Market Split and Basis Reduction: Towards a Solution of the Cornuéjols-Dawande Instances

K. Aardal* R. E. Bixby† C. A. J. Hurkens‡ A. K. Lenstra§
J. W. Smeltink¶

Abstract

At the IPCO VI conference Cornuéjols and Dawande proposed a set of 0-1 linear programming instances that proved to be very hard to solve by traditional methods, and in particular by linear programming based branch-and-bound. They offered these *market split* instances as a challenge to the integer programming community. The market split problem can be formulated as a system of linear diophantine equations in 0-1 variables.

In our study we use the algorithm of Aardal, Hurkens, and Lenstra (1998) based on lattice basis reduction. This algorithm is not restricted to deal with market split instances only but is a general method for solving systems of linear diophantine equations with bounds on the variables.

We show computational results from solving both feasibility and optimization versions of the market split instances with up to 7 equations and 60 variables, and discuss various branching strategies and their effect on the number of enumerated nodes. To our knowledge, the largest feasibility and optimization instances solved before had 6 equations and 50 variables, and 4 equations and 30 variables respectively.

We also present a probabilistic analysis describing how to compute the probability of generating infeasible market split instances. By generating instances the way prescribed by Cornuéjols and Dawande, one obtains relatively many feasible instances for sizes larger than 5 equations and 40 variables.

---

*`aardal@cs.uu.nl`. Department of Computer Science, Utrecht University. Research partially supported by the ESPRIT Long Term Research Project nr. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*), by the project TMR-DONET nr. ERB FMRX-CT98-0202, both of the European Community, and by NSF through the Center for Research on Parallel Computation, Rice University, under Cooperative Agreement No. CCR-9120008.

†`bixby@caam.rice.edu`. Department of Computational and Applied Mathematics, Rice University.

‡`wscor@win.tue.nl`. Department of Mathematics and Computing Science, Eindhoven University of Technology. Research partially supported by the project TMR-DONET nr. ERB FMRX-CT98-0202 of the European Community.

§`arjen.lenstra@citicorp.com`. Emerging Technology, Citibank N.A.

¶`job@cs.uu.nl`. Department of Computer Science, Utrecht University. Research partially supported by the ESPRIT Long Term Research Project nr. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*), and by the project TMR-DONET nr. ERB FMRX-CT98-0202, both of the European Community.

1 Introduction and Problem Description

The feasibility version of the market split problem is described as follows. A company with two divisions supplies retailers with several products. The goal is to allocate each retailer to one of the divisions such that division 1 controls a fraction f_i, $0 \leq f_i \leq 1$, of the market for product i, and division 2 controls $1 - f_i$. There are n retailers and $m \leq n$ products. Let a_{ij} be the demand of retailer j for product i, and let d_i be determined as $\lfloor f_i d'_i \rfloor$, where d'_i is the total amount of product i that is supplied to the retailers. The decision variable x_j takes value 1 if retailer j is allocated to division 1 and 0 otherwise. The question is: "does there exist an allocation of the retailers to the divisions such that the desired market split is obtained?" One can formulate this feasibility problem (FP) mathematically as follows:

$$\textbf{FP}\text{: does there exist a vector } \boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{d}, \quad \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1}? \tag{1}$$

The market split problems was first published in the book by Williams [13]. There, the application was related to the oil market in the UK, and the constraints were slightly more involved compared to the description given above.

Let $X = \{\boldsymbol{x} \in \{0,1\}^n : \sum_{j=1}^n a_{ij}x_j = d_i,\ 1 \leq i \leq m\}$. Problem FP (1) is NP-hard due to the bounds on the variables. The algorithm that we use was developed for the more general problem:

$$\text{does there exist a vector } \boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{d}, \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}? \tag{2}$$

We assume that $\boldsymbol{A}$ is an integral $m \times n$ matrix, where $m \leq n$, $\boldsymbol{d}$ is an integral m-vector, and $\boldsymbol{l}$ and $\boldsymbol{u}$ are integral n-vectors. We denote the ith row of the matrix $\boldsymbol{A}$ by $\boldsymbol{a}_i$. Without loss of generality we assume that $\gcd(a_{i1}, a_{i2}, ..., a_{in}) = 1$ for $1 \leq i \leq m$, and that $\boldsymbol{A}$ has full row rank.

In the optimization version (OPT) of the market split problem we want to find the minimum slack, positive or negative, that needs to be added to the diophantine equations in order to make the system feasible:

$$\textbf{OPT}\text{: } \min\{\sum_{i=1}^{m}(s_i + w_i) \ \text{ s.t. } (\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{w}) \in X^S\}, \tag{3}$$

where $X^S = \{(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{w}) : \sum_{j=1}^n a_{ij}x_j + s_i - w_i = d_i,\ 1 \leq i \leq m,\ \boldsymbol{x} \in \{0,1\}^n,\ \boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_+^m\}$.

The Cornuéjols-Dawande instances [2] of the market split problem were generated such that they were hard for linear programming based branch-and-bound, and they appear to be hard for several other methods as well. The input was generated as follows. For given m, let $n = 10(m-1)$ and let the coefficients a_{ij} be integer numbers drawn uniformly and independently from the interval $[0, D-1]$, where $D = 100$. The right-hand-side coefficients are computed as $d_i = \lfloor \frac{1}{2} \sum_{j=1}^n a_{ij} \rfloor$, $1 \leq i \leq m$. This corresponds to a market split where $f_i = \frac{1}{2}$ for $1 \leq i \leq m$. Cornuéjols and Dawande [2] argued that with this choice of data, most of the instances of the feasibility problem (1) are infeasible, which implies that the optimization variant (3) has an objective value greater than zero. If branch-and-bound is used to solve OPT (3), then, due to the symmetry of the input, the value of the LP-relaxation remains at zero even after many variables have been fixed. A behavior of branch-and-bound that was observed by Cornuéjols and Dawande was that $2^{\alpha n}$ nodes needed to be evaluated, where α typically takes values between 0.6 and 0.7.

The algorithm we use in our study is described briefly in Section 2. The algorithm was developed by Aardal, Hurkens, and Lenstra [1] for solving a system of linear diophantine equations with bounds on the variables, such as problem (2), and is based on Lovász' lattice basis reduction algorithm as described by Lenstra, Lenstra, and Lovász [6]. Aardal et al. motivate their choice of basis reduction as the main ingredient of their algorithm by arguing that one can interpret problem (2) as checking whether there exists a *short* (with respect to the bounds $\boldsymbol{l}$, $\boldsymbol{u}$) integral vector satisfying the system $\boldsymbol{Ax} = \boldsymbol{d}$. Given a lattice, the basis reduction algorithm finds a basis spanning that lattice such that the basis consists of short, nearly orthogonal vectors. Hence, a lattice is chosen that seems particularly useful for problem (2). An initial basis that spans the given lattice is derived, and the basis reduction algorithm is applied to this basis. The parameters of the initial basis are chosen such that the reduced basis contains one vector $\boldsymbol{x}_d$ satisfying $\boldsymbol{Ax}_d = \boldsymbol{d}$, and $n-m$ linearly independent vectors $\boldsymbol{x}_0$ satisfying $\boldsymbol{Ax}_0 = \boldsymbol{0}$. Due to the basis reduction algorithm all these vectors are relatively short. If the vector $\boldsymbol{x}_d$ satisfies the bounds, then the algorithm terminates, and if not, one observes that $\boldsymbol{A}(\boldsymbol{x}_d + \lambda\boldsymbol{x}_0) = \boldsymbol{d}$ for any integer multiplier λ and any vector $\boldsymbol{x}_0$ such that $\boldsymbol{Ax}_0 = \boldsymbol{0}$. Hence, one can branch on integer linear combinations of vectors $\boldsymbol{x}_0$ satisfying $\boldsymbol{Ax}_0 = \boldsymbol{0}$ in order to obtain either a vector satisfying the diophantine equations as well as the lower and upper bounds, or a proof that no such vector exists.

In our computational study we solve both feasibility and optimization versions of the market split problem. The optimization version can be solved by a slightly adapted version of the Aardal-Hurkens-Lenstra algorithm. We have solved instances with up to 7 equations and 60 variables. To our knowledge, the largest feasibility instances solved so far had 6 constraints and 50 variables, and the largest optimization instances had 4 constraints and 30 variables. These results were reported by Cornuéjols and Dawande [2]. Our computational experience is presented in Section 3.

When performing the computational study we observed that the larger the instances became, the more often feasible instances were generated. This motivated us to analyze the expected number of solutions for instances generated according to Cornuéjols and Dawande. Our conclusion is that for a given value of $m > 4$, one needs to generate slightly fewer variables than is given by the expression $n = 10(m-1)$ (keeping all other parameters the same) in order to generate infeasible instances with high probability. We present our analysis together with numerical support in Section 4.

2 An Outline of the Algorithm

Here we give a summary of the algorithm developed by Aardal, Hurkens, and Lenstra [1] to solve problem (2). They also give a brief overview of the basis reduction algorithm and the use of basis reduction in integer programming. For a detailed description of the basis reduction algorithm we refer to Lenstra, Lenstra, and Lovász [6].

The main idea behind the algorithm is to use an integer relaxation of the set $X = \{x \in \mathbb{Z}^n : \boldsymbol{Ax} = \boldsymbol{d},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$. The relaxation that Aardal et al. consider is $X_{IR} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{Ax} = \boldsymbol{d}\}$. To determine whether X_{IR} is empty can be done in polynomial time. Aardal et al. rewrite the set X_{IR} as follows:

$$X_{IR} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{x} = \boldsymbol{x}_d + \boldsymbol{X}_0\boldsymbol{\lambda},\ \boldsymbol{\lambda} \in \mathbb{Z}^{n-m}\}, \tag{4}$$

where $\boldsymbol{x}_d \in \mathbb{Z}^n$ satisfies $\boldsymbol{A}\boldsymbol{x}_d = \boldsymbol{d}$, and where $\boldsymbol{X}_0$ is an integral $n \times (n-m)$ matrix such that the columns $\boldsymbol{x}_0^j$, $1 \leq j \leq n-m$, of $\boldsymbol{X}_0$ are linearly independent and such that each column $\boldsymbol{x}_0^j$ satisfies $\boldsymbol{A}\boldsymbol{x}_0^j = \boldsymbol{0}$. Note that reformulation (4) is not unique. Expression (4) states that any integer vector $\boldsymbol{x}$ satisfying $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{d}$ can be described as a vector $\boldsymbol{x}_d$, satisfying $\boldsymbol{A}\boldsymbol{x}_d = \boldsymbol{d}$, plus an integer linear combination of vectors that form a basis for the lattice $\{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}$. Of course this observation also holds for the integer vectors $\boldsymbol{x}$ satisfying the bound constraints, if such vectors exist. Aardal et al. argue that if we are able to find a vector $\boldsymbol{x}_d$ that is reasonably short, then this vector will hopefully satisfy the bound constraints. If that is the case, one is done, and if not one needs to check whether there exists an integer linear combination of the columns of $\boldsymbol{X}_0$, $\boldsymbol{X}_0\boldsymbol{\lambda}$, such that $\boldsymbol{x} = \boldsymbol{x}_d + \boldsymbol{X}_0\boldsymbol{\lambda}$ satisfies the bounds.

One way of obtaining the vectors $\boldsymbol{x}_d$ and $\boldsymbol{x}_0^j$, $1 \leq j \leq n-m$, is by using the Hermite normal form of the matrix $\boldsymbol{A}$, see Schrijver [11], and Aardal et al. [1]. The elements of the Hermite normal form, however, tend to be relatively large, whereas we want the vector $\boldsymbol{x}_d$ to contain small elements. Moreover having the other numbers in the computation large may cause numerical problems depending on which branching strategy we apply. Aardal et al. therefore chose to use lattice basis reduction to derive the vectors $\boldsymbol{x}_d$ and $\boldsymbol{x}_0^j$, $1 \leq j \leq n-m$.

Let $\boldsymbol{I}^{(p)}$ denote the p-dimensional identity matrix, and let $\boldsymbol{0}^{(p\times q)}$ denote the $(p \times q)$-matrix consisting of only zeros. Let N_1 and N_2 be positive integral numbers. Consider the following linearly independent column vectors $\boldsymbol{B} = (\boldsymbol{b}_j)_{1\leq j\leq n+1}$:

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{I}^{(n)} & \boldsymbol{0}^{(n\times 1)} \\ \boldsymbol{0}^{(1\times n)} & N_1 \\ N_2\boldsymbol{A} & -N_2\boldsymbol{d} \end{pmatrix}. \tag{5}$$

The vectors of $\boldsymbol{B}$ span the lattice $L \subset \mathbb{R}^{n+m+1}$ that contains vectors of the form

$$(\boldsymbol{x}^T, N_1 y, N_2(\boldsymbol{a}_1\boldsymbol{x} - d_1 y), \ldots, N_2(\boldsymbol{a}_m\boldsymbol{x} - d_m y))^T = \boldsymbol{B}\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix}, \tag{6}$$

where y is a variable associated with the right-hand-side vector $\boldsymbol{d}$.

Proposition 1 *The integer vector $\boldsymbol{x}_d$ satisfies $\boldsymbol{A}\boldsymbol{x}_d = \boldsymbol{d}$ if and only if the vector*

$$(\boldsymbol{x}_d^T, N_1, \boldsymbol{0}^{(1\times m)})^T = \boldsymbol{B}\begin{pmatrix} \boldsymbol{x}_d \\ 1 \end{pmatrix} \tag{7}$$

belongs to the lattice L, and the integer vector $\boldsymbol{x}_0$ satisfies $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$ if and only if the vector

$$(\boldsymbol{x}_0^T, 0, \boldsymbol{0}^{(1\times m)})^T = \boldsymbol{B}\begin{pmatrix} \boldsymbol{x}_0 \\ 0 \end{pmatrix} \tag{8}$$

belongs to the lattice L.

Aardal, Hurkens, and Lenstra [1] proved that if there exists an integer vector $\boldsymbol{x}$ satisfying the system $\boldsymbol{Ax} = \boldsymbol{d}$, and if the numbers N_1 and N_2 are chosen appropriately, i.e., large enough with respect to the input and relative to each other, then the first $n - m + 1$ columns of the reduced basis that is obtained by applying the basis reduction algorithm to $\boldsymbol{B}$ will be of the following form:

$$\begin{pmatrix} \boldsymbol{X}_0^{(n\times(n-m))} & \boldsymbol{x}_d \\ \boldsymbol{0}^{(1\times(n-m))} & N_1 \\ \boldsymbol{0}^{(m\times(n-m))} & \boldsymbol{0}^{(m\times 1)} \end{pmatrix}, \tag{9}$$

where $\boldsymbol{X}_0 = (\boldsymbol{x}_0^1, \ldots, \boldsymbol{x}_0^{(n-m)})$. Due to Proposition 1 we can conclude that $\boldsymbol{x}_d$ satisfies $\boldsymbol{Ax}_d = \boldsymbol{d}$, and that $\boldsymbol{Ax}_0^j = \boldsymbol{0},\ 1 \leq j \leq n - m$.

Problem (2) can now be formulated equivalently as (cf. (4)):

$$\text{does there exist a vector } \boldsymbol{\lambda} \in \mathbb{Z}^{(n-m)} \text{ s. t. } \boldsymbol{l} - \boldsymbol{x}_d \leq \boldsymbol{X}_0\boldsymbol{\lambda} \leq \boldsymbol{u} - \boldsymbol{x}_d? \tag{10}$$

Unless $\boldsymbol{x}_d$ satisfies the bound constraints, one needs to branch on the variables λ_j in order to check whether the polytope $P = \{\boldsymbol{\lambda} \in \mathbb{Z}^{(n-m)} : \boldsymbol{l} - \boldsymbol{x}_d \leq \boldsymbol{X}_0\boldsymbol{\lambda} \leq \boldsymbol{u} - \boldsymbol{x}_d\}$ is empty. The basis reduction algorithm runs in polynomial time. If one wants an overall algorithm that runs in polynomial time for a fixed number of variables, one needs to apply the algorithms of H.W. Lenstra, Jr. [7] or of Lovász and Scarf [9]. Otherwise, one can, for instance, apply integral branching on the unit vectors in $\boldsymbol{\lambda}$-space or linear programming based branch-and-bound. By integral branching in the $\boldsymbol{\lambda}$-space we mean the following. Assume we are at node k of the search tree. Take any unit vector $\boldsymbol{e}_j$, i.e., the jth vector of the $(n - m)$-dimensional identity matrix, that has not yet been considered at the predecessors of k. Measure the width of the polytope P in this direction, i.e., determine $u_k = \max\{\boldsymbol{e}_j^T\boldsymbol{\lambda} : \boldsymbol{\lambda} \in P \cap \{\lambda_j\text{'s fixed at predecessors of } k\}\}$ and $l_k = \min\{\boldsymbol{e}_j^T\boldsymbol{\lambda} : \boldsymbol{\lambda} \in P \cap \{\lambda_j\text{'s fixed at predecessors of } k\}\}$. Create $\lfloor u_k \rfloor - \lceil l_k \rceil + 1$ subproblems at node k of the search tree by fixing λ_j to the values $\lceil l_k \rceil, \ldots, \lfloor u_k \rfloor$. The question is now in which order we should choose the unit vectors in our branching scheme. One alternative is to just take them in any predetermined order, and another is to determine, at node k, which unit vector yields the smallest value of $\lfloor u_k \rfloor - \lceil l_k \rceil$. This branching scheme is similar to the scheme proposed by Lovász and Scarf [9] except that we in general are not sure whether the number of branches created at each level is bounded by some constant depending only on the dimension. What we hope for, and what seems to be the case given our computational results, is that $\lfloor u_k \rfloor - \lceil l_k \rceil$ is small at most nodes of the tree. A natural question in the context of branching is whether we may hope that linear programming based branch-and-bound is more efficient on the polytope P as compared to the polytope X. As can be observed in Section 3 we typically reduce the number of nodes if we branch on P instead of X by several orders of magnitude. One way of explaining this is that we obtain a scaling effect by going from description (2) of our problem to description (10), see Aardal et al. [1].

3 Computational Experience

3.1 The Feasibility Version

We solved 17 instances of problem FP (1) reformulated as problem (10). Three of the instances were feasible and 14 infeasible. The input was generated as described in Section 1. The instances M16 and M17 are the instances "markshare1" and "markshare2" of MIPLIB [10].

In our computational study we wanted to determine the effect of the reformulation of problem (1) to problem (10). Therefore, we solved formulation (10) by linear programming based branch-and-bound in order to compare our experience with the branch-and-bound results reported by Cornuéjols and Dawande [2] on formulation (1). Given that we consider an integer relaxation (4) rather than a linear relaxation when determining (10), we also wanted to investigate the effect of maintaining the integral representation by applying integral branching. This explains the choice of the three branching methods that we considered in our study: linear programming based branch-and-bound on the variables λ_j, branching on the unit vector in $\boldsymbol{\lambda}$-space that yields the smallest value of $\lfloor u_k \rfloor - \lceil l_k \rceil$ at node k of the search tree as described in Section 2, and branching on the unit vectors in the predetermined order $j = n-m, \ldots, 1$, see Section 2. For the linear programming based branch-and-bound we used CPLEX, see below. Since CPLEX solves optimization problems we needed to reformulate (10) for this branching method as follows. We introduced the objective function $\max \sum_{j=1}^{n-m} \lambda_j$, and if a feasible solution was found we terminated the search. An example of an enumeration tree derived by the third method (branching on the unit vectors in the predetermined order $j = n-m, \ldots, 1$) for an instance of size 4×30 is shown in Figure 1. Notice that few branches are created at each node of the tree, and that no branching occurs below level eight.

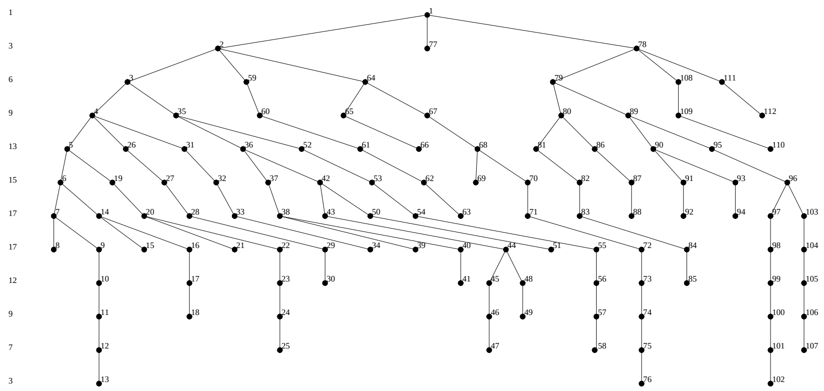

Figure 1: Enumeration tree for integer branching $\boldsymbol{e}_j$, $j = (n-m), \ldots, 1$

Table 1: Results for the feasibility version

Inst.	m	n	type	LP B&B on $\boldsymbol{\lambda}$ # nodes	LP B&B on $\boldsymbol{\lambda}$ time (s)	thinnest $\boldsymbol{e}_j$ # nodes	thinnest $\boldsymbol{e}_j$ time (s)	$\boldsymbol{e}_j$, $j = (n-m), \ldots, 1$ # nodes	$\boldsymbol{e}_j$, $j = (n-m), \ldots, 1$ time (s)
M1	5	40	N	7,203	40	1,723	652	3,107	58
M2	5	40	N	10,488	58	3,839	1,335	8,252	118
M3	5	40	Y	5,484	30	1,398	558	2,155	39
M4	5	40	N	16,484	84	5,893	1,685	14,100	175
M5	5	40	N	17,182	94	3,027	1,163	5,376	85
M6	5	40	N	11,500	62	2,762	1,010	5,322	87
M7	5	40	N	16,666	88	4,025	1,391	9,710	123
M8	5	40	N	7,483	42	2,386	899	4,310	68
M9	5	40	N	6,393	36	1,674	660	3,115	50
M10	5	40	N	17,206	90	3,791	1,319	7,860	115
M11	6	50	N	413,386	3,690	88,619	53,713	152,399	3,532
M12	6	50	Y	53,273	428	14,456	9,078	8,479	229
M13	6	50	N	375,654	3,080	86,505	51,626	141,259	3,596
M14	6	50	N	381,813	2,984	108,725	75,794	204,367	4,193
M15	6	50	Y	96,470	768	69,981	37,774	129,402	2,700
M16	6	50	N	114,215	932	40,274	24,969	79,130	1,639
M17	7	60	N	108,154	1,228	36,288	36,023	73,877	2,095

The information in Table 1 is interpreted as follows. In the first three columns, "Instance", "m", and "n", the instance names and the dimension of the instances are given. A "Y" in column "type" means that the instance is feasible, and an "N" that it is not feasible. Note that it is not known a priori whether the instances are feasible, but it is established by our algorithm. In the following six columns the number of nodes and the computing times are given for the three branching methods. The computing times for the method "thinnest $\boldsymbol{e}_j$" and "$\boldsymbol{e}_j,\ j = (n-m), \ldots, 1$", are given in seconds on a Sun Ultra Enterprise 2 with two 168 MHz Ultra Sparc processors (our implementation is sequential), SpecInt95 6.34, SpecFp95 9.33. The linear programming branch-and-bound computations were carried out on an Alphaserver 4100 5/400 with four 400 MHz 21164 Alpha Processors (sequential implementation) SpecInt95 12.1, SpecFp95 17.2. The times reported on for this method are the actual times obtained with the Alphaserver multiplied by a factor of 2, in order to make it easier to compare the times of the three methods. The basis reduction in the algorithm by Aardal, Hurkens, and Lenstra is done using LiDIA, a library for computational number theory [8]. The average computing time for the Aardal-Hurkens-Lenstra algorithm was for the three instance sizes 1.6, 3.1, and 4.8 seconds respectively. These computations were all carried out on the Sun Ultra Enterprise 2. For the LP-based branch-and-bound on the variables λ_j we used CPLEX 6.5 [4], and for the other two methods we use the enumeration scheme developed by Verweij [12]. The linear programming subproblems that arise in these two methods when determining $\lfloor u_k \rfloor$ and $\lceil l_k \rceil$, are solved using CPLEX version 6.0.1 [3].

An important conclusion of the experiments is that the reformulation itself is essential. Cornuéjols and Dawande [2] could only solve instances up to size 4×30 using CPLEX version 4.0.3. We also applied CPLEX versions 6.0 and 6.5 to the initial problem formulation (1) and observed a similar behavior, whereas CPLEX produced very good results on the reformulated problem (10). Cornuéjols and Dawande did solve feasibility instances of size 6×50 by a group relaxation approach. Their computing times are a factor of 3-10 slower than ours. No previous computational results for instances of size 7×60 have, to our knowledge, been reported.

If we consider the number of nodes that we enumerate using the three methods, we note that branching on the thinnest unit vector (in $\boldsymbol{\lambda}$-space) yields the fewest number of nodes for all instances except instance M12, which is a feasible instance. We do, however, need to determine the unit vector that yields the thinnest direction at each node of the tree, which explains the longer computing times. Branching on unit vectors in the predetermined order $j = n-m, \ldots, 1$, also requires fewer nodes for most instances than the linear programming based branching on the variables λ_j. In terms of computing times, linear programming based branch-and-bound is for most instances the fastest method, but does not differ too much from the times needed for branching on unit vectors $\boldsymbol{e}_j$, $j = n-m, \ldots, 1$. It is worth noticing that our integer branching implementation, in particular the communication between our branching scheme and CPLEX, can be made more efficient. Our results indicate that integer branching is an interesting method, in particular if we can find reasonably good branching directions quickly, as in the third method. In our case it seems as if the unit vectors in $\boldsymbol{\lambda}$-space yield thin branching directions. To investigate this we applied the generalized basis reduction algorithm of Lovász and Scarf [9] to our polytope P. The reduced basis vectors yielded thinner directions than the strategy "thinnest $\boldsymbol{e}_j$" in only about 6% of the cases for the instances of size 5×40. This implies that the unit vectors in $\boldsymbol{\lambda}$-space, in some order, basically form a reduced basis in the Lovász-Scarf sense. The computations involved in determining a Lovász-Scarf reduced basis are fairly time consuming. For a problem of dimension 5×40, at the root node of the tree, one has to solve at least 100 linear programs to determine the basis. For each level of the tree the number of linear programs solved at each node will decrease as the dimension of the subproblems decrease. If the unit basis would generate bad search directions, then a heuristic version of the Lovász-Scarf algorithm would be a possibility.

3.2 The Optimization Version

The algorithm by Aardal, Hurkens, and Lenstra [1] was primarily designed to solve feasibility problems, but can with simple adaptions be used to solve the optimization version (3) of the market split instances as well. Below, we report on the results obtained by using three different strategies to solve the optimization version. All strategies are based on linear programming based branch-and-bound, and we us CPLEX in our study.

Strategy 1: Here, we solve a sequence of feasibility problems. We start with the feasibility version (10). If the instance is infeasible, then we proceed by considering the following sequence of feasibility problems for $v = 1, 2, \ldots$ until a feasible solution is found.

$$\text{do there exist vectors } \boldsymbol{x} \in \{0,1\}^n,\ \boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}^m_{\geq 0}: \tag{11}$$
$$\sum_{j=1}^{n} a_{ij}x_j + s_i - w_i = d_i,\ 1 \leq i \leq m,\ \sum_{i=1}^{m}(s_i + w_i) = v?$$

These feasibility problems are then reformulated as problems of type (10) using the algorithm of Aardal et al. For each of these feasibility problems we apply linear programming based branch-and-bound on the variables λ_j. As mentioned in the previous section we need to actually reformulate the feasibility problems as optimization problems in order to use CPLEX. We therefore investigate the influence of the choice of objective function on the search that CPLEX is performing. In Strategy 1 we use the objective function $\max \sum_{j=1}^{n+m-1} \lambda_j$.

Strategy 2: This is the same as Strategy 1 except that the objective function is a perturbation of the objective function zero. Here we sketch the principle of the perturbation. What is basically done to construct the perturbed objective function is to perturb the variables of the linear programming dual as follows. Notice that the number of constraints in the linear relaxation of the reformulation (10) of the feasibility problem (11) is $p = 2n + 2m$; we have $2n$ constraints corresponding to the upper and lower bounds on the $\boldsymbol{x}$-variables, and $2m$ constraints corresponding to the nonnegativity requirements on the slack variables s_i and w_i, $1 \leq i \leq m$. Let $\varepsilon = 10^{-6}$ and let, for $i = 1, \ldots, p$, Z_i be drawn uniformly and independently from the interval $[0, 1]$. Let $\delta_i = \varepsilon Z_i$. If the dual variable $y_i \leq 0$ in the original formulation we let $y_i \leq \delta_i$, and if $y_i \geq 0$ we let $y_i \geq -\delta_i$. For y_i such that $y_i \leq \delta_i$, make the substitution $Y_i = y_i - \delta_i$, and for $y_i \geq -\delta_i$ we substitute y_i by $Y_i = y_i + \delta_i$. This substitution implies a perturbation of the primal objective function.

Strategy 3: Here we view the problem as an optimization problem directly, which implies that only one problem is solved instead of a sequence of problems as in Strategies 1 and 2. We extract the expressions of the slack variables s_i and w_i in terms of the variables λ_j and minimize the sum of the slack variables expressed in the λ_j's.

For all computations we used CPLEX version 6.5. The computations were made on an Alphaserver 4100 5/400 as described in the previous subsection. From the results in Table 2 we can conclude that instances of sizes up to 7×60 are relatively easy to solve to optimality after using the reformulation of the problem implied by the algorithm of Aardal, Hurkens, and Lenstra [1]. This represents a large improvement over earlier results, where the largest optimization instances had dimension 4×30, see Cornuéjols and Dawande [2]. If we consider the number of nodes that we enumerate when applying linear programming based branch-and-bound on the variables λ_j, we observe that this number is significantly smaller than the number $2^{\alpha n}$ for α between 0.6 and 0.7 that Cornuéjols and Dawande observed when applying branch-and-bound on the x_j-variables. For instances of size 4×30 they enumerated between 10^6 and 2×10^6 nodes. For the same number of enumeration nodes we can solve

Table 2: Results for the optimization version

Inst.	Strat. 1 # nodes	Strat. 1 time (s)	Strat. 2 # nodes	Strat. 2 time (s)	Strat. 3 # nodes	Strat. 3 time (s)
M1	20,022	57	13,699	38	53,267	180
M2	6,451	44	75,174	222	109,498	281
M3	5,484	15	5,230	15	41,518	118
M4	36,847	107	75,985	211	176,437	456
M5	32,880	93	16,379	45	271,208	705
M6	35,710	105	21,277	62	220,603	875
M7	64,090	180	88,160	254	396,416	1,261
M8	33,937	99	35,471	103	122,526	383
M9	9,910	29	19,083	56	200,987	625
M10	28,402	76	146,224	436	99,502	299
M11	1,165,498	6,018	1,728,646	7,811	6,158,407	29,486
M12	53,273	214	73,801	283	5,101,843	22,002
M13	810,496	505	1,410,840	6,607	6,057,528	26,333
M14	384,882	1,505	1,117,107	4,748	7,861,402	34,083
M15	96,470	384	17,007	60	9,558,521	37,960
M16	282,665	1,190	319,029	1,248	71,253	348
M17	567,837	3,767	1,823,915	11,597	3,425,941	27,168

instances of more than twice that size. We can also observe that solving the reformulated optimization version (Strategy 3) instead of a sequence of feasibility problems (Strategies 1 and 2) is more time consuming in most cases. One reason is that the optimum objective value is small, so the number of problems we need to solve in Strategies 1 and 2 is small, but in case of the infeasible instances greater than one. If one expects the optimum value to be large and no reasonable bounds are known, then it is probably better to consider Strategy 3.

Next to the instances we report on here we also generated another five instances of size 7×60. All these instances were feasible so we decided not to report on the results of the computations here. For the size 6×50 we also had to generate quite a few instances to obtain infeasible ones. This motivated us to investigate whether the relation $n = 10(m-1)$, as Cornuéjols and Dawande suggested, is actually likely to produce infeasible instances if we keep all other parameters as they suggested. Our probabilistic analysis is presented in the next section.

4 The Expected Number of Solutions

Here, we derive an expression for the expected number of solutions for problem formulation (1), given that the coefficients a_{ij} are generated uniformly and independently from the set $\{0,\dots,D-1\}$, and that $d_i = \lfloor f\sum_{j=1}^{n} a_{ij}\rfloor$, cf. Section 1. In addition, we relate the expected number of solutions to the probability of generating an infeasible instance.
The analysis is easily adapted for arbitrary rational fractions f_i. Cornuéjols and Dawande [2] use $D = 100$ and $f = \frac{1}{2}$.

4.1 The Probability that a Subset Induces a Solution

Consider a subset $S \subseteq \{1,2,\dots,n\}$, and let $x_j = 1$ if $j \in S$, and $x_j = 0$ otherwise. We compute the probability that $\sum_{j\in S} a_{ij} = \lfloor f\sum_{j=1}^{n} a_{ij}\rfloor$, in which case the vector $\boldsymbol{x}$ as defined above satisfies $\boldsymbol{Ax} = \boldsymbol{d}$ for row i. Define a random variable $Z_i(S) = \sum_{j\in S} a_{ij} - \lfloor f\sum_{j=1}^{n} a_{ij}\rfloor$ denoting the difference between the left-hand side and the right-hand side of row i. The probability that we want to compute is therefore $\Pr[Z_i(S) = 0]$. Let the random variables $Y_i(S)$ and U_i be defined as

$$Y_i(S) = \sum_{j\in S} a_{ij} - f\sum_{j=1}^{n} a_{ij} = \sum_{j\in S}(1-f)a_{ij} - \sum_{j\notin S} f a_{ij}\ , \tag{12}$$

and $U_i = f\sum_{j=1}^{n} a_{ij} - \lfloor f\sum_{j=1}^{n} a_{ij}\rfloor$. Hence, we can write $Z_i(S) = Y_i(S) + U_i$.

For any rational fraction $f = P/Q$ $(P,Q \in \mathbb{N}, \gcd(P,Q) = 1)$, we have $Y_i(S) \in \frac{1}{Q}\mathbb{Z}$ and $U_i \in \frac{1}{Q}\mathbb{Z} \cap [0,1)$. Since $Y_i(S) + U_i \equiv 0 \pmod 1$, we have

$$\Pr[Z_i(S) = 0] = \Pr[Y_i(S) = -U_i] = \sum_{k=0}^{Q-1} \Pr[Y_i(S) = \frac{-k}{Q}]\ . \tag{13}$$

We can give an approximation using the normal distribution as described in Section 4.2, or compute this probability exactly using the probability generating function of $Y_i(S)$, see Section 4.3. In either case, we obtain an expression $\Pr[Z_i(S) = 0] = q(n,D,|S|)$, i.e., the probability that $\boldsymbol{x}$ induced by S defines a solution for $\boldsymbol{a}_i\boldsymbol{x} = \boldsymbol{d}_i$ depends on n, D and the size of S only. The probability that S induces a solution for $\boldsymbol{Ax} = \boldsymbol{d}$ is given by $q(n,D,|S|)^m$. Let $\mathcal{S}$ denote the set of all subsets that induce a solution. The expected number of solutions is derived by summing over all subsets S, i.e.,

$$\mathrm{E}[\#\mathcal{S}] = \sum_{S\subset\{1,\dots,n\}} q(n,D,|S|)^m = \sum_{s=0}^{n} \binom{n}{s} q(n,D,s)^m\ . \tag{14}$$

4.2 An Approximation

As a first approach we compute $q(n,D,|S|)$ by approximating the distribution of $Y_i(S)$ by the normal distribution. Each of the coefficients a_{ij} has expectation $\frac{1}{2}(D-1)$ and variance $\frac{1}{12}(D^2-1)$. Since they are drawn independently, we obtain $\mathrm{E}[Y_i(S)] = \frac{1}{2}(D-1)(|S|-nf)$ and $\mathrm{Var}[Y_i(S)] = \frac{1}{12}(D^2-1)(|S|(1-2f)+f^2 n)$. Note that for $f = \frac{1}{2}$, the variance reduces to $(D^2-1)n/48$. For rational $f = P/Q$, the probability that subset S induces a solution for row i is given by $\Pr[Z_i(S)=0] = \Pr[1/(2Q)-1 \le Y_i(S) \le 1/(2Q)]$. Using the Central Limit Theorem [5], we approximate this expression by the normal distribution, and find

$$\Pr\left[\frac{1}{2Q}-1 < Y_i(S) < \frac{1}{2Q}\right] \approx \frac{1}{\sqrt{2\pi}}\int_\alpha^\beta \exp(-\frac{1}{2}u^2)du \; , \tag{15}$$

with

$$\alpha = \frac{\frac{1}{2Q}-1-\mathrm{E}[Y_i(S)]}{\sqrt{\mathrm{Var}[Y_i(S)]}} \; , \quad \text{and} \quad \beta = \frac{\frac{1}{2Q}-\mathrm{E}[Y_i(S)]}{\sqrt{\mathrm{Var}[Y_i(S)]}} \; . \tag{16}$$

Since we are adding exponentially many of these terms in expression (14), it is unclear how accurate the result is. Therefore an exact formulation is derived as well.

4.3 An Exact Formulation

The probability generating function of a_{ij} is

$$G_{a_{ij}}(x) = \mathrm{E}[x^{a_{ij}}] = \sum_{k=0}^{D-1}\frac{1}{D}x^k = \frac{1}{D}(1+x+\cdots+x^{D-1}) = \frac{1}{D}\cdot\frac{x^D-1}{x-1} \; . \tag{17}$$

Since the coefficients a_{ij} are independent, the probability generating function of $Y_i(S)$ is given by

$$\begin{aligned}
G_{Y_i(S)}(x) &= \mathrm{E}[x^{Y_i(S)}] = \mathrm{E}[x^{\sum_{j\in S}(1-f)a_{ij} - \sum_{j\notin S} f a_{ij}}] \\
&= \prod_{j\in S}\mathrm{E}[x^{(1-f)a_{ij}}]\prod_{j\notin S}\mathrm{E}[x^{-fa_{ij}}] \\
&= \prod_{j\in S}\left(\frac{1}{D}\cdot\frac{x^{(1-f)D}-1}{x^{(1-f)}-1}\right)\prod_{j\notin S}\left(\frac{1}{D}\cdot\frac{x^{(-f)D}-1}{x^{(-f)}-1}\right) \\
&= \left(\frac{1}{D}\right)^n\left(\frac{x^{(1-f)D}-1}{x^{(1-f)}-1}\right)^{|S|}\left(\frac{x^{(-f)D}-1}{x^{(-f)}-1}\right)^{n-|S|} \\
&= \frac{1}{D^n}\,\frac{1}{x^{f(D-1)(n-|S|)}}\left(\frac{x^{(1-f)D}-1}{x^{(1-f)}-1}\right)^{|S|}\left(\frac{x^{fD}-1}{x^f-1}\right)^{n-|S|} \; .
\end{aligned} \tag{18}$$

For rational $f = P/Q$, we can expand expression (18) to $\sum_j c_j x^{j/Q}$, where c_j denotes the probability $\Pr[Y_i(S) = \frac{j}{Q}]$. To compute $\Pr[Z_i(S)=0]$, we need to evaluate only the coefficients c_j for $j = 0, -1, \ldots, -Q+1$. For $f = \frac{1}{2}$, expression (18) is equal to

$$\frac{1}{D^n}\,\frac{1}{x^{(D-1)(n-|S|)/2}}\left(\frac{x^{D/2}-1}{x^{1/2}-1}\right)^n \; . \tag{19}$$

In order to find the Taylor expansion of the factors in (18) and (19) we use the following lemma.

Lemma 2 $\left(\frac{y^D-1}{y-1}\right)^N = \sum_{j=0}^{\infty} a_j y^j$ *with*

$$a_j = \sum_{k=0}^{\min\{N,\lfloor j/D\rfloor\}} \binom{N}{k} (-1)^k \binom{j-Dk+N-1}{j-Dk} . \tag{20}$$

Proof: Write $(\frac{y^D-1}{y-1})^N$ as the product of $(1-y^D)^N$ and $(1-y)^{-N}$ with $(1-y^D)^N = \sum_{j=0}^{N} \binom{N}{j}(-1)^j y^{Dj}$ and $(1-y)^{-N} = \sum_{j=0}^{\infty} \binom{j+N-1}{j} y^j$. Note that $a_j = 0$ for $j > (D-1)N$, as $(\frac{y^D-1}{y-1})^N = (1+y+\cdots+y^{D-1})^N$. □

For $f=\frac{1}{2}$, we apply the lemma with $N=n$ and $y=x^{1/2}$, to obtain

$$\Pr[Z_i(S)=0] = q(n,D,|S|) = \frac{1}{D^n}\left(a_{(D-1)(n-|S|)} + a_{(D-1)(n-|S|)-1}\right) . \tag{21}$$

For $f \neq \frac{1}{2}$, we use the lemma to find the coefficients a_j of each factor in expression (18) and compute the coefficients c_j by convolution of the two power series obtained.

4.4 The Probability of Generating Infeasible Instances

Finally, we relate the expected number of solutions to the probability of generating an infeasible instance. For simplification, we neglect the dependency between two distinct subsets, each not providing a solution. We further use $\log(1+x) = x + O(x^2)$, yielding

$$\begin{aligned}
\Pr[\#\mathcal{S}=0] &= \Pr[S \notin \mathcal{S},\ \forall S \subset \{1,2,\ldots,n\}] \\
&\approx \prod_{S\subset\{1,\ldots,n\}} \Pr[S \notin \mathcal{S}] \\
&= \exp\Big(\log \prod_{S\subset\{1,\ldots,n\}} (1-q(n,D,|S|)^m)\Big) \\
&= \exp\Big(\sum_{S\subset\{1,\ldots,n\}} \log(1-q(n,D,|S|)^m)\Big) \\
&\approx \exp\Big(\sum_{S\subset\{1,\ldots,n\}} -q(n,D,|S|)^m\Big) \\
&= \exp\left(-\mathrm{E}[\#\mathcal{S}]\right) .
\end{aligned} \tag{22}$$

For $f=\frac{1}{2}$ there is a correlation due to symmetry that we cannot neglect. If all row sums are even, then a subset S induces a solution if and only if its complement $\bar{S} = \{1,\ldots,n\}\backslash S$ does. We first compute the conditional probability that S does not induce a solution, given that $\bar{S}$ does not for the first k rows. Let $\mathcal{S}_i$ denote the set of subsets that induce a solution with respect to row i and let $q_S = q(n,D,|S|)$.

$$
\begin{aligned}
\Pr\ & \big[S \notin \mathcal{S} \mid (\forall\, i : 1 \le i \le k, \bar{S} \notin \mathcal{S}_i) \wedge (\forall\, i : k+1 \le i \le m, \bar{S} \in \mathcal{S}_i)\big] \\
&= \Pr\big[\exists\, i : 1 \le i \le k, \sum_j a_{ij} \text{ even}\big] + \Pr\big[\forall\, i : 1 \le i \le k, \sum_j a_{ij} \text{ odd}\big] \cdot \Big\{ \\
&\qquad \Pr\big[\exists\, i : k+1 \le i \le m, \sum_j a_{ij} \text{ odd}\big] \cdot 1 + \\
&\qquad\quad \Pr\big[\forall\, i : k+1 \le i \le m, \sum_j a_{ij} \text{ even}\big] \cdot \big(1 - (\frac{q_S}{1-q_S})^k\big)\Big\} \\
&= (1 - 2^{-k}) + 2^{-k}(1 - 2^{-(m-k)}) + 2^{-k}2^{-(m-k)}\big(1 - (\frac{q_S}{1-q_S})^k\big) \\
&\approx 1 - 2^{-m} q_S^k \, . && (23)
\end{aligned}
$$

Using (23) we derive an expression for the probability that both S and $\bar{S}$ are not in $\mathcal{S}$.

$$
\begin{aligned}
\Pr\big[S, \bar{S} \notin \mathcal{S}\big] \quad &\approx \quad \sum_{k=1}^{m} \binom{m}{k} (1 - q_{\bar{S}})^k q_{\bar{S}}^{m-k} \big(1 - 2^{-m} q_S^k\big) \\
&= \quad 1 - q_{\bar{S}}^m + (\frac{q_{\bar{S}}}{2})^m - \left(\frac{q_{\bar{S}} + q_S - q_S q_{\bar{S}}}{2}\right)^m \, . && (24)
\end{aligned}
$$

Since $q_{\bar{S}} \approx q_S$, we obtain

$$
\begin{aligned}
\Pr[\#\mathcal{S} = 0] \quad &\approx \quad \prod_{S:1\in S} \Pr[S, \bar{S} \notin \mathcal{S}] \\
&= \quad \exp \sum_{S:1\in S} \log \Pr[S, \bar{S} \notin \mathcal{S}] \\
&\approx \quad \exp \sum_{S:1\in S} -q_S^m (1 - \frac{1}{2^m} + (1 - \frac{q_S}{2})^m) \\
&\approx \quad \exp\Big(\sum_{S\subset\{1,\ldots,n\}} (-q_S^m)(1 - \frac{1}{2}\frac{1}{2^m})\Big) \\
&= \quad \exp\Big(- \mathrm{E}[\#\mathcal{S}](1 - \frac{1}{2^{m+1}})\Big) =: \tilde{p}_{m,n} \, . && (25)
\end{aligned}
$$

Note that for large values of $\mathrm{E}[\#\mathcal{S}]$ this approximation yields significantly different results compared to (22).

4.5 Computational Results

We have computed the expected number of solutions using probability generating functions (gen), and using an approximation by the normal distribution (approx) for several values of m and n. The results are presented in Table 3. In our computations we use $f = \frac{1}{2}$ and $D = 100$. The horizontal lines in the table indicate the relation $n = 10(m-1)$ as proposed by Cornuéjols and Dawande. We notice that the value obtained by the approximation overestimates the exact value with a relative error of at most 5.2%.

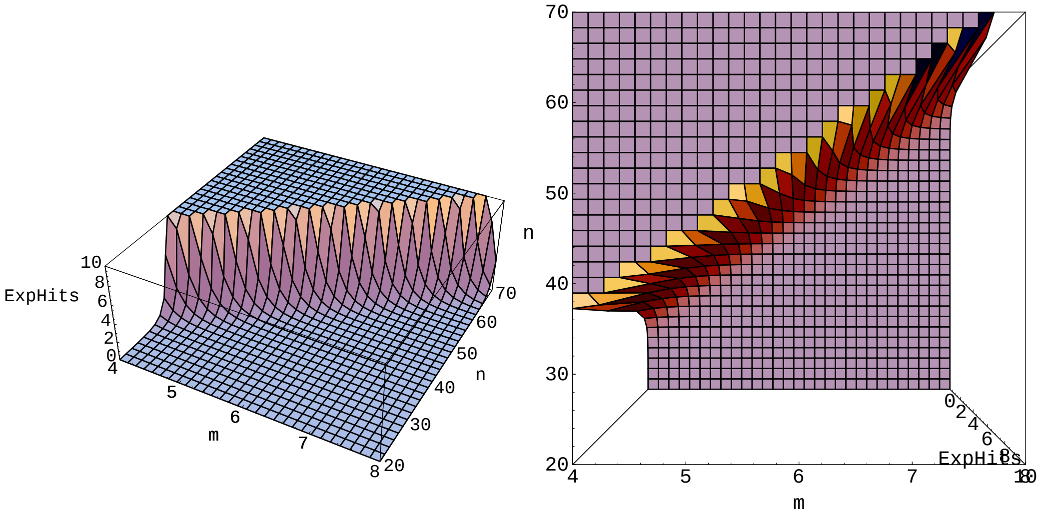

Figure 2: The expected number of solutions for $m = 4, \ldots, 8$, truncated at 10

Table 3 and Figure 2 show that, for fixed m, the expected number of solutions grows rapidly with n. In particular, we observe that for $m \geq 6$ and $n = 10(m-1)$, the expected number of solutions is greater than 0.9, which implies that the probability of generating an infeasible instance is less than 0.4. This confirms our experience with the instances we generated for our computational experiments reported on in Section 3. If one wants to generate infeasible instances with high probability for $m \geq 6$, then one needs to use slightly fewer columns for a given value of m than the relation $n = 10(m-1)$ indicates.

Let $p_{m,n}$ be the unknown probability of generating an infeasible instance. To test our approximation $\tilde{p}_{m,n}$ as given in expression (25) for a given instance size m, n, we generate N instances and count the infeasible ones. For such an experiment, let X_i be 1 if the ith sample is infeasible and 0 otherwise. Since X_i are independent binary random variables, $T = X_1 + \cdots + X_N$ is binomially distributed with parameters N and $p_{m,n}$ ($T \sim \mathrm{Bin}(N, p_{m,n})$). Our null hypothesis is $p_{m,n} = \tilde{p}_{m,n}$. Under this assumption $T \sim \mathrm{Bin}(N, \tilde{p}_{m,n})$. By looking at the 2.5% quantile and the 97.5% quantile, we construct a 95% confidence interval for T:

$$q_{0.025}(N, \tilde{p}_{m,n}) \leq T \leq q_{0.975}(N, \tilde{p}_{m,n}) \,. \tag{26}$$

In our experiment we chose $N = 1000$, and instance sizes as given in Table 4. For each combination of m and n we obtained a realization $t_{m,n} = x_1 + \cdots + x_N$ of T. In Table 4 we further report on our estimator $\tilde{p}_{m,n}$, the quantiles corresponding to the 95%-confidence interval, the value of our realization $t_{m,n}$, and $\hat{p}_{m,n} = t_{m,n}/N$. For all our tested instances we found that the outcome of our experiment lies within the 95%-confidence interval. This indicates that there is no reason to reject our null hypothesis.

Table 3: The expected number of solutions computed exactly using the probability generating function (gen) and approximated by the normal distribution (approx)

n	$m=4$		$m=5$		$m=6$		$m=7$		$m=8$	
	gen	approx	gen	approx	gen	approx	gen	approx	gen	approx
20	0.0004	0.0004	2.3091e-6	2.3692e-6	1.3071e-8	1.3519e-8	7.5214e-11	7.8457e-11	4.3865e-13	4.6155e-13
21	0.0007	0.0008	4.0743e-6	4.1700e-6	2.2299e-8	2.2964e-8	1.2316e-10	1.2759e-10	6.8386e-13	7.1256e-13
22	0.0014	0.0014	7.2921e-6	7.4623e-6	3.9302e-8	4.0507e-8	2.1514e-10	2.2339e-10	1.1927e-12	1.2479e-12
23	0.0025	0.0026	0.00001	0.00001	6.8320e-8	7.0205e-8	3.6171e-10	3.7380e-10	1.9270e-12	2.0023e-12
24	0.0046	0.0047	0.00002	0.00002	1.2125e-7	1.2461e-7	6.3449e-10	6.5644e-10	3.3604e-12	3.5006e-12
25	0.0086	0.0087	0.00004	0.00004	2.1378e-7	2.1924e-7	1.0881e-9	1.1219e-9	5.5765e-12	5.7801e-12
26	0.0159	0.0161	0.00008	0.00008	3.8208e-7	3.9177e-7	1.9192e-9	1.9797e-9	9.7509e-12	1.0121e-11
27	0.0295	0.0298	0.0001	0.0001	6.8118e-7	6.9740e-7	3.3415e-9	3.4385e-9	1.6516e-11	1.7079e-11
28	0.0548	0.0555	0.0003	0.0003	1.2257e-6	1.2544e-6	5.9298e-9	6.1020e-9	2.9002e-11	3.0013e-11
29	0.1023	0.1035	0.0005	0.0005	2.2050e-6	2.2540e-6	1.0449e-8	1.0733e-8	4.9914e-11	5.1513e-11
30	0.1913	0.1936	0.0009	0.0009	3.9929e-6	4.0798e-6	1.8657e-8	1.9159e-8	8.8096e-11	9.0939e-11
31	0.3585	0.3626	0.0016	0.0016	7.2371e-6	7.3878e-6	3.3199e-8	3.4045e-8	1.5357e-10	1.5819e-10
32	0.6732	0.6808	0.0030	0.0030	0.00001	0.00001	5.9626e-8	6.1124e-8	2.7250e-10	2.8068e-10
33	1.2668	1.2805	0.0055	0.0055	0.00002	0.00002	1.0696e-7	1.0953e-7	4.8000e-10	4.9361e-10
34	2.3879	2.4131	0.0102	0.0103	0.00004	0.00004	1.9319e-7	1.9774e-7	8.5633e-10	8.8043e-10
35	4.5090	4.5552	0.0189	0.0192	0.00008	0.00008	3.4895e-7	3.5685e-7	1.5215e-9	1.5623e-9
36	8.5279	8.6128	0.0353	0.0358	0.0001	0.0001	6.3353e-7	6.4759e-7	2.7288e-9	2.8010e-9
37	16.1533	16.3098	0.0659	0.0668	0.0003	0.0003	1.1512e-6	1.1758e-6	4.8844e-9	5.0085e-9
38	30.6410	30.9297	0.1234	0.1250	0.0005	0.0005	2.1000e-6	2.1440e-6	8.8038e-9	9.0240e-9
39	58.2021	58.7363	0.2314	0.2343	0.0009	0.0009	3.8357e-6	3.9137e-6	1.5859e-8	1.6241e-8
40	110.6973	111.6878	0.4345	0.4400	0.0017	0.0018	7.0282e-6	7.1681e-6	2.8719e-8	2.9400e-8
41	210.7999	212.6397	0.8174	0.8274	0.0032	0.0033	0.00001	0.00001	5.2021e-8	5.3216e-8
42	401.8960	405.3196	1.5399	1.5582	0.0060	0.0061	0.00002	0.00002	9.4624e-8	9.6756e-8
43	767.0835	773.4649	2.9050	2.9388	0.0112	0.0114	0.00004	0.00004	1.7224e-7	1.7601e-7
44	1465.6670	1477.5812	5.4876	5.5499	0.0209	0.0212	0.00008	0.00008	3.1459e-7	3.2134e-7
45	2803.3082	2825.5860	10.3793	10.4945	0.0391	0.0397	0.0001	0.0002	5.7517e-7	5.8721e-7
46	5366.9806	5408.6988	19.6556	19.8690	0.0733	0.0743	0.0003	0.0003	1.0545e-6	1.0761e-6
47	1.0285e4	1.0363e4	37.2658	37.6617	0.1375	0.1393	0.0005	0.0005	1.9356e-6	1.9744e-6
48	1.9726e4	1.9874e4	70.7327	71.4684	0.2582	0.2617	0.0010	0.0010	3.5613e-6	3.6313e-6
49	3.7868e4	3.8145e4	134.3989	135.7680	0.4856	0.4920	0.0018	0.0018	6.5607e-6	6.6867e-6
50	7.2753e4	7.3275e4	255.6339	258.1855	0.9144	0.9262	0.0033	0.0034	0.00001	0.00001
51	1.3989e5	1.4087e5	486.7099	491.4721	1.7238	1.7457	0.0062	0.0062	0.00002	0.00002
52	2.6918e5	2.7103e5	927.5451	936.4449	3.2537	3.2940	0.0116	0.0117	0.00004	0.00004
53	5.1834e5	5.2184e5	1769.2812	1785.9346	6.1478	6.2225	0.0216	0.0220	0.00008	0.00008
54	9.9884e5	1.0054e6	3377.8566	3409.0581	11.6287	11.7673	0.0405	0.0411	0.0001	0.0001
55	1.9261e6	1.9386e6	6454.3702	6512.8979	22.0181	22.2758	0.0761	0.0772	0.0003	0.0003
56	3.7165e6	3.7402e6	1.2343e4	1.2453e4	41.7308	42.2104	0.1429	0.1449	0.0005	0.0005
57	7.1757e6	7.2207e6	2.3622e4	2.3830e4	79.1668	80.0606	0.2687	0.2724	0.0009	0.0009
58	1.3863e7	1.3948e7	4.5245e4	4.5635e4	150.3239	151.9916	0.5058	0.5127	0.0017	0.0017
59	2.6799e7	2.6961e7	8.6723e4	8.7457e4	285.6905	288.8057	0.9531	0.9658	0.0032	0.0033
60	5.1835e7	5.2143e7	1.6634e5	1.6772e5	543.4188	549.2448	1.7978	1.8214	0.0060	0.0061
61	1.0031e8	1.0090e8	3.1928e5	3.2189e5	1034.5026	1045.4103	3.3943	3.4383	0.0112	0.0114
62	1.9424e8	1.9535e8	6.1324e5	6.1817e5	1970.9501	1991.3942	6.4148	6.4966	0.0211	0.0214
63	3.7629e8	3.7843e8	1.1786e6	1.1879e6	3757.9874	3796.3444	12.1341	12.2862	0.0395	0.0401
64	7.2936e8	7.3343e8	2.2666e6	2.2842e6	7170.6817	7242.7199	22.9726	23.2561	0.0733	0.0754
65	1.4144e9	1.4221e9	4.3616e6	4.3951e6	1.3692e4	1.3828e4	43.5290	44.0578	0.1397	0.1417
66	2.7440e9	2.7589e9	8.3980e6	8.4614e6	2.6164e4	2.6419e4	82.5476	83.5352	0.2629	0.2666
67	5.3262e9	5.3545e9	1.6179e7	1.6299e7	5.0029e4	5.0511e4	156.6664	158.5126	0.4952	0.5021
68	1.0343e10	1.0396e10	3.1186e7	3.1412e7	9.5728e4	9.6634e4	297.5662	301.0208	0.9337	0.9465
69	2.0092e10	2.0196e10	6.0146e7	6.0581e7	1.8328e5	1.8499e5	565.6096	572.0801	1.7619	1.7858
70	3.9050e10	3.9248e10	1.1606e8	1.1688e8	3.5114e5	3.5437e5	1075.8876	1088.0185	3.3274	3.3720
71	7.5923e10	7.6305e10	2.2406e8	2.2563e8	6.7315e5	6.7925e5	2047.9738	2070.7375	6.2893	6.3723
72	1.4767e11	1.4840e11	4.3279e8	4.3578e8	1.2912e6	1.3027e6	3901.0477	3943.8021	11.8969	12.0517
73	2.8734e11	2.8875e11	8.3636e8	8.4207e8	2.4781e6	2.4999e6	7435.8186	7516.1883	22.5215	22.8106
74	5.5932e11	5.6202e11	1.6170e9	1.6278e9	4.7588e6	4.8001e6	1.4183e4	1.4334e4	42.6664	43.2066
75	1.0891e12	1.0943e12	3.1277e9	3.1484e9	9.1434e6	9.2218e6	2.7068e4	2.7354e4	80.8888	81.8993
76	2.1215e12	2.1314e12	6.0523e9	6.0920e9	1.7577e7	1.7725e7	5.1694e4	5.2231e4	153.4610	155.3527
77	4.1339e12	4.1531e12	1.1717e10	1.1792e10	3.3807e7	3.4089e7	9.8781e4	9.9795e4	291.3437	294.8880
78	8.0580e12	8.0948e12	2.2693e10	2.2837e10	6.5057e7	6.5593e7	1.8887e5	1.9078e5	553.4833	560.1297
79	1.5712e13	1.5783e13	4.3968e10	4.4245e10	1.2525e8	1.2627e8	3.6133e5	3.6494e5	1052.1712	1064.6447
80	3.0646e13	3.0782e13	8.5223e10	8.5754e10	2.4126e8	2.4319e8	6.9164e5	6.9847e5	2001.4513	2024.8798

Table 4: The probability of generating an infeasible instance

m	n	$\mathrm{E}[\#\mathcal{S}]$	$\tilde{p}$	$q_{0.025}$	$q_{0.975}$	t	$\hat{p}$
4	25	0.0086	0.9917	986	997	993	0.993
4	26	0.0159	0.9847	977	992	979	0.979
4	27	0.0295	0.9718	961	982	978	0.978
4	28	0.0548	0.9483	934	962	949	0.949
4	29	0.1023	0.9056	887	923	888	0.888
4	30	0.1913	0.8308	807	854	821	0.821
4	31	0.3585	0.7066	678	735	704	0.704
4	32	0.6732	0.5209	490	552	537	0.537
4	33	1.2668	0.2931	265	322	283	0.283
4	34	2.3879	0.0989	81	118	85	0.085
5	33	0.0055	0.9946	990	999	995	0.995
5	34	0.0102	0.9900	983	996	987	0.987
5	35	0.0189	0.9816	973	989	981	0.981
5	36	0.0353	0.9658	954	977	964	0.964
5	37	0.0659	0.9372	922	952	929	0.929
5	38	0.1234	0.8856	866	905	887	0.887
5	39	0.2314	0.7963	771	821	774	0.774
5	40	0.4345	0.6520	622	681	628	0.628
5	41	0.8174	0.4473	416	478	466	0.466
5	42	1.5399	0.2196	194	246	200	0.200

Acknowledgements

We would like to thank Bram Verweij for his assistance in implementing our integral branching algorithm using his enumeration scheme [12]. We also want to thank David Applegate and Bill Cook for their many useful comments on our work and for allowing us to use their DEC Alphaservers.

References

[1] K. Aardal, C. Hurkens, A. K. Lenstra (1998). Solving a system of diophantine equations with lower and upper bounds on the variables. Research report UU-CS-1998-36, Department of Computer Science, Utrecht University.

[2] G. Cornuéjols, M. Dawande (1998). A class of hard small 0-1 programs. In: R. E. Bixby, E. A. Boyd, R. Z. Ríos-Mercado (eds.) *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference*. Lecture Notes in Computer Science 1412, pp 284–293, Springer-Verlag, Berlin Heidelberg.

[3] *CPLEX 6.0 Documentation Supplement* (1998). ILOG Inc., CPLEX Division, Incline Village NV.

[4] *CPLEX 6.5 Documentation Supplement* (1999). ILOG Inc., CPLEX Division, Incline Village NV.

[5] G. R. Grimmett, D. R. Stirzaker (1982). *Probability and Random Processes*, Oxford University Press, Oxford.

[6] A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, 515–534.

[7] H. W. Lenstra, Jr. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8, 538–548.

[8] LiDIA – A library for computational number theory. TH Darmstadt / Universität des Saarlandes, Fachbereich Informatik, Institut für Theoretische Informatik. `http://www.informatik.th-darmstadt.de/pub/TI/LiDIA`

[9] L. Lovász, H. E. Scarf (1992). The generalized basis reduction algorithm. *Mathematics of Operations Research* 17, 751–764.

[10] MIPLIB. `http://www.caam.rice.edu/~bixby/miplib/miplib3.html`

[11] A. Schrijver (1986). *Theory of Linear and Integer Programming.* Wiley, Chichester.

[12] A. M. Verweij (1998). *The UHFCO Library.* Department of Computer Science, Utrecht University.

[13] H. P. Williams (1978). *Model Building in Mathematical Programming.* John Wiley & Sons Ltd., Chichester.