longest. For TGBA, because the algorithm computes the reduced Gröbner basis by fixing b, we can see the size of the reduced Gröbner basis in TGBA is less than that in GBA. But it is greater than that in GRIN and MGBA, because TGBA still introduces the additional variables to compute the reduced Gröbner basis. The size of the reduced Gröbner basis generated by MGBA is much less than those in the other three algorithms and also the performance in MGBA is the best.

5.7 Concluding remarks

We have proposed a new algorithm *Minimised Geometric Buchberger Algorithm* for IP. It combines the GRIN method and the truncated Gröbner bases method to compute a generating set of the Gröbner bases in the original space and then refine it into a minimal test set, i.e., a truncated reduced Gröbner basis of $IP_{A,C}(b)$ with fixed right hand side. Our primary experiments indicate that the algorithm is much faster than others such as the geometric Buchberger algorithm, the truncated geometric Buchberger algorithm and the algorithm in GRIN.

Yet we are still far from applying our algorithm to large scale problem instances. There is still much room for further improvements in the efficiency and applicability of the Gröbner basis approach. The future research includes the parallelisation of the algorithm and the application of the algorithm to stochastic IP problems. Due to the high degree of inherent parallelism in the algorithm, we expect that the parallelisation will result in a practical IP solver. Applying Gröbner bases technique to stochastic IP is a new strategy for solving stochastic IP based on symbolic computation. The property of Gröbner basis corresponding directly to the test set of the IP problem does seem particularly useful for solving the classes of stochastic problems where some or all variables are integer which the general numerical methods can not handle. So, with our new algorithm for IP, we can provide an efficient method for solving the stochastic IP problems. This idea will be described in the next chapter.

Chapter 6

Applying MGBA to Stochastic Integer Programming

6.1 Introduction

As shown in Chapter 5, MGBA provides a new mechanism for solving IP problems. In particular, we have realised that for many complex IP problems such as stochastic IP problems where there is no efficient solving method available, MGBA provides a new promising solution. In this chapter, we describe a method for solving a class of stochastic IP problems, chance constrained IP problem, using algorithm MGBA.

The rest of this chapter is organized as follows. In section 2, we describe a class of chance constrained IP problems from practical applications. In section 3 we introduce a test set for the chance constrained IP problems and prove that its properties guarantee the search from an optimal solution of reduced IP to the optimum of the chance constrained IP can always terminate. Section 4 presents an algorithm for solving chance constrained IP problems. In section 5, we apply the new method for chance constrained IP to solve a practical complex IP problem: job scheduling. We give an example to illustrate the whole procedure from modeling to solution and some experimental results.

6.2 Chance constrained IP problem

An IP problem $IP_{A,c}(b)$ with deterministic coefficients c_i, a_{ij}, b_i is limited use for modelling real world applications. For example, future productivities in a production problem, inflows into a reservoir connected to a hydro power station, demands at various nodes in a transportation network, and so on, are often appropriately modeled as uncertain parameters, which are at best characterized by probability distributions. The uncertainty about the realized values of those

	Products		Cost	Capacity
Raws	$prod_1$	$prod_2$	C	P
raw_1	2	3	2	1
raw_2	6	3	3	1
relation	$\geq$	$\geq$	$=$	$\leq$
D	180	162	ω	100

Table 6.1: Productivities $\pi(raw_i, prod_j)$

parameters can not always be wiped out just by inserting their mean values or some other (fixed) estimates during the modeling process. So, IP model is no longer be appropriate for describing the above practical problems and we need the model with some uncertain parameters, that is, the stochastic program model.

We consider the following problem, idealized for the purpose of easy presentation. From two raw materials, raw_1 and raw_2, we may simultaneously produce two different goods, $prod_1$ and $prod_2$ (as may happen for example in a refinery). The output of products per unit of the raw materials as well as the unit costs of the raw materials $C = (c_{raw_1}, c_{raw_2})^{\mathrm{T}}$ (yielding the production cost ω), the demands for the products $D = (d_{prod_1}, d_{prod_2})^{\mathrm{T}}$ and the production capacity P, i.e., the maximal total amount of raw materials that can be processed, are given in Table 6.1.

According to this formulation of our production problem, we have an IP model as following:

$$\begin{aligned}
&\text{Min } (2x_{raw_1} + 3x_{raw_2}) \text{ subject to} \\
&\qquad x_{raw_1} + x_{raw_2} \leq 100 \\
&\qquad 2x_{raw_1} + 6x_{raw_2} \geq 180 \\
&\qquad 3x_{raw_1} + 3x_{raw_2} \geq 162 \\
&\text{where } x_{raw_i} \in \mathbb{N} \text{ for } i = 1, 2
\end{aligned}$$

However, this is obviously not always a realistic assumption. From a stochastic viewpoint, there is a possibility that demands will exceed the allotted production capacity in any period, resulting in shortfall, in which the excess demand is backlogged (and so incur penalty costs), or the excess demand is satisfied from inventory which in turn needs to be replenished (and so incur inventory cost), or the excess demand is lost (and so lose the margin and possibly market share). If we don't consider the specific remedial strategy, we must schedule the productivity $\pi(raw_i, prod_j)$ to avoid frequent occurrences of a shortfall. In the schedule

discussed above, the shortfall probability is

$$1 - Probability(2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1,\ 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2)$$

where $Probability(2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1,\ 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2)$ means the probability of the two inequalities $2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1$ and $3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2$ holding and ξ_i (i=1,2) is a random variable. For a given probability of no shortfall γ, we will provide the optimal scheduling of productivities. So, here we model the problem as a stochastic program with a probabilistic constraint.

Min $(2x_{raw_1} + 3x_{raw_2})$ subject to
$x_{raw_1} + x_{raw_2} \leq 100$
$2x_{raw_1} + 6x_{raw_2} \geq 180 + \hat{\xi}_1$
$3x_{raw_1} + 3x_{raw_2} \geq 162 + \hat{\xi}_2$
$Prob\{2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1, 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2\} \geq \gamma$

where $x_{raw_i} \in \mathbb{N}$ and $\hat{\xi}_i$ is the mean of random ξ_i, i=1,2. The probabilistic constraint in the above model states that the probability of the demands not exceeding the allotted production capacities is at least γ.

Stochastic programming with probabilistic constraints as a decision model under uncertainty is called chance constrained programming. If some or all variables in the chance constrained programming problems are integers, it is called chance constrained IP (CIP).

We abstract the following model as a general class of CIP:

Min $h(x)$ subject to
$Prob\{Tx \geq \xi\} \geq \gamma$
$Ax = b$
where $x \in \mathbb{N}^n$ and ξ is a vector of random variables.

This model has two properties: (1) the probabilistic constraint is not separable, (2) integer variables are required to model setup decision. In available techniques for chance constrained programming, there is no satisfactory solution method existing for models that have integer variables and nonseparable chance (probabilistic) constraint arising from nonnormal distributions.

By applying MGBA, we can solve this problem. The idea is dividing the problem into two parts: one is composed of the probabilistic constraint and some complicated constraints, called *membership oracle* and the other is a simple IP after removing the membership oracle from the problem, called *reduced IP*. We first compute the test set (reduced Gröbner basis of a toric ideal) for reduced IP by using MGBA. The test set provides a set of directions that can be used to trace paths from every nonoptimal solution to the optimal solution of the reduced IP. So, for any feasible solution of the reduced IP, we get an optimal

solution by searching the set of directions. Simultaneously, we can also walk back from the optimal solution to every feasible solution of the reduced IP by simply reversing these paths. So, with the same test set, we find the optimal solution of the CIP by walking back from the optimal solution of the reduced IP to other feasible solutions and querying the membership oracle to check whether the reached point is feasible for the CIP. We prove that the search terminates with either the optimal solution of CIP or all paths are searched, i.e., the CIP is infeasible.

In next section, we will introduce a test set for CIP.

6.3 Test set for chance constrained IP

We consider a class of the chance constrained IP problems of the form:

$$\begin{aligned} \text{CIP:}\quad & \text{Min } cx \text{ subject to} \\ & Ax = b \\ & Prob\{Tx \geq \xi\} \geq \gamma \\ & x \in \mathbb{N}^n \end{aligned}$$

where ξ is a vector of random variables, $c \in \mathbb{R}^n$, $A \in \mathbb{Z}^{m \times n}$, $T \in \mathbb{R}^{l \times n}$, $b \in \mathbb{Z}^m$ and $\gamma \in \mathbb{R}$. First we divide CIP into two parts: reduced IP (RIP) and membership oracle. The reduced IP is the form as follow.

$$\begin{aligned} \text{RIP:}\quad & \text{Min } cx \text{ subject to} \\ & Ax = b \\ & x \in \mathbb{N}^n \end{aligned}$$

The membership oracle is composed of $Prob\{Tx \geq \xi\} \geq \gamma$. In some models for practical problems, the membership oracle may includes some complicated constraints except probability constraints, as example in Section 6.5.

By MGBA, we compute a test set for RIP and derive an optimal solution of RIP by using this test set to reduce any feasible solution of RIP. Now we need a test set for CIP, which provides a set of directions that can be used to walk in a systematic manner from the optimal solution of RIP to other solutions, querying the membership oracle each time to check feasibility with respect to CIP. The walking always terminates at the optimum for CIP or finding CIP infeasible. First we give the definition of test set for CIP.

Definition 6.3.1 A set $\Omega \subseteq \mathbb{Z}^n$ is a test set for CIP (with respect to $>_c$) if it provides a set of directions so that

(1) walking from RIP optimum using the directions always lands on a point that is feasible for RIP if nonnegativity constraints are satisfied;

(2) all feasible points of CIP can be reached using only the directions;

(3) on every path, every step deeper into the polytope (starting from the RIP optimum) increases the cost monotonely.

Let $\mathcal{G}_{>_c} = \{\alpha(i) - \beta(i), i = 1, ..., s\}$ be the minimal test set for RIP. We construct a set Ω for CIP as follows. Let $\Omega = \{\beta(i) - \alpha(i), i = 1, ..., s\}$ i.e., reverse all directions in the test set $\mathcal{G}_{>_c}$. We have the following theorem.

Theorem 6.3.1 The set Ω is a test set for CIP.

Proof: Consider the $>_c$-skeleton of the appropriate fiber. Suppose we now reverse the directions of all edges in this skeleton and call the resulting graph the reverse $>_c$-skeleton of the fiber. By Corollary 4.4.1, in the $>_c$-skeleton of a fiber, there exists a directed path from every nonoptimal point α of RIP to the unique optimum β. Thus, for any feasible point α of RIP, there exists a directed path $P(\alpha)$ in the reverse $>_c$-skeleton of this fiber, from the RIP optimum to this feasible in the fiber. Because the set of all feasible points of CIP is a subset of one of RIP, all feasible points of CIP can be reached by the reverse $>_c$-skeleton. Also, the objective function value of points reached as the path is traversed from the RIP optimum to α, increases monotonically.

So, we only show that we can build the reverse $>_c$-skeleton in a fiber by using Ω. Let Ω denote the set of vectors in $\mathcal{G}_{>_c}$ with directions reversed. At a feasible lattice point θ in a fiber of RIP, we draw all vectors from Ω that can be translated through some $v \in \mathbb{N}^n$ such that its tail is incident at θ. As before, the heads of the translated vectors are also incident at feasible lattice points in the same fiber. Since only the vectors in $\mathcal{G}_{>_c}$ were reversed, this construction and the previous one have the same underlying undirected graph in every fiber. This proves the claim. □

Example 6.3.1 We construct a test set Ω by reversing all vectors of $\mathcal{G}_{>_c}$ in Example 4.4.1 and draw a reversed $>_c$-skeleton as Fig.6.1.

$$\Omega = \{\overrightarrow{g_i} = [\alpha, \beta] : [\beta, \alpha] \in \mathcal{G}_{>_c} \text{ and } i = 1, 2, 3, 4\},$$
$$\overrightarrow{g_1} = [(0,0,1),(2,0,0)], \overrightarrow{g_2} = [(0,1,0),(1,0,1)],$$
$$\overrightarrow{g_3} = [(0,0,2),(1,1,0)], \overrightarrow{g_4} = [(0,2,0),(0,0,3)].$$

By the above definition and theorem, if we get the test set $\mathcal{G}_{>_c}$ for RIP, then it is easy to computer the test set Ω for CIP by simply reversing all vectors in $\mathcal{G}_{>_c}$. In Chapter 4, we have shown that the test set $\mathcal{G}_{>_c}$ for RIP $IP_{A,c}(b)$ is the reduced Gröbner basis for the toric ideal I_A and we can use MGBA to compute the test set. So, by MGBA, we can also compute the test set Ω for CIP. Further, we can compute the optimal solution of CIP by using this test set. In next section, we will describe an algorithm for CIP.

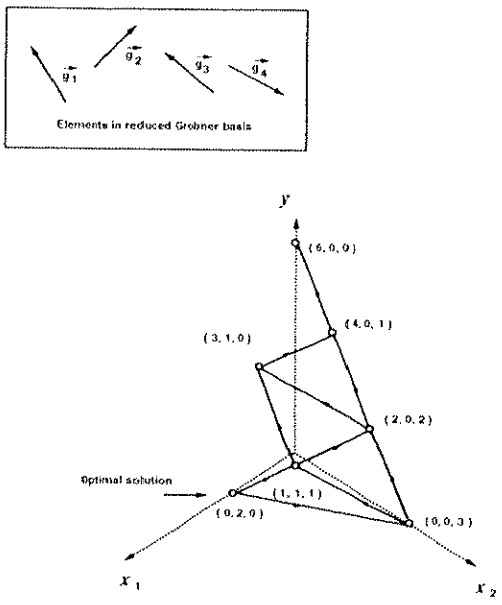


Figure 6.1: reverse $>_c$-skeleton

6.4 Algorithm for chance constrained IP

The definition and theorem about the test set for CIP provide us an algorithm for finding the optimal solution of CIP from the RIP optimum. We use the paths in the reverse $>_c$-skeleton of RIP to find the CIP optimum X_0. Let β be the RIP optimum and $P(\alpha)$ be a path in the reverse $>_c$-skeleton of the fiber from β to α. Let $\mathcal{G}$ be a set of path in the form $P(\alpha)$. We may assume that β is infeasible for CIP since otherwise we have done.

Algorithm 6.4.1 Feasible Checking Algorithm

This algorithm checks whether CIP is feasible or infeasible, if feasible, its optimum will be found.

Initialize: $\mathcal{G} = \{P(\beta)\}$, X_0=M (M$\in \mathbb{N}^n$ is a vector whose components have large magnitude.)

Repeat For $P(\alpha) \in \mathcal{G}$, let $P(\omega_1), \ldots P(\omega_q)$ be the paths obtained by adding to $P(\alpha)$ those edges in Ω such that $\omega_i \geq 0$. This ensures that $P(\omega_i)$ is a path in the reverse $>_c$-skeleton. A path that leads to an infeasible point may be assumed to be pruned at α.

For $i = 1$ to q

if ω_i feasible for CIP and $X_0 >_c \omega_i$, let $X_0 = \omega_i$ and prune $P(\omega_i)$.

else if $\omega_i >_c X_0$ **then** prune $P(\omega_i)$
else $\mathcal{G} = \mathcal{G} \cup \{P(\omega_i)\}$
$\mathcal{G} = \mathcal{G} - \{P(\alpha)\}$
Until all paths in $\mathcal{G}$ are pruned.

Theorem 6.4.1 Algorithm 6.3.1 terminates in finite steps and

(1) CIP is infeasible if and only if X_0=M, or

(2) CIP is feasible, X_0 is an optimal solution of CIP.

Proof: Finiteness of the algorithm is clear since the optimal solution of CIP is bounded with respect to $>_c$.

Also, it is clear that CIP is infeasible if and only if X_0=M. Now we show X_0 is the CIP optimum when CIP is feasible.

Let μ be any feasible of CIP. Clearly μ is feasible for RIP and there exists a path $P(\mu)$, in the reverse $>_c$-skeleton, from the optimum β of RIP to μ. Note that $P(\mu)$ may not be unique but any such path will suffice for this proof. We identify $P(\mu)$ with the sequence of nodes in the reverse $>_c$-skeleton that constitute it, i.e. $P(\mu) = \{\beta = u_1, ..., u_m = \mu\}$. Now let $j \in \{1, ..., m\}$ be the largest number such that $P(u_j)$ is in the set $\mathcal{G}$ of the above procedure at some stage. We have the following two cases:

(i) $P(u_j)$ was pruned since u_j became feasible for CIP.

In this case, $u_j >_c X_0$, that is $cu_j \geq cX_0$. Also $c\mu \geq cu_j$, since the successive nodes in $P(\mu)$ have monotonically increasing cost when the path is traversed from β to μ. Therefore $c\mu \geq cX_0$.

(ii) $P(u_j)$ was pruned since $u_j >_c X_0$.

Again $c\mu \geq cu_j \geq cX_0$.

Therefore, we have shown that in either case, we have $c\mu \geq cX_0$. So, X_0 is an optimal solution when CIP is feasible. □

Combining the feasible checking algorithm and MGBA, we can provide an algorithm for solving CIP. Before describing this algorithm, we first consider another part of the CIP, membership oracle.

The membership oracle is composed of the probabilistic constraint. Some times in order to keep the reduced IP as simple as possible, the membership oracle also includes some complicated constraints. In each step of walking the directions of the test set Ω from the RIP optimum to a feasible solution, we need to query the membership oracle, i.e., check whether the point is feasible for the CIP. Assume we have a probabilistic constraint $Prob\{Tx \geq \xi\} \geq \gamma$ in the

membership oracle and a set of samples $\{\xi_i, i = 1, ..., s\}$. Then a feasible point v of RIP is feasible for CIP if and only if

$$Tv \leq \xi_i,\ i \in \Xi \subseteq \{1, ..., s\}\ and\ |\Xi| \geq \gamma s$$

Now we give a whole algorithm for chance constrained IP.

Algorithm 6.4.2 Algorithm for CIP

1. Decompose CIP into RIP and membership oracle.
2. Compute the test set $\mathcal{G}_{>_c}$ of RIP by using MGBA (Algorithm 5.5.1).
3. Compute an optimal solution x^R from any feasible solution of RIP.
4. Derive the test set Ω of CIP by reversing all directions of vectors in $\mathcal{G}_{>_c}$
5. Compute the optimal solution of CIP with the test set Ω and the RIP optimum x^R by using the feasible checking algorithm (Algorithm 6.3.1).

Example 6.4.1 We consider the CIP problem in a refinery described in Section 6.2. The form of CIP is as follow:

$$\begin{aligned}
&\text{Min } (2x_{raw_1} + 3x_{raw_2}) \text{ subject to}\\
&x_{raw_1} + x_{raw_2} \leq 100\\
&2x_{raw_1} + 6x_{raw_2} \geq 180 + \hat{\xi}_1\\
&3x_{raw_1} + 3x_{raw_2} \geq 162 + \hat{\xi}_2\\
&Prob\{2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1, 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2\} \geq \gamma
\end{aligned}$$

Suppose $\gamma = 0.8$ and the following 5 samples of $\xi = (\xi_1, \xi_2)$ are used in the probabilistic constraint.

$$S = \{(12, 4), (5, 16), (8, 3), (2, 0), (3, 7)\}$$

Thus, two means of ξ_1 and ξ_2 are $\hat{\xi}_1 = 6$ and $\hat{\xi}_2 = 6$ respectively.

Step 1. Decompose CIP:

We decompose the above CIP into RIP and membership oracle.

$$\begin{aligned}
\text{RIP:}\quad &\text{Min } (2x_{raw_1} + 3x_{raw_2}) \text{ subject to}\\
&x_{raw_1} + x_{raw_2} \leq 100\\
&2x_{raw_1} + 6x_{raw_2} \geq 186\\
&3x_{raw_1} + 3x_{raw_2} \geq 168
\end{aligned}$$

The membership oracle is the probabilistic constraint.

$$Prob\{2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1, 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2\} \geq 0.8$$

Step 2. Compute the test set $\mathcal{G}_{>_c}$ for RIP:

First we convert the above RIP into the standard IP form by adding slack variables $a_i \in \mathbb{N}$, $i = 1, 2, 3$.

$$\begin{aligned}\text{RIP:}\quad &\text{Min } (2x_{raw_1} + 3x_{raw_2}) \text{ subject to}\\ &x_{raw_1} + x_{raw_2} + a_1 = 100\\ &2x_{raw_1} + 6x_{raw_2} - a_2 = 186\\ &3x_{raw_1} + 3x_{raw_2} - a_3 = 168\end{aligned}$$

By the algorithm MGBA, we compute the test set for RIP and get

$$\begin{aligned}\mathcal{G}_{>_c} = \{&[(0,1,0,4,0),(1,0,0,0,0)],\\ &[(1,0,0,2,3),(0,0,1,0,0)],\\ &[(2,0,0,0,3),(0,1,1,2,0)]\}.\end{aligned}$$

Step 3. Compute the optimal solution for RIP:

For any feasible solution of RIP, we can compute the optimal solution x^R by using $\mathcal{G}_{>_c}$ to reduce this feasible solution. Actually we use the system MGBS in Chapter 5 to find a feasible solution of RIP and compute x^R to get:
$x_{raw_1} = 37$, $x_{raw_2} = 19$, $a_1 = 44$, $a_2 = 2$, $a_3 = 0$, i.e., $x^R = (37, 19, 44, 2, 0)$.

Step 4. Derive the test set for CIP:

The test set Ω can be obtained by simply reversing the directions of all vectors in the test set $\mathcal{G}_{>_c}$ of RIP.

$$\begin{aligned}\Omega = \{&[(1,0,0,0,0),(0,1,0,4,0)],\\ &[(0,0,1,0,0),(1,0,0,2,3)],\\ &[(0,1,1,2,0),(2,0,0,0,3)]\}.\end{aligned}$$

Step 5. Compute the optimal solution of CIP:

We use the feasible checking algorithm to compute the optimal solution for CIP. First we check whether the optimal solution of RIP $x^R = (37, 19, 44, 2, 0)$ is feasible for CIP, that is to check whether x^R satisfies the probabilistic constraint in membership oracle. For every sample of ξ, we check the two linear nonequalities in the probabilistic constraint:

$$\begin{aligned}Eq_1:&\ 2x_{raw_1} + 6x_{raw_2} \geq 180 + \xi_1,\\ Eq_2:&\ 3x_{raw_1} + 3x_{raw_2} \geq 162 + \xi_2.\end{aligned}$$

If both are true, we assign a truth value T, otherwise assign F. Then we compute the probability γ by counting the number of T. If $\gamma \geq 0.8$, then x^R is feasible for CIP, otherwise x^R is infeasible for CIP. The whole checking procedure is illustrate in Table 6.2.

From Table 6.2, we see x^R is not feasible for CIP because it violates the probabilistic constraint in membership oracle with $\gamma = 0.4$.

Then from x^R, we walk on the directions of Ω to reach all new points. The new points can be obtained by using Ω to reduce x^R as follows.

$$\begin{aligned}v_1 &= x^R - (1,0,0,0,0) + (0,1,0,4,0) = (36,20,44,6,0),\\ v_2 &= x^R - (0,0,1,0,0) + (1,0,0,2,3) = (38,19,43,4,3),\\ v_3 &= x^R - (0,1,1,2,0) + (2,0,0,0,3) = (39,18,43,0,3).\end{aligned}$$

x^R	sample of ξ	Eq_1	Eq_2	result
$x_{raw_1} = 37, x_{raw_2} = 19$	$\xi_1 = 12, \xi_2 = 4$	F	T	F
$x_{raw_1} = 37, x_{raw_2} = 19$	$\xi_1 = 5, \xi_2 = 16$	T	F	F
$x_{raw_1} = 37, x_{raw_2} = 19$	$\xi_1 = 12, \xi_2 = 4$	T	T	T
$x_{raw_1} = 37, x_{raw_2} = 19$	$\xi_1 = 12, \xi_2 = 4$	T	T	T
$x_{raw_1} = 37, x_{raw_2} = 19$	$\xi_1 = 12, \xi_2 = 4$	T	F	F
$\gamma = \frac{2}{5} = 0.4$				

Table 6.2: Checking procedure

For a new point v, we have two checks. We first check whether v is negative, if so, we prune the path $P(v)$ from x^R to v, otherwise, we check whether v is feasible for the membership oracle, i.e., feasible for CIP, if so, we prune the path $P(v)$ and save v in X_0 if $X_0 >_c v$. If the point v is infeasible for CIP and $v >_c X_0$, then we prune $P(v)$, otherwise, we add $P(v)$ into the set of path $\mathcal{G}$ and continue to walk until all paths are pruned. Here v_1, v_2, and v_3 are all infeasible for CIP and they are less than X_0 w.r.t $>_c$. So, we add $P(v_1)$, $P(v_2)$ and $P(v_3)$ into $\mathcal{G}$. We continue to walk along v_1 to get following points:

$v_{11} = v_1 - (1,0,0,0,0) + (0,1,0,4,0) = (35,21,44,10,0),$
$v_{12} = v_1 - (0,0,1,0,0) + (1,0,0,2,3) = (37,20,43,8,3),$
$v_{13} = v_1 - (0,1,1,2,0) + (2,0,0,0,3) = (38,19,43,4,3).$

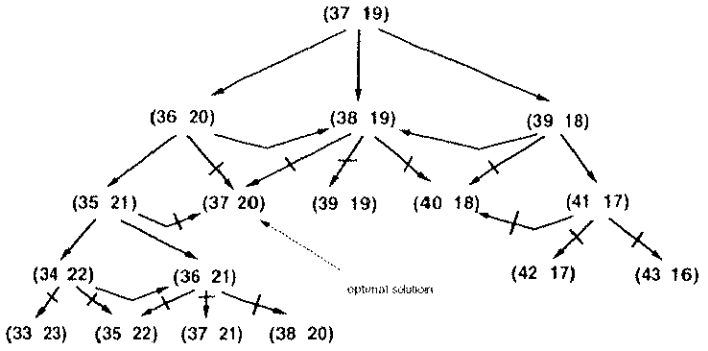


Figure 6.2: Walking procedure in Example 6.4.1

Point v_{11} is infeasible for CIP and $X_0 >_c v_{11}$, so add $P(v_{11})$ into $\mathcal{G}$. Point v_{12} is feasible for CIP and $X_0 >_c v_{12}$, so $X_0 = v_{12}$. Point v_{13} is the point v_2. We continue to walk along the leaving points until pruning all paths i.e. $\mathcal{G} = \emptyset$. Finally, we get the optimal solution $X_0 = (37, 20, 43, 8, 3)$. The whole search is illustrated by Fig.6.2 in which we use two elements x_{raw_1} and x_{raw_2} to represent a whole point, for example, the point $x^R = (37, 19, 44, 2, 0)$ is represented by (37 19).

6.5 Application

In this section, we apply our algorithm for chance constrained IP to solve the problem of job scheduling. Job scheduling is a famous IP problem in which many algorithms solve it successfully [6][93][77]. When we consider the randomness of some elements such as demands, the job scheduling becomes a stochastic IP problem. By the new algorithm for CIP proposed in Section 6.3, we can solve this problem efficiently. First we describe the job scheduling problem in the following subsection.

6.5.1 Job scheduling problem

We schedule n job types (indexed by i), whose demand in any given period is a random vector $(D_1, ..., D_n)$, on m machines (indexed by j) that have a capacity of C_j, $j = 1, ..., m$. The probability distribution of demand of the n job types in any period is $F(x_1, ..., x_n) = Probability\{D_1 \leq x_1, ..., D_n \leq x_n\}$, with means $(\hat{D}_1, ..., \hat{D}_n)$. The setup time for job type i on machine j is S_{ij}. Let K_{ij} denote the setup cost for job type i on machine j. To facilitate lot splitting, we have M_i, $i = 1, ..., n$, a set of management specified integers, that limits the maximum pieces (or lots) the demand of job type i can be partitioned into. Thus, if M_1=4, then on any machine only multiples of $\frac{1}{4}$ of the demand for product type 1 can be produced. Let L'_{ij} be the cost of producing a unit of product type i on machine j, and define $L_{ij} = \hat{D}_i / M_i L'_{ij}$. The processing time for *a unit of job type* i *on* machine j is p_{ij}. Let γ be the probability of no shortfall. The problem is modelled as the following chance constrained IP:

CIP: Min $\sum_i \sum_j K_{ij} z_{ij} + L_{ij} y_{ij}$ subject to

(1) $\sum_{j=1}^{m} y_{ij} = M_i$, $i = 1, .., n$

(2) $M_i z_{ij} \geq y_{ij}$, $i = 1, .., n$, $j = 1, ..., m$

(3) $\sum_{i=1}^{n} p_{ij}(\hat{D}_i / M_i) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j$, $j = 1, ..., m$

(4) $Prob\{\sum_{i=1}^{n} p_{ij}(D_i / M_i) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, ..., m\} \geq \gamma$

(5) $z_{ij} \in \{0, 1\}$, $y_{ij} \in \{0, 1, ..., M_i\}$

Variable z_{ij}=1 if job type i is scheduled on machine j, otherwise it is set to zero. Variable y_{ij}, an integer between zero and M_i, represents how many multiples of $\frac{1}{M_i}$ of demand of product i are schedule on machine j. Thus, if in a period the demand realized for product i is D_i, $(y_{ij}/M_i)D_i$ will be scheduled on machine j.

The objective is to minimize the expected costs due to setups and production. Eq. (1) states that all of the demand for product i needs to be scheduled. Eq. (2) states that production for job type i cannot take place on machine j unless the appropriate setup has occurred. Eq. (3) states that the average demand allocated to a machine does not exceed its capacity. These are the complicating constraints of CIP. Eq. (4), the probabilistic constraint, states that the probability of production time and setup time not exceeding the capacities on the respective machines is at least γ. Notice the difference of demand used in constraint (3) and (4), the demand in constraint (3) is mean $\hat{D}_i$, while the demand in (4) is random variable D_i.

Now according to the algorithm for CIP, we first decompose the above CIP into a reduced IP which includes constraint (1), (2), and (5) and membership oracle which includes (3) and (4). This decomposition makes RIP so simple that we can compute the optimal solution easily. the RIP is :

RIP: Min $\sum_i \sum_j K_{ij} z_{ij} + L_{ij} y_{ij}$ subject to

(1) $\sum_{j=1}^{m} y_{ij} = M_i,\ i = 1, .., n$

(2) $M_i z_{ij} \geq y_{ij},\ i = 1, .., n,\ j = 1, ..., m$

(5) $z_{ij} \in \{0, 1\},\ y_{ij} \in \{0, 1, ..., M_i\}$

The membership oracle is composed of a probabilistic constraint and complicated constraints. It is:

membership oracle:

(3) $\sum_{i=1}^{n} p_{ij}(\hat{D}_i/M_i)y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j,\ j = 1, ..., m$

(4) $Prob\{\sum_{i=1}^{n} p_{ij}(D_i/M_i)y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j,\ j = 1, ..., m\} \geq \gamma$

Let $(d_1^k, ..., d_n^k)$, $k = 1, ..., S$, be the samples of the demand. We construct a table with m columns (one for each machine) and $S+1$ rows ($R_{pq}^r, p = 1, ..., S+1, q = 1, ..., m$). Each element in the first S rows represents the slack in machine j for the sample corresponding to that row for the solution (y_{ij}^r, z_{ij}^r). Thus, $R_{pq}^r = C_q - \sum_{i=1}^{n} ((y_{iq}^r/M_i)d_i^p + z_{iq}^r S_{iq})$, for $p = 1, ..., S$, $q = 1, ..., m$. The $(S+1)$st row stores the slack in the complicated constraint, $C_q - \sum_{i=1}^{n} ((y_{iq}^r/M_i)\hat{D}_i + z_{iq}^r S_{iq})$. Some of the elements in the table can be negative, implying that more than the available capacity has been used up for production and setups. Note that α^r is feasible for CIP if $(S+1)$st row elements of R^r are all nonnegative, and there are at least γS other rows of R^r that have all nonnegative elements.

6.5.2 An example

Here we illustrate the procedures described above using an example of scheduling a job on two machines. Thus $n = 1$ and $m = 2$. Let the production costs per unit be $L'_{11} = 3$ and $L'_{12} = 4$ and setup cost per batch be $K_{11} = 5$ and $K_{12} = 3$. The machines are identical with a capacity of 10, the processing times and setup times are $p_{11} = p_{12} = 1$ and $S_{11} = S_{12} = 1$ respectively and $M_1 = 3$. We are interested in finding the optimal schedule for $\gamma = 0.6$ using 100 samples of demand drawn from a normal distribution with a mean of 12 and a variance of 6.

This instance corresponds to the following CIP:

$$\begin{aligned}
\text{Min } & 5z_{11}+3z_{12}+12y_{11}+16y_{12} \text{ subject to} \\
& y_{11}+y_{12}=3 \\
& y_{11}-3z_{11}\leq 0 \\
& y_{12}-3z_{12}\leq 0 \\
& 4y_{11}+z_{11}\leq 10 \\
& 4y_{12}+z_{12}\leq 10 \\
& Prob\{\tfrac{1}{3}D_1y_{11}+z_{11}\leq 10, \tfrac{1}{3}D_1y_{12}+z_{12}\leq 10\}\geq 0.6 \\
& z_{ij}\in\{0,1\}, y_{ij}\in\{0,1,2,3\}
\end{aligned}$$

Now, we use Algorithm 6.4.2 to solve the above CIP problem.

Step 1. Decompose CIP:

We decompose CIP into RIP and membership oracle. The RIP is stated as follow:

$$\begin{aligned}
\text{Min } & 5z_{11}+3z_{12}+12y_{11}+16y_{12} \text{ subject to} \\
& y_{11}+y_{12}=3 \\
& y_{11}-3z_{11}\leq 0 \\
& y_{12}-3z_{12}\leq 0 \\
& z_{ij}\in\{0,1\}, y_{ij}\in\{0,1,2,3\}
\end{aligned}$$

The membership oracle is stated as follow:

$$\begin{aligned}
& 4y_{11}+z_{11}\leq 10 \\
& 4y_{12}+z_{12}\leq 10 \\
& Prob\{\tfrac{1}{3}D_1y_{11}+z_{11}\leq 10, \tfrac{1}{3}D_1y_{12}+z_{12}\leq 10\}\geq 0.6
\end{aligned}$$

Step 2. Compute the test set $\mathcal{G}_{>_c}$ for RIP:

We convert RIP into the form of the general integer program by adding slack variables and forcing the requirement that the variables z_{ij} have a value zero or one. So, the RIP can be restated as:

$$\begin{aligned}
\text{Min } & 5z_{11}+3z_{12}+12y_{11}+16y_{12} \text{ subject to} \\
& y_{11}+y_{12}=3
\end{aligned}$$

$$
\begin{aligned}
&y_{11} - 3z_{11} + a_{11} = 0\\
&y_{12} - 3z_{12} + a_{12} = 0\\
&z_{11} + b_{11} = 1\\
&z_{12} + b_{12} = 1\\
&z_{11}, z_{12}, y_{11}, y_{12}, a_{11}, a_{12}, b_{11}, b_{12} \in \mathbb{N}
\end{aligned}
$$

Then by MGBA, we compute the reduced Gröbner basis $\mathcal{G}_{>_c}$ of RIP:

$$
\begin{aligned}
\mathcal{G}_{>_c} = \{&[(0,0,0,1,1,0,0,0),(0,0,1,0,0,1,0,0)],\\
&[(0,1,0,1,0,2,1,0),(1,0,1,0,2,0,0,1)],\\
&[(0,1,0,0,0,3,0,0),(0,0,0,0,0,0,0,1)],\\
&[(1,0,0,0,3,0,0,0),(0,0,0,0,0,0,1,0)],\\
&[(1,0,1,0,2,1,0,0),(0,0,0,1,0,0,1,0)],\\
&[(0,0,0,2,0,0,1,0),(1,0,2,0,1,2,0,0)]\}
\end{aligned}
$$

Step 3. Compute the optimal solution for RIP:

It is easy to see (1,1,1,2,2,1,0,0) is a feasible solution of RIP. So, we use $\mathcal{G}_{>_c}$ to reduce this feasible point and get the optimal point (1,0,3,0,0,0,0,1), i.e., the optimal solution of RIP

$$z_{11} = 1, z_{12} = 0, y_{11} = 3, y_{12} = 0, a_{11} = 0, a_{12} = 0, b_{11} = 0, b_{12} = 1.$$

Step 4. Derive the test set for CIP:

The test set Ω can be obtained simply by reversing the directions of all vectors in the test set $\mathcal{G}_{>_c}$ of RIP. So, we get Ω as follow.

$$
\begin{aligned}
\Omega = \{&[(0,0,1,0,0,1,0,0),(0,0,0,1,1,0,0,0)],\\
&[(1,0,1,0,2,0,0,1),(0,1,0,1,0,2,1,0)],\\
&[(0,0,0,0,0,0,0,1),(0,1,0,0,0,3,0,0)],\\
&[(0,0,0,0,0,0,1,0),(1,0,0,0,3,0,0,0)],\\
&[(0,0,0,1,0,0,1,0),(1,0,1,0,2,1,0,0)],\\
&[(1,0,2,0,1,2,0,0),(0,0,0,2,0,0,1,0)]\}
\end{aligned}
$$

Step 5. Compute the optimal solution of CIP:

First we check the feasibility of the optimal point of RIP $x^R = (1\ 0\ 3\ 0\ 0\ 0\ 0\ 1)$ for CIP. The solution is not feasible for CIP because it violates constraint $4y_{11} + z_{11} \leq 10$ in membership oracle.

Then from the point x^R, we walk on the directions of Ω to all new points. The new points can be obtained by using Ω to reduce initial point v as follows.

$v_1 = x^R$−(0,0,1,0,0,1,0,0)+(0,0,0,1,1,0,0,0)=(1,0,2,1,1,-1,0,1)
$v_2 = x^R$−(1,0,1,0,2,0,0,1)+(0,1,0,1,0,2,1,0)=(0,1,2,1,-2,2,1,0)
$v_3 = x^R$−(0,0,0,0,0,0,0,1)+(0,1,0,0,0,3,0,0)=(1,1,3,0,0,3,0,0)
$v_4 = x^R$−(0,0,0,0,0,0,1,0)+(1,0,0,0,3,0,0,0)=(2,0,3,0,3,0,-1,1)
$v_5 = x^R$−(0,0,0,1,0,0,1,0)+(1,0,1,0,2,1,0,0)=(2,0,4,-1,2,1,-1,1)
$v_6 = x^R$−(1,0,2,0,1,2,0,0)+(0,0,0,2,0,0,1,0)=(0,0,1,2,-1,-2,1,1)

We check the above points with same way as in Example 6.2.1. We prune v_1, v_2, v_4, v_5 and v_6 because they are all negative. Point v_3 is not feasible for CIP because it violates constraint $4y_{11} + z_{11} \leq 10$. But $X_0 >_c v_3$. So, we continue to walk along v_3 to get following points:

$v_{31} = v_3-(0,0,1,0,0,1,0,0)+(0,0,0,1,1,0,0,0)=(1,1,2,1,1,2,0,0)$
$v_{32} = v_3-(1,0,1,0,2,0,0,1)+(0,1,0,1,0,2,1,0)=(0,2,2,1,-2,5,1,-1)$
$v_{33} = v_3-(0,0,0,0,0,0,0,1)+(0,1,0,0,0,3,0,0)=(1,2,3,0,0,6,0,-1)$
$v_{34} = v_3-(0,0,0,0,0,0,1,0)+(1,0,0,0,3,0,0,0)=(2,1,3,0,3,3,-1,0)$
$v_{35} = v_3-(0,0,0,1,0,0,1,0)+(1,0,1,0,2,1,0,0)=(2,1,4,-1,2,4,-1,0)$
$v_{36} = v_3-(1,0,2,0,1,2,0,0)+(0,0,0,2,0,0,1,0)=(0,1,1,2,-1,1,1,0)$

Points v_{32}, v_{33}, v_{34}, v_{35} and v_{36} are negative and will be pruned. Point v_{31} is feasible for CIP and $X_0 >_c v_{31}$. By Algorithm 6.4.2, v_{31} is the optimal solution of CIP. So, the optimal schedule is:

$y_{11} = 2,\ y_{12} = 1,\ z_{11} = 1,\ z_{12} = 1, cost = 48$ and $\gamma = 0.83$.

6.5.3 Experimental results

We have implemented the algorithm for CIP by C language on a Sun Ultra Enterprise 3000 and developed a solver `CIPS` (Chance constrained IP Solver) for generic chance constrained IP problems. We also conduct a experiment for the job scheduling problems on this machine. The experiment includes five problems with respective to the different number of jobs and number of machines. For simplicity, we assume that all the number of splitting pieces M_i for the job are the same. So we use M to replace M_i for each problem. In every problem, we change the value of probability γ which cause a change in optimal solutions. There are 100 samples used in the membership oracle for each problem.

The experimental results are in Table 6.3. The first column of the table gives the problems in which n is the number of jobs, m the number of machines and M the number of splitting pieces. The second column gives the probability of production time and setup time not exceeding the capacities on the respective machines. The third column gives an optimal solution of each problem, i.e., the optimal schedules. The execution time for each problem is given in final column of the table. The timings are in CPU seconds on the Sun Ultra Enterprise 3000.

In this experiment, the problems become large as the number of jobs and number of machines increase. The largest problem with 7 jobs and 4 machines has 68 constraints and 112 variables. From the data of Table 6.3, we can see the time to answer a membership query is small, but the entire walk back procedure may talk a long time at higher values of γ. If the values of γ are lower, the optimal solution can be found relatively quickly. Also, we can see that the computation of the test set, i.e., the reduced Gröbner basis is still main difficult. The size of

Gröbner basis increase quickly and the computation for it becomes expensive as the problems become large. So, we need to improve the efficiency of our algorithm MGBA in future.

Problems	γ	Optimal schedule	Time (sec.)
$n=1$ $m=2$ $M=3$	0.83	$y_{11}=1, y_{12}=2.$ cost=48	0.07
	0.9	No optimal solution	0.07
$n=2$ $m=3$ $M=2$	0.84	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=2, y_{23}=0.$ cost=44	2.36
	0.86	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=1, y_{23}=1.$ cost=55	2.89
	0.88	$y_{11}=0, y_{12}=1, y_{13}=1, y_{21}=0, y_{22}=1, y_{23}=1.$ cost=97	182.98
$n=4$ $m=3$ $M=2$	0.5	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=2, y_{23}=0,$ $y_{31}=2, y_{32}=0, y_{33}=0, y_{41}=2, y_{42}=0, y_{43}=0.$ cost=93	35.59
	0.62	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=1, y_{23}=1,$ $y_{31}=2, y_{32}=0, y_{33}=0, y_{41}=2, y_{42}=0, y_{43}=0.$ cost=104	45.71
$n=6$ $m=3$ $M=2$	0.5	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=2, y_{23}=0,$ $y_{31}=2, y_{32}=0, y_{33}=0, y_{41}=2, y_{42}=0, y_{43}=0,$ $y_{51}=2, y_{52}=0, y_{53}=0, y_{61}=0, y_{62}=2, y_{63}=0.$ cost=122	433.64
	0.62	$y_{11}=0, y_{12}=2, y_{13}=0, y_{21}=0, y_{22}=1, y_{23}=1,$ $y_{31}=2, y_{32}=0, y_{33}=0, y_{41}=2, y_{42}=0, y_{43}=0,$ $y_{51}=2, y_{52}=0, y_{53}=0, y_{61}=0, y_{62}=2, y_{63}=0.$ cost=122	1219.18
$n=7$ $m=4$ $M=2$	0.42	$y_{11}=0, y_{12}=2, y_{13}=0, y_{14}=0, y_{21}=0, y_{22}=2,$ $y_{23}=0, y_{24}=0, y_{31}=2, y_{32}=0, y_{33}=0, y_{34}=0,$ $y_{41}=2, y_{42}=0, y_{43}=0, y_{44}=0, y_{51}=2, y_{52}=0,$ $y_{53}=0, y_{54}=0, y_{61}=0, y_{62}=0, y_{63}=0, y_{64}=2,$ $y_{71}=0, y_{72}=0, y_{73}=0, y_{74}=2.$ cost=121	20824.87
	0.47	$y_{11}=0, y_{12}=2, y_{13}=0, y_{14}=0, y_{21}=0, y_{22}=2,$ $y_{23}=0, y_{24}=0, y_{31}=2, y_{32}=0, y_{33}=0, y_{34}=0,$ $y_{41}=2, y_{42}=0, y_{43}=0, y_{44}=0, y_{51}=0, y_{52}=0,$ $y_{53}=0, y_{54}=2, y_{61}=0, y_{62}=0, y_{63}=2, y_{64}=0,$ $y_{71}=0, y_{72}=2, y_{73}=0, y_{74}=0.$ cost=124	21072.43

Table 6.3: Experimental results

Chapter 7

Conclusion

The research of IP has being carried out for 50 years and many numerical algorithms for IP have been proposed. However, solving IP based on symbolic computation, i.e., Gröbner basis technique and applying this technique to solve stochastic IP is very new. This thesis studies IP problems from modelling with logic to solving with Gröbner bases, from general IP problems to real practical complex IP problem namely chance constrained IP problems, and combines the modelling tool and several IP solvers to give an efficient solution for real practical complex IP problems. Here, we summarise the contributions of this thesis and discuss the future works as follows.

7.1 Contributions

The research contributions of this thesis contain theoretical and practical aspects.

1. Theoretical contributions:

   (1) introduced a logical modelling language $\mathcal{L}^+$, by which, we model real world IP problems conveniently and efficiently.

   (2) proposed a systematic method for translating formulas in $\mathcal{L}^+$ to IP formulas, whose transformation algorithm is rigorously designed, verified and implemented in Mathematica 3.0.

   (3) designed a new algorithm *Minimised Geometric Buchberger Algorithm* (MGBA) for IP, which provides more significant performance improvement than other symbolic algorithms for IP.

   (4) introduced a new approach for chance constrained IP based on MGBA, which is the first symbolic method for solving generic chance constrained IP problems.

2. Practical contributions:

(1) developed an IP modelling system `TIP` (Transformation of Integer Programming), which is connected via Mathlink and CGI to IP-solvers such as CPLEX or solvers that we have developed.

(2) developed an IP solver `MGBS` (Minimised Geometric Buchberger Solver), which runs on a Sun Ultra Enterprise 3000.

(3) developed a solver `CIPS` (Chance constrained IP Solver) for generic chance constrained IP, which runs on a Sun Ultra Enterprise 3000.

(4) applied `CIPS` to solve a real practical stochastic IP problem namely job scheduling up to relatively large size with seven jobs and four machines.

7.2 Future work

The objective of our research on IP is to solve real world IP problems. The real world IP problems have two characteristics:

(a) Complexity

IP problems are concerned with many fields of science and technology, as well as our daily life. In practical application, IP has more complex formulations. There are IP forms with multi-objective function and IP forms with stochastic variables such as probabilistic constraints, two-stage stochastic constraints, and multi-stage stochastic constraints.

(b) Large scale

Some real world IP problems are too large to be solved by all existing algorithms. They have tens of thousands variables and thousands constraints and their optimal solutions can not be computed until now.

Based on the above characteristics of real world IP problems, our future work is as follows.

1. We will refine and enhance the logical modelling system by introducing constructs for specifying uncertainty. The translation algorithm will be further developed to support dealing with stochastic variables. Also, we will concentrate on the development of MGBA for solving more complex stochastic integer program problems such as two-stage or multi-stage stochastic IP problems.

2. Due to the complexity of computation, we will explore the inherent parallelism of MGBA and build up parallel implementation of this algorithm. We have a parallel implementation of Buchberger algorithm for IP [55]. So, we can hope that based on this parallel implementation, we can develop an efficient and powerful parallel solver for large scale stochastic IP problems in future.

Appendix

■ This appendix gives all the program of TIP.

Function **ΓFormQ**[x] checks whether an expression is Γ–form .

```
ΓFormQ[Γ[_, _]] := True;
ΓFormQ[_] := False;
```

Function **VarQ**[x] checks whether x is a propositional variable or an integer variable.

```
VarQ[x_Symbol] /; x =!= True ∧ x =!= False := True;
VarQ[_] := False;
```

Function **NegPVarQ**[x] checks whether x is a negated propositional variable.

```
NegPVarQ[¬ x_Symbol] /; x =!= True ∧ x =!= False := True;
NegPVarQ[_] := False;
```

Function **PLiteralQ**[p] checks whether p is a propositional variable or a negated propositional variable

```
PLiteralQ[p_] := VarQ[p] ∨ NegPVarQ[p];
NotPLiteralQ[p_] := ¬ PLiteralQ[p]
```

Function **BoundedQ**[x] checks whether a variable x is properly bounded.

```
BoundedQ[m_Integer ≤ x_?VarQ ≤ n_Integer] := True;
BoundedQ[_] := False;
```

Function **ConstraintQ**[c] checks whether an expression c is a constraint . Note that we use == to denote the equality of symbol of L^*.

```
ConstraintQ[x_ ≥ y_ | x_ > y_ | x_ ≤ y_ | x_ < y_ | x_ == y_ ] := True;
ConstraintQ[_] := False;
```

Function **PredOrder**[x,y] defines a predicate order in a Γ–formula.

```
PredOrder[x_?VarQ, r_[_, _]] := True;
PredOrder[Not[_], r_[_, _]] := True;
PredOrder[x_, y_] := False
```

Function **PLiteral2IPvar**[e] translates a propositional variable e= p and a formula e=¬p into IP–formulas.

```
PLiteral2IPvar[x_?VarQ] := Bounds[[Position[Bounds, x][[1, 1]], 2]] ;
PLiteral2IPvar[¬ x_?VarQ] := 1 - Bounds[[Position[Bounds, x][[1, 1]], 2]] ;
```

Function **Raux**[x,y,z] is an auxiliary function of **RuleR**[x].

```
Raux[{x___?PLiteralQ, y___?NotPLiteralQ}, m_, n_] :=
 Module[{newvarlist = Map[Unique[d]&, {y}]}, Bounds = Bounds⋃Map[0 ≤ # ≤ 1&, newvarlist];
  Prepend[Map[RuleR, MapThread[Function[{d, a}, d > 0 ⇒ a], {newvarlist, {y}}]],
   (Apply[Plus, newvarlist⋃Map[PLiteral2IPvar, {x}]] ≥ m n)]];
```

Function **RuleE**[x] defines rule E.

```
RuleE[x_] := x //. {p_ ⇒ q_ -> ¬ p ∨ q, p_ ⇔ q_ -> (p ⇒ q) ∧ (q ⇒ p)};
```

Function **RuleT**[x] defines rule T.

```
RuleT[x1_ ∨ x2_] := Γ[1, {RuleT[x1], RuleT[x2]}];
RuleT[x1_ ∧ x2_] := Γ[2, {RuleT[x1], RuleT[x2]}];
RuleT[¬ x_] := ¬ RuleT[x];
RuleT[atleast[m_, S_]] := Γ[m, Map[RuleT, S]];
RuleT[atmost[m_, S_]] := Γ[Length[S] - m, Map[RuleT, Map[Not, S]]];
RuleT[none[S_]] := Γ[Length[S], Map[RuleT, Map[Not, S]]];
RuleT[x_?VarQ] := PVarDecl[x];
RuleT[x_?ConstraintQ] := IPVarDecl[x];
RuleT[True] := Throw[{True, Bounds}];
RuleT[False] := Throw["Wrong specification"]
```

Function **RuleN**[x] defines rule N.

```
RuleN[Γ[m_, S_]] := Γ[m, Map[RuleN, S]];
RuleN[¬ Γ[m_, S_]] := Γ[Length[S] - m + 1, Map[RuleN, Map[Not, S]]];
RuleN[¬ (x1_ == x2_)] := Γ[1, {x1 > x2, x1 < x2}];
RuleN[x_] := x;
```

Function **RuleF**[x] defines rule F.

```
RuleF[x_] := x //. {RF1, RF2}
RF1 = Γ[1, {x1___, Γ[1, {x2___}], x3___}] :> Γ[1, {x1, x2, x3}];
RF2 = Γ[m_, {x1___, Γ[n_, S_], x2___}] /; m == Length[{x1, x2}] + 1 ∧ n == Length[S] :>
  Γ[n + m - 1, S ⋃ {x1, x2}];
```

Function **RuleR**[x] defines rule R.

```
RuleR[r_[x1_, x2_]] := r[x1, x2];
RuleR[p_?PLiteralQ] := PLiteral2IPvar[p] ⩾ 1;
RuleR[Γ[m_, S_]] /; m == Length[S] := Map[RuleR, S];
RuleR[Γ[m_, S_]] /; m < Length[S] := Raux[Sort[S, PredOrder], m, 1];
RuleR[d_ > 0 ⇒ Γ[m_, S_]] /; m == Length[S] := Map[RuleR, Map[(d > 0 ⇒ #)&, S]];
RuleR[d_ > 0 ⇒ Γ[m_, S_]] /; m < Length[S] := Raux[Sort[S, PredOrder], m, d];
RuleR[d_ > 0 ⇒ p_?PLiteralQ] := PLiteral2IPvar[p] ⩾ d;
RuleR[d_ > 0 ⇒ x1_ ⩾ x2_] := Module[{L}, (x1 - x2 ⩾ L (1 - d))_L];
RuleR[d_ > 0 ⇒ x1_ > x2_] := Module[{L}, (x1 - x2 > (L - 1) (1 - d))_L];
RuleR[d_ > 0 ⇒ x1_ ⩽ x2_] := Module[{U}, (x1 - x2 ⩽ U (1 - d))^U];
RuleR[d_ > 0 ⇒ x1_ < x2_] := Module[{U}, (x1 - x2 < (U + 1) (1 - d))^U];
RuleR[d_ > 0 ⇒ x1_ == x2_] := {RuleR[d > 0 ⇒ x1 ⩾ x2], RuleR[d > 0 ⇒ x1 ⩽ x2]};
```

Function **Lowerbound**[A] computes the lowerbound of constraint A with respects to the given Bounds, where Bounds is specifed as a list of terms of the form $a \leq x \leq b$. **Lowerbound** uses auxiliary function **Lb**.

```
Lowerbound[A_?AtomQ] := Lb[A];
Lowerbound[A_] := Map[Lb, A];
Lb[term : a_ * x_] /; a > 0 :=
 Module[{i = Position[Bounds, _ ≤ x ≤ _][[1, 1]]}, term /. {Bounds[[i, 2]] → Bounds[[i, 1]]}];
Lb[term : a_ * x_] /; a < 0 :=
 Module[{i = Position[Bounds, _ ≤ x ≤ _][[1, 1]]}, term /. {Bounds[[i, 2]] → Bounds[[i, 3]]}];
Lb[x_?VarQ] := Bounds[[Position[Bounds, _ ≤ x ≤ _][[1, 1]], 1]]
Lb[a_] := a
```

Likewise function **Upperbound** is defined.

```
Upperbound[A_] := Map[Ub, A];
Upperbound[A_?AtomQ] := Ub[A];
Ub[term : a_ * x_] /; a > 0 :=
 Module[{i = Position[Bounds, _ ≤ x ≤ _][[1, 1]]}, term /. {Bounds[[i, 2]] → Bounds[[i, 3]]}];
Ub[term : a_ * x_] /; a < 0 :=
 Module[{i = Position[Bounds, _ ≤ x ≤ _][[1, 1]]}, term /. {Bounds[[i, 2]] → Bounds[[i, 1]]}];
Ub[x_?VarQ] := Bounds[[Position[Bounds, _ ≤ x ≤ _][[1, 1]], 3]];
Ub[a_] := a;
```

Function **RRC**[x] (Remove Redundant Constraints) is defined below using the following auxiliary functions: **RCaux** checks whether a given constraint is redundant and outputs {} if redundant or else a simplified constraint after substituting the value of the Bounds.

```
RRC[A_] := Map[RCaux, A];
RCaux[ (A_ ≤ B_)^U_ ] := Module[{Uw = Upperbound[A]},
  If[Uw > 0, A ≤ (B /. {U → Uw}), {}]];
RCaux[ (A_ < B_)^U_ ] :=
 Module[{Uw = Upperbound[A]},
  If[Uw ≥ 0, A < (B /. {U → Uw}), {}]];
RCaux[ (A_ ≥ B_)_L_ ] := Module[{Lw = Lowerbound[A]},
  If[Lw < 0, A ≥ (B /. {L → Lw}), {}]];
RCaux[ (A_ > B_)_L_ ] :=
 Module[{Lw = Lowerbound[A]},
  If[Lw ≤ 0, A > (B /. {L → Lw}), {}]];
RCaux[A_] := A
```

Function **MakeBounds**[x] checks whether the input bound of each integer variable is right and generates a decision variable and its bound for each propositional variable.

```
MakeBounds[ x_?BoundedQ ] := x;
MakeBounds[x_?VarQ] := 0 ≤ Pvar_x ≤ 1;
```

Founction **PVarDecl**[e] checks whether each propositional variable is properly declared.

```
PVarDecl[v_] := If[MemberQ[Bounds, v, {2, 3}],
  v, Print["Propositional Variable ", v, " not declared."];
  Throw[Bounds]];
```

Founction **IPVarDecl**[e] checks whether each integer variable is properly declared.

```
IPVarDecl[v_?VarQ] :=
 If[MemberQ[Bounds, v, 2], v, Print["Integer variable ", v, " not declared."];
  Throw[Bounds]];
IPVarDecl[t : f_[___]] := Map[IPVarDecl, t];
IPVarDecl[x_Integer] := x;
IPVarDecl[x_] :=
 Module[{}, Print["Constraint ", x, " is not well formed"]; Throw[Bounds]]
```

Founction **IneqSimplify**[e] simplifies expressions.

```
IneqSimplify[a_ > b_] := Simplify[a - b] > 0;
IneqSimplify[a_ ≥ b_] := Simplify[a - b] ≥ 0;
IneqSimplify[a_ < b_] := Simplify[a - b] < 0;
IneqSimplify[a_ ≤ b_] := Simplify[a - b] ≤ 0;
```

Founction **Options**[GenIP] specifies that the generated IP-formulas will be simplified.

```
Options[GenIP] = {IPFormSimplify -> True}
```

This is the main function **GenIP**. **GenIP** needs three arguments **spec**, **decl** and **opts**. **spec** is a logical specification, **decl** is a list of propositional variables and the bounds of integer variables of the form $a \le x \le b$, and **opts** is an option to specify whether the generated IP-formulas will be simplified or not with **IPFormSimplify->True** or **IPFormSimplify->False**.

```
GenIP[spec_, decl_, opts___] :=
 Catch[Block[{Pvar = Unique[δ] , Bounds = Map[MakeBounds, decl]} ,
  With[{ipform =
      Flatten[{Composition[RRC, Flatten, RuleR, RuleF, RuleN, RuleT, RuleE][spec]}]},
    If[IPFormSimplify /. {opts} /. Options[GenIP],
     Map[IneqSimplify, ipform], ipform, ipform] ⋃
    Bounds]]]
```

Bibliography

[1] W. W. Adams and P. Loustaunau, *An Introduction to Gröbner bases.* American Mathematical Society, volume 3, 1994.

[2] B. Bank and R. Mandel, *Parametric integer optimization.* Akademie-Verlag, Berlin, 1988.

[3] V. V. Batyrev, *On the classification of smooth projective toric varieties.* Tohoku Mathematical Journal 43:569-585, 1991.

[4] D. Bayer and I. Morrison, *Gröbner bases and geometric invariant theory.* Journal of Symbolic Computation 6:209–217, 1988.

[5] D. Bayer and M. Stillman, *On the complexity of computing syzygies.* Journal of Symbolic Computation 6:135–147, 1988.

[6] J. E. Beasley, *Advances in Linear and Integer Programming.* Oxford Science Publications, 1996.

[7] T. Becker and V. Weispfennig, *Gröbner bases: a computational approach to commutative algebra.* Springer-Verlag, Berlin, 1993.

[8] L. J. Billera and S. Sturmfels, *Fiber polytopes.* Annals of Mathematics 135:527–549, 1992.

[9] R. E. Bixby, *Implementing the simplex method: the initial basis.* ORSA Journal on Computing, 4:267–284, 1992.

[10] E. Boros and A. Prékopa, *Closed-form two-sided bounds for probabilities that at least r and exactly r out of n events occur.* Mathematical Operations Research 14:317–342, 1989.

[11] A. L. Brearley, G. Mitra, H. P. Williams, *Analysis of mathematical programming problems prior to applying the simplex algorithm.* Mathematical programming 8:54-83, 1975.

[12] G. G. Brown and M. P. Olson, *Dynamic factorization in large scale optimization.* Technical Report NPSOR-93-008, Naval Postgraduate School, Monterey, California, 1993.

[13] B. Buchberger, *A criterion for detecting unnecessary reductions in the construction of Gröbner bases.* LNCS 72, pp. 3–21, 1979.

[14] B. Buchberger, *Gröbner bases: an algorithm method in polynomial ideal theory.* in Multidimensional Systems Theory (N.K.Bose ed.), Reidel, Dordrecht, pp. 184-232, 1985.

[15] B. Buchberger and J. Elias, *Using Gröbner bases for detecting polynomial identities: a case study on fermat's ideal.* Journal of Number Theory 41:272–279, 1992.

[16] B. Buchberger and F. Winkler (ed.), *Gröbner bases and applications.* Cambridge University Press, 1998.

[17] T. M. Cavalier, P. M. Pardalos and A. L. Soyster, *Modelling and integer programming techniques applied to propositional calculus.* Computer and Operations Research, 17(6):561–570, 1990.

[18] P. Conti and C.Traverso, *Buchberger algorithm and integer programming.* Proceedings AAECC-9, New Orleans, pp. 130–139, LNCS 539 Springer-Verlag.

[19] W. Cook, A. M. H. Gerards, A. Schrijver and E. Tardos, *Sensitivity theorems in integer linear programming.* Mathematical Programming 34:251–264, 1986.

[20] D. Cox, J. Little and D. O'Shea *Ideals, varieties and algorithms.* Springer, New York, 1992.

[21] *Using the CPLEX Callable Library.* CPLEX Optimization, Inc. USA, 1995.

[22] F. DiBiase and R. Urbanke, *An algorithm to calculate the kernel of certain polynomial ring homomorphisms.* Experimental Mathematics 4:227–234, 1995.

[23] E. Domenjoud, *Solving systems of linear Diophantine equations: An algebraic approach.* Journal of Symbolic Computation 8:173–182, 1989.

[24] M. A. Duran and L. F. Grossmann, *An outer approximation algorithm for a class of mixed integer nonlinear programs.* Mathematical Programming 36:307–339, 1986.

[25] J. Eckstein, *Parallel branch-and-bound algorithms for general mixed-integer programming on the CM-5.* Technical Report TMC-257, Mathematical Sciences Research Group, 1993.

[26] D. Eisenbud, *Commutative algebra with a view toward algebraic geometry.* Number 150 in Graduate Texts in Mathematics, Springer-Verlag, New York, 1994.

[27] Y. Ermoliev and R. J-B. Wets, *Numerical techniques for stochastic optimization.* Springer-Verlag, Berlin, 1988.

[28] G. Ewald, *Combinatorial convexity and algebraic geometry.* Number 168 in Graduate Texts in Mathematics, New York, 1996.

[29] W. Fulton, *Introduction to toric varieties.* Number 131 in Annals of Mathematics Studies, Princeton University Press, Princeton, 1993.

[30] J. H. Gallier, *Logic for Computer Science: Foundations of Automatic Theorem Proving.* John Wiley & Sons, Inc. New York, USA, 1987.

[31] R. Gebauer and H. M. Möller, *On an installation of Buchberger's algorithm.* Journal of Symbolic Computation 6:275–286, 1988.

[32] A. Geoffrion, *Introduction to structured modelling.* Management Science, 33:547–588, 1987.

[33] P. Gianni, *Properties of Gröbner bases under specializations.* LNSC 358, pp. 293–297, 1987.

[34] P. Gianni, B. Trager and G. Zacharias, *Gröbner bases and primary decomposition of polynomial ideals.* Journal of Symbolic Computation 6:149–167, 1988.

[35] A. Giovini, T. Mora, G. Niesi, L. Robbiano and C. Traverso, *"One sugar cube, please" or selection strategies on the Buchberger algorithm.* Proceedings of the International Symposium on Symbolic and Algebraic Computation, pp. 49–54, 1991.

[36] A. Giovini, G. Niesi and E. Armando, *CoCoA: An experimental system for computer and commutative algebra.* version 1.5. Department of Mathematics, University of Genova, 1991.

[37] J. E. Graver,*On the foundations of linear and integer programming I.* Mathematical Programming 8:207-226, 1975.

[38] D. Grayson and M. Stillman *MACAULAY 2: Acomputer algebra system.* Available from http://www.math.uiuc.edu/~dan.

[39] M. Grötschel, L. Lovasz and A. Schrijver, *Geometric algorithms and combinatorial optimization.* Springer-Verlag, Berlin, Germany, 1988.

[40] E. Hadiconstantinou and G. Mitra, *A linear and discrete programming framework for representing qualitative knowledge.* Journal of Economic Dynamics and Control 18:273–297, 1994.

[41] R. Hartshorne, *Algebraic geometry.* Springer-Verlag 1977.

[42] A. Heck, *Introduction to Maple, a computer algebra system* . Springer-Verlag, 1993.

[43] J. L. Hein, *Discrete Structures, Logic, and Computability.* Jones and Bartlett Publishers, London, 1995.

[44] J. N. Hooker, *A quantitative approach to logical inference.* Decision Support Systems 4:45–69, 1988.

[45] S. Hosten and B. Sturmfels, *Grin: An implementation of Gröbner bases for integer programming.* In Balas, E., Clausen, J., editors, Integer Programming and Combinatorial Optimization, pp. 207–276, LNCS 920 Springer-Verlag.

[46] F. Hreinsdottir, *A case where choosing a product order makes the calculation of a Gröbner basis much faster.* Journal of Symbolic Computation 18: 373–378, 1994.

[47] R. G. Jeroslow, *Computation-oriented reduction of predicate to propositional logic.* Decision Support Systems 4:183–197, 1988.

[48] R. G. Jeroslow, *Logic-based decision support: Mixed integer programming model frmulation.* North Holland, 1989.

[49] D. Kahneman and A. Tversky, *Rational choice and the framing of decision making.* The Journal of business, 59(2):251–278, 1986.

[50] P. Kall and S. W. Wallace, *Stochastic Programming.* John Wiley & Sons Ltd, Chichester, 1994.

[51] A. Kaufmann and A. Henry-Labordere, *Integer and Mixed Programming: Theory and Applications.* Academic Press, Inc. New York, 1977.

[52] S. E. Kimbrough and R. M. Lee, *Logic modelling: A tool for management science.* Decision Support Systems 4:3–16, 1988.

[53] P. Kotler, *Marketing management: Analysis, planning and control.* Prentice Hall, 1980.

[54] Q. Li, F. Janssen, Z. F. Yang and T. Ida, *ILIN: an implementation of the integer labeling algorithm for integer programming.* IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences. Vol. E81-A No. 2 pp. 304–309, 1998.

[55] Q. Li, Y. K. Guo and T. Ida, *A parallel algebraic approach towards integer programming.* Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems, Washington,D.C., U.S.A. pp. 59–64, 1997.

[56] Q. Li, Y. K. Guo, T. Ida and J. Darlington, *The minimised geometric Buchberger algorithm: An optimal algebraic algorithm for integer programming.* Proceedings of the International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii U.S.A. pp. 331–338, 1997.

[57] B. Mareschal, *Stochastic multicriteria decision making and uncertainty.* European Journal of Operational Research 26:58–64, 1986.

[58] S. V. Maturana, *Issues in the design of modeling languages for mathematical programming.* European Journal of Operational Research 72:243–261, 1994.

[59] E. Mendelson, *Introduction to Mathematical Logic.* Van Nostrand Princeton, 4th ed. 1997.

[60] J. E. Mitchell and M. J. Todd, *Solving combinatorial optimization problems using Karmarkar's algorithm.* Mathematical Programming 56:245–284, 1992.

[61] G. Mitra, C. Lucas, S. Moody and E. Hadijiconstrantinou, *Tools for reformulating logical forms into zero-one mixed integer programs.* European Journal of Operational Research 72:262–276, 1994.

[62] K. I. M. McKinnon and H.P.Williams, *Constructing Integer Programming Models by The Predicate Calculus.* Annals of Operations Research 21:227-246, 1989.

[63] T. Mora and L. Robbiano, *Gröbner fan of an ideal.* Journal of Symbolic Computation 6:183–208, 1988.

[64] R. M. Nauss, *Bond portfolio analysis using integer programming.* in Financial Optimisation by Cambridge University Press, pp. 260–287, 1993.

[65] G. L. Nemhauser and L. A. Wolsey *Integer and combinatorial optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, 1988.

[66] T. Oda, *Convex bodies and algebraic geometry: An introduction to the theory of toric varieties.* Springer-Verlag, New York, 1985.

[67] R. G. Parker and R. L. Rardin, *Discrete Optimization.* Academic Press, San Diego, CA92101, 1988.

[68] R. Raman and I. E. Grossman, *Modelling and computational techniques for logic based integer programming.* Proceedings of AIChE Annual Meeting, Miami, pp. 1–36, 1992.

[69] R. Raman and I. E. Grossmann, *Relation between MILP modelling and logical inference for process synthesis.* Computers and chemical engineering 15(2):73-84,1991.

[70] R. Raman and I. E. Grossmann, *Symbolic integration of logic in mixed-integer programming techniques for process synthesis.* Computers and Chemical Engineering 17(9):909–927, 1993.

[71] J. Romeuf, *A polynomial algorithm for solving systems of two linear Diophantine equations.* Theoretical Computer Science 74:329–340, 1990.

[72] B. Roy, *Decision aid and decision making.* in Multiple Criteria Decision Aid, Springer-Verlag: pp. 17–35, 1990.

[73] H. E. Scarf, *Neighborhood systems for production sets with indivisibilities.* Econometrica 54:507–522, 1986.

[74] A. Schrijver, *Theory of Linear and Integer Programming* . John Wiley & Sons Ltd, 1986.

[75] R. Schultz, L. Stougie and M. H. Vlerk, *Solving stochastic programs with complete integer recourse: a framework using Gröbner bases.* Technique Report SC 95-23, Konrad-Zuse-Zentrum, Berlin, 1995.

[76] D. Shannon and M. Sweedler, *Using Gröbner bases to determine algebra membership, split surjective algebra homeomorphisms determine birational equivalence.* Journal of Symbolic Computation 6:267–273, 1988.

[77] G. Sierksma, *Linear and Integer Programming.* Marcel Dkker, Inc. New York, 1996.

[78] B. Sturmfels, *Algorithm in invariant theory.* RISC Series in Symbolic Computation, Springer-Verlag, New York, 1993.

[79] B. Sturmfels *Gröbner Bases and Convex Polytopes.* American Mathematical Society, volume 8. 1996.

[80] B. Sturmfels, *Gröbner bases of toric varieties.* Tohoku Mathematical Journal 43:249-261, 1991.

[81] B. Sturmfels and N. White, *Computing combinatorial decompositions of rings.* Combinatorica 11:275–293, 1991.

[82] H. A. Taha, *Integer programming theory, applications, and computations.* Academic Press, Inc. New York, USA, 1975.

[83] S. R. Tayur, R. R. Thomas and N. R. Natrj, *An algebraic geometry algorithm for scheduling in presence of setups and correlated demands.* Mathematical Programming 69:369-401, 1995.

[84] R. R. Thomas, *A geometric Buchberger algorithm for integer programming.* Mathematics of Operations Research 20:864–884, 1995.

[85] R. R. Thomas, J. de Loera and B. Sturmfels, *Gröbner bases and triangulations of the second hypersimplex.* Combinatorica 15:409–424, 1995.

[86] R. R. Thomas and B. Sturmfels, *Variation of cost functions in integer programming.* Mathematical Programming 77:357–387, 1997.

[87] R. R. Thomas and R. Weismantel, *Truncated Gröbner bases for integer programming.* Applicable Algebra in Engineering, Communication and Computing 8:241–257, 1997.

[88] R. R. Thomas and R. Weismantel, *Test sets and inequalities for integer programs.* Proceedings of the 5th International IPCO conference, Vancouver, LNCS 1084, pp. 16-30, 1996.

[89] R. Urbaniak, R. Weismantel and G. Ziegler, *A variant of Buchberger's algorithm for integer programming.* SIAM Journal on Discrete Mathematics 10:96–108.

[90] P. M. Vaidya, *A new algorithm for minimizing convex functions over convex sets.* Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, pp. 338–343, 1989.

[91] S. W. Wallace and R. J-B. Wets, *Preprocessing in stochastic programming: the case of linear programs.* ORSA Journal on Computing, 4:45–59, 1992.

[92] V. Weispfenning, *Comprehensive Gröbner bases.* Journal of Symbolic Computation 14:1–29, 1992.

[93] S. Walukiewicz, *Integer Programming.* Kluwer Academic Publishers, 1991.

[94] H. P. William, *Computational logic and integer programming: connections between the methods of logic, AI and OR.* Proceedings of Symposium on Applied Mathematical Modelling and Optimisation, Brunel University, pp. 1–25, 1991.

[95] H. P. William, *Linear and integer programming applied to the propositional calculus.* Journal of Systems Research and Information Science 2:81–100, 1987.

[96] H. P. William, *Model building in mathematical programming.* Wiley, New York, 1985.

[97] J. M. Wilson, *Compact normal forms in propositional logic and integer programming formulations.* Computers and Operations Research, 17(3):309–314, 1990.

[98] S. Wolfram, *The Mathematica Book.* WolframMedia Inc. Champaing, Illinois, 1996.

Index

List of Notations

Acronyms

Symbols