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


CRPC-TR99787-S
June 1999

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

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. 2024 (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.
‡ wecor@cs.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. 2024 (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 $[f_i d_i]$, 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:

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

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 $X = \{ x \in (0, 1)^n : \sum_{j=1}^{n} a_{ij} x_j = d_i, \ 1 \leq i \leq m \}$. 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:

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

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

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

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

The Cornújols-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 $1 \leq i \leq m$. Cornújols 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 $2^m$ nodes needed to be evaluated, where $\alpha$ 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 $l$, $u$) integral vector satisfying the system $Ax = d$. 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 $x_d$ satisfying $Ax_d = d$, and $n-m$ linearly independent vectors $x_0$ satisfying $Ax_0 = 0$. Due to the basis reduction algorithm all these vectors are relatively short. If the vector $x_d$ satisfies the bounds, then the algorithm terminates, and if not, one observes that $A(x_d + \lambda x_0) = d$ for any integer multiplier $\lambda$ and any vector $x_0$ such that $Ax_0 = 0$. Hence, one can branch on integer linear combinations of vectors $x_0$ satisfying $Ax_0 = 0$ 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 $X = \{x \in \mathbb{Z}^n : Ax = d, l \leq x \leq u\}$. The relaxation that Aardal et al. consider is $X_{IR} = \{x \in \mathbb{Z}^n : Ax = d\}$. To determine whether $X_{IR}$ is empty can be done in polynomial time. Aardal et
al. rewrite the set $X_{IR}$ as follows:

$$X_{IR} = \{ x \in \mathbb{Z}^n : x = x_d + X_0 \lambda, \lambda \in \mathbb{Z}^{n-m} \},$$

where $x_d \in \mathbb{Z}^n$ satisfies $Ax_d = d$, and where $X_0$ is an integral $n \times (n-m)$ matrix such that the columns $x_d^j, 1 \leq j \leq n-m$, of $X_0$ are linearly independent and such that each column $x_d^j$ satisfies $Ax_d^j = 0$. Note that reformulation (4) is not unique. Expression (4) states that any integer vector $x$ satisfying $Ax = d$ can be described as a vector $x_d$, satisfying $Ax_d = d$, plus an integer linear combination of vectors that form a basis for the lattice \{ $x \in \mathbb{Z}^n : Ax = 0$ \}. Of course this observation also holds for the integer vectors $x$ satisfying the bound constraints, if such vectors exist. Aardal et al. argue that if we are able to find a vector $x_d$ 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 $X_0, X \lambda$, such that $x = x_d + X \lambda$ satisfies the bounds.

One way of obtaining the vectors $x_d$ and $x_d^j, 1 \leq j \leq n-m$, is by using the Hermite normal form of the matrix $A$, 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 $x_d$ 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 $x_d$ and $x_d^j, 1 \leq j \leq n-m$.

Let $I(p)$ denote the $p$-dimensional identity matrix, and let $0(p \times q)$ 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 $B = (b_j)_{1 \leq j \leq n+1}$:

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

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

$$(x^T, N_1 y, N_2(a_1 x - d_1 y), \ldots, N_2(a_m x - d_m y))^T = B \begin{pmatrix} x \\ y \end{pmatrix}, \tag{6}$$

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

**Proposition 1** The integer vector $x_d$ satisfies $Ax_d = d$ if and only if the vector

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

belongs to the lattice $L$, and the integer vector $x_0$ satisfies $Ax_0 = 0$ if and only if the vector

$$(x_0^T, 0, 0^{(1 \times m)})^T = B \begin{pmatrix} x_0 \\ 0 \end{pmatrix}, \tag{8}$$

belongs to the lattice $L$. 

4
Aardal, Hurkens, and Lenstra [1] proved that if there exists an integer vector $\mathbf{x}$ satisfying the system $A\mathbf{x} = \mathbf{d}$, 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 $B$ will be of the following form:

$$
\begin{pmatrix}
X_0^{(n \times (n-m))} & \mathbf{x}_d \\
0_{(1 \times (n-m))} & N_1 \\
0_{(m \times (n-m))} & 0_{(m \times 1)}
\end{pmatrix},
$$

(9)

where $X_0 = (\mathbf{x}_0^1, \ldots, \mathbf{x}_0^{(n-m)})$. Due to Proposition 1 we can conclude that $\mathbf{x}_d$ satisfies $A\mathbf{x}_d = \mathbf{d}$, and that $A\mathbf{x}_0^j = \mathbf{0}$, $1 \leq j \leq n - m$.

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

$$
does\ there\ exist\ a\ vector\ \lambda \in \mathbb{Z}^{(n-m)}\ s.\ t.\ l - \mathbf{x}_d \leq X_0\lambda \leq \mathbf{u} - \mathbf{x}_d ?
$$

(10)

Unless $\mathbf{x}_d$ satisfies the bound constraints, one needs to branch on the variables $\lambda_j$ in order to check whether the polytope $P = \{ \lambda \in \mathbb{Z}^{(n-m)} : l - \mathbf{x}_d \leq X_0\lambda \leq \mathbf{u} - \mathbf{x}_d \}$ 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 $\lambda$-space or linear programming based branch-and-bound. By integral branching in the $\lambda$-space we mean the following. Assume we are at node $k$ of the search tree. Take any unit vector $\mathbf{e}_j$, i.e., the $j$th 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 $u_k = \max \{ \mathbf{e}_j^T \lambda : \lambda \in P \cap \{ \lambda_j \text{'s fixed at predecessors of } k \} \}$ and $l_k = \min \{ \mathbf{e}_j^T \lambda : \lambda \in P \cap \{ \lambda_j \text{'s fixed at predecessors of } k \} \}$. Create $[u_k] - [l_k] + 1$ subproblems at node $k$ of the search tree by fixing $\lambda_j$ to the values $[l_k], \ldots, [u_k]$. 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 $[u_k] - [l_k]$. 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 $[u_k] - [l_k]$ 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 \( \lambda_j \), branching on the unit vector in \( \lambda \)-space that yields the smallest value of \( |u_k| - |l_k| \) 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 \times 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.

![Enumeration tree for integer branching](image)

Figure 1: Enumeration tree for integer branching \( \mathbf{e}_j, j = (n - m), \ldots, 1 \)
Table 1: Results for the feasibility version

<table>
<thead>
<tr>
<th>Inst.</th>
<th>m</th>
<th>n</th>
<th>type</th>
<th>LP B&amp;B on $\lambda$</th>
<th>thinnest $e_j$</th>
<th>$e_j, j = (n - m), \ldots, 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># nodes</td>
<td>time (s)</td>
<td># nodes</td>
<td>time (s)</td>
<td># nodes</td>
<td>time (s)</td>
</tr>
<tr>
<td>M1</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>7,203</td>
<td>40</td>
<td>1,723</td>
</tr>
<tr>
<td>M2</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>10,488</td>
<td>58</td>
<td>3,839</td>
</tr>
<tr>
<td>M3</td>
<td>5</td>
<td>40</td>
<td>Y</td>
<td>5,484</td>
<td>30</td>
<td>1,398</td>
</tr>
<tr>
<td>M4</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>16,484</td>
<td>84</td>
<td>5,893</td>
</tr>
<tr>
<td>M5</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>17,182</td>
<td>94</td>
<td>3,027</td>
</tr>
<tr>
<td>M6</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>11,500</td>
<td>62</td>
<td>2,762</td>
</tr>
<tr>
<td>M7</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>16,666</td>
<td>88</td>
<td>4,025</td>
</tr>
<tr>
<td>M8</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>7,483</td>
<td>42</td>
<td>2,386</td>
</tr>
<tr>
<td>M9</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>6,393</td>
<td>36</td>
<td>1,674</td>
</tr>
<tr>
<td>M10</td>
<td>5</td>
<td>40</td>
<td>N</td>
<td>17,206</td>
<td>90</td>
<td>3,791</td>
</tr>
<tr>
<td>M11</td>
<td>6</td>
<td>50</td>
<td>N</td>
<td>413,386</td>
<td>3,690</td>
<td>88,619</td>
</tr>
<tr>
<td>M12</td>
<td>6</td>
<td>50</td>
<td>N</td>
<td>53,273</td>
<td>428</td>
<td>14,456</td>
</tr>
<tr>
<td>M13</td>
<td>6</td>
<td>50</td>
<td>N</td>
<td>375,654</td>
<td>3,080</td>
<td>86,505</td>
</tr>
<tr>
<td>M14</td>
<td>6</td>
<td>50</td>
<td>N</td>
<td>381,813</td>
<td>2,984</td>
<td>108,725</td>
</tr>
<tr>
<td>M15</td>
<td>6</td>
<td>50</td>
<td>Y</td>
<td>96,470</td>
<td>768</td>
<td>69,981</td>
</tr>
<tr>
<td>M16</td>
<td>6</td>
<td>50</td>
<td>N</td>
<td>114,215</td>
<td>932</td>
<td>40,274</td>
</tr>
<tr>
<td>M17</td>
<td>7</td>
<td>60</td>
<td>N</td>
<td>108,154</td>
<td>1,228</td>
<td>36,288</td>
</tr>
</tbody>
</table>

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 $e_j$” and “$e_j, j = (n - m), \ldots, 1$”, 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 $\lambda_j$ 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 $[u_k]$ and $[l_k]$, are solved using CPLEX version 6.0.1 [3].

An important conclusion of the experiments is that the reformulation itself is essential. Cormuéjols and Dawande [2] could only solve instances up to size $4 \times 30$ 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). Cormuéjols and Dawande did solve feasibility instances of size $6 \times 50$ 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 $7 \times 60$ 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 $\lambda$-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 $\lambda_j$. 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 $e_j$, $j = n - m, \ldots, 1$. 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 $\lambda$-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 $e_j$” in only about 6% of the cases for the instances of size $5 \times 40$. This implies that the unit vectors in $\lambda$-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 $5 \times 40$, 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 adoptions 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 use 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.

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

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 $\lambda_j$. 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 $\mathbf{x}$-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 $\lambda_j$ and minimize the sum of the slack variables expressed in the $\lambda_j$'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 $7 \times 60$ 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 $4 \times 30$, 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 $\lambda_j$, we observe that this number is significantly smaller than the number $2^{an}$ for $a$ 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 $4 \times 30$ they enumerated between $10^6$ and $2 \times 10^6$ nodes. For the same number of enumeration nodes we can solve
<table>
<thead>
<tr>
<th>Inst.</th>
<th>Strat. 1</th>
<th></th>
<th>Strat. 2</th>
<th></th>
<th>Strat. 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># nodes</td>
<td>time (s)</td>
<td># nodes</td>
<td>time (s)</td>
<td># nodes</td>
<td>time (s)</td>
</tr>
<tr>
<td>M1</td>
<td>20,022</td>
<td>57</td>
<td>13,699</td>
<td>38</td>
<td>53,267</td>
<td>180</td>
</tr>
<tr>
<td>M2</td>
<td>6,451</td>
<td>44</td>
<td>75,174</td>
<td>222</td>
<td>109,498</td>
<td>281</td>
</tr>
<tr>
<td>M3</td>
<td>5,484</td>
<td>15</td>
<td>5,230</td>
<td>15</td>
<td>41,518</td>
<td>118</td>
</tr>
<tr>
<td>M4</td>
<td>36,847</td>
<td>107</td>
<td>75,985</td>
<td>211</td>
<td>176,437</td>
<td>456</td>
</tr>
<tr>
<td>M5</td>
<td>32,880</td>
<td>93</td>
<td>16,379</td>
<td>45</td>
<td>271,208</td>
<td>705</td>
</tr>
<tr>
<td>M6</td>
<td>35,710</td>
<td>105</td>
<td>21,277</td>
<td>62</td>
<td>220,603</td>
<td>875</td>
</tr>
<tr>
<td>M7</td>
<td>64,090</td>
<td>180</td>
<td>88,160</td>
<td>254</td>
<td>396,416</td>
<td>1,261</td>
</tr>
<tr>
<td>M8</td>
<td>33,937</td>
<td>99</td>
<td>35,471</td>
<td>103</td>
<td>122,526</td>
<td>383</td>
</tr>
<tr>
<td>M9</td>
<td>9,910</td>
<td>29</td>
<td>19,083</td>
<td>56</td>
<td>200,987</td>
<td>625</td>
</tr>
<tr>
<td>M10</td>
<td>28,402</td>
<td>76</td>
<td>146,224</td>
<td>436</td>
<td>99,502</td>
<td>299</td>
</tr>
<tr>
<td>M11</td>
<td>1,165,498</td>
<td>6,018</td>
<td>1,728,646</td>
<td>7,811</td>
<td>6,158,407</td>
<td>29,486</td>
</tr>
<tr>
<td>M12</td>
<td>53,273</td>
<td>214</td>
<td>73,801</td>
<td>283</td>
<td>5,101,843</td>
<td>22,002</td>
</tr>
<tr>
<td>M13</td>
<td>810,496</td>
<td>505</td>
<td>1,410,840</td>
<td>6,607</td>
<td>6,057,528</td>
<td>26,333</td>
</tr>
<tr>
<td>M14</td>
<td>384,882</td>
<td>1,505</td>
<td>1,117,107</td>
<td>4,748</td>
<td>7,861,402</td>
<td>34,083</td>
</tr>
<tr>
<td>M15</td>
<td>96,470</td>
<td>384</td>
<td>17,007</td>
<td>60</td>
<td>9,558,521</td>
<td>37,960</td>
</tr>
<tr>
<td>M16</td>
<td>282,665</td>
<td>1,190</td>
<td>319,029</td>
<td>1,248</td>
<td>71,253</td>
<td>348</td>
</tr>
<tr>
<td>M17</td>
<td>567,837</td>
<td>3,767</td>
<td>1,823,915</td>
<td>11,597</td>
<td>3,425,941</td>
<td>27,168</td>
</tr>
</tbody>
</table>

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 $7 \times 60$. All these instances were feasible so we decided not to report on the results of the computations here. For the size $6 \times 50$ 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 \( \{0, \ldots, D-1\} \), 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 \( f \). 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 \( S \subseteq \{1, 2, \ldots, n\} \), 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 \( \mathbf{x} \) as defined above satisfies \( A \mathbf{x} = \mathbf{d} \) 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

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

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 \( Y_i(S) + U_i \equiv 0 \) \((\text{mod} 1)\), we have

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

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 \( \mathbf{x} \) induced by \( S \) defines a solution for \( A \mathbf{x} = \mathbf{d} \) depends on \( n, D \) and the size of \( S \) only. The probability that \( S \) induces a solution for \( A \mathbf{x} = \mathbf{d} \) is given by \( q(n, D, |S|)^m \).

Let \( S \) 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.,

\[
\mathbb{E}[\#S] = \sum_{S \subseteq \{1, \ldots, n\}} q(n, D, |S|)^m = \sum_{s=0}^{n} \binom{n}{s} q(n, D, s)^m.
\]

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 \( \mathbb{E}[Y_i(S)] = \frac{1}{2}(D-1)(|S| - nf) \) and \( \text{Var}[Y_i(S)] = \frac{1}{12}(D^2 - 1)(|S|(1 - 2f) + f^2 n) \). 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
\[
\Pr[Z_i(S) = 0] = \Pr[1/(2Q) - 1 \leq Y_i(S) \leq 1/(2Q)].
\]
Using the Central Limit Theorem [5], we approximate this expression by the normal distribution, and find
\[
\Pr \left[ \frac{1}{2Q} - 1 < Y_i(S) < \frac{1}{2Q} \right] \approx \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\beta} \exp\left(-\frac{1}{2}u^2\right) du ,
\]
with
\[
\alpha = \frac{\frac{1}{2Q} - 1 - \mathbb{E}[Y_i(S)]}{\sqrt{\text{Var}[Y_i(S)]}}, \quad \text{and} \quad \beta = \frac{\frac{1}{2Q} - \mathbb{E}[Y_i(S)]}{\sqrt{\text{Var}[Y_i(S)]}}.
\]
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
\[
G_{a_{ij}}(x) = E[x^{a_{ij}}] = \sum_{k=0}^{D-1} \frac{1}{D} x^k = \frac{1}{D} (1 + x + \cdots + x^{D-1}) = \frac{1}{D} \cdot \frac{x^D - 1}{x - 1}.
\]

Since the coefficients \(a_{ij}\) are independent, the probability generating function of \(Y_i(S)\) is given by
\[
G_{Y_i(S)}(x) = E[x^{Y_i(S)}] = E[x^{\sum_{j \in S} (1-f)a_{ij} - \sum_{j \not\in S} fa_{ij}}]
\]
\[
= \prod_{j \in S} E[x^{(1-f)a_{ij}}] \prod_{j \not\in S} E[x^{-fa_{ij}}]
\]
\[
= \prod_{j \in S} \left( \frac{1}{D} \cdot \frac{x^{(1-f)D} - 1}{x^{(1-f)} - 1} \right) \prod_{j \not\in S} \left( \frac{1}{D} \cdot \frac{x^{-fD} - 1}{x^{-f} - 1} \right)
\]
\[
= \left( \frac{1}{D} \right)^{|S|} \left( \frac{x^{(1-f)D} - 1}{x^{(1-f)} - 1} \right)^{|S|} \left( \frac{x^{-fD} - 1}{x^{-f} - 1} \right)^{|n| - |S|}
\]
\[
= \frac{1}{D^n} \frac{1}{x^{f(D-1)(n-|S|)}} \left( \frac{x^{(1-f)D} - 1}{x^{(1-f)} - 1} \right)^{|S|} \left( \frac{x^{-fD} - 1}{x^{-f} - 1} \right)^{|n| - |S|}.
\]

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

In order to find the Taylor expansion of the factors in (18) and (19) we use the following lemma.
Lemma 2 \[ \left( \frac{yD-1}{y-1} \right)^N = \sum_{j=0}^{\infty} a_j y^j \] with
\[ a_j = \sum_{k=0}^{\min\{N,[j/D]\}} \binom{N}{k} (-1)^k \left( \frac{j - Dk + N - 1}{j - Dk} \right). \]  

Proof: Write \( \left( \frac{yD-1}{y-1} \right)^N \) as the product of \((1 - y^D)^N\) and \((1 - y)^{-N}\) as \( (1 - y^D)^N = \sum_{j=0}^{N} \binom{N}{j} (-1)^j y^D \) 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 \( (\frac{yD-1}{y-1})^N = (1 + y + \cdots + y^{D-1})^N \).  

For \( f = \frac{1}{2} \), we apply the lemma with \( N = n \) and \( y = x^{1/2} \), to obtain
\[ \Pr[Z_i(S) = 0] = q(n, D, |S|) = \frac{1}{D^n} (a_{(D-1)(n-|S|)} + a_{(D-1)(n-|S|)-1}). \]  

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
\[ \Pr[\#S = 0] = \Pr[S \notin S, \forall S \subset \{1,2,\ldots,n\}] \]
\[ \approx \prod_{S \subset \{1,2,\ldots,n\}} \Pr[S \notin S] \]
\[ = \exp\left( \log \prod_{S \subset \{1,2,\ldots,n\}} (1 - q(n, D, |S|)^m) \right) \]
\[ = \exp\left( \sum_{S \subset \{1,2,\ldots,n\}} \log(1 - q(n, D, |S|)^m) \right) \]
\[ \approx \exp\left( \sum_{S \subset \{1,2,\ldots,n\}} -q(n, D, |S|)^m \right) \]
\[ = \exp(-E[\#S]). \]  

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 \( \tilde{S} = \{1,\ldots,n\} \setminus S \) does. We first compute the conditional probability that \( S \) does not induce a solution, given that \( \tilde{S} \) does not for the first \( k \) rows. Let \( S_i \) denote the set of subsets that induce a solution with respect to row \( i \) and let \( q_S = q(n, D, |S|) \).
\[
\Pr \left[ S \in \mathcal{S} \mid (\forall i : 1 \leq i \leq k, \tilde{S} \not\in \mathcal{S}_i) \land (\forall i : k + 1 \leq i \leq m, \tilde{S} \in \mathcal{S}_i) \right] \\
= \Pr \left[ \exists i : 1 \leq i \leq k, \sum_j a_{ij} \text{ even} \right] + \Pr \left[ \forall i : 1 \leq i \leq k, \sum_j a_{ij} \text{ odd} \right] \cdot \left\{ \right. \\
\Pr \left[ \exists i : k + 1 \leq i \leq m, \sum_j a_{ij} \text{ odd} \right] \cdot 1 + \\
\Pr \left[ \forall i : k + 1 \leq i \leq m, \sum_j a_{ij} \text{ even} \right] \cdot \left( 1 - \left( \frac{q_S}{1 - q_S} \right)^k \right) \\
= (1 - 2^{-k}) + 2^{-k}(1 - 2^{-(m-k)}) + 2^{-k}2^{-(m-k)}\left( 1 - \left( \frac{q_S}{1 - q_S} \right)^k \right) \\
\approx 1 - 2^{-m}q_S^k. \quad (23)
\]

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

\[
\Pr [S, \tilde{S} \not\in \mathcal{S}] \approx \sum_{k=1}^{m} \binom{m}{k} (1 - q_S)^{q_S^{m-k}}(1 - 2^{-m}q_S^k) \\
= 1 - q_S^m + \left( \frac{q_S}{2} \right)^m - \left( \frac{q_S + q_S - q_Sq_S}{2} \right)^m. \quad (24)
\]

Since \( q_S \approx q_S \), we obtain

\[
\Pr[\#S = 0] \approx \prod_{S \subseteq \mathcal{S}} \Pr[S, \tilde{S} \not\in \mathcal{S}] \\
= \exp \sum_{S \subseteq \mathcal{S}} \log \Pr[S, \tilde{S} \not\in \mathcal{S}] \\
\approx \exp \sum_{S \subseteq \mathcal{S}} -q_S^m\left(1 - \frac{1}{2^m} + (1 - \frac{q_S}{2})^m\right) \\
\approx \exp \left( \sum_{S \subseteq \{1, \ldots, n\}} (-q_S^m)(1 - \frac{1}{2^m}) \right) \\
= \exp \left( -E[\#S](1 - \frac{1}{2^m+1}) \right) =: \tilde{p}_{m,n}. \quad (25)
\]

Note that for large values of \( E[\#S] \) 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 \(i\)th 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 \(p_{m,n}\) \((T \sim \text{Bin}(N, p_{m,n}))\). Our null hypothesis is \(p_{m,n} = \tilde{p}_{m,n}\). Under this assumption \(T \sim \text{Bin}(N, \tilde{p}_{m,n})\). By looking at the 2.5% quantile and the 97.5% quantile, we construct a 95% confidence interval for \(T\):

\[
q_{0.025}(N, \tilde{p}_{m,n}) \leq T \leq q_{0.975}(N, \tilde{p}_{m,n}).
\]

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 \(\bar{p}_{m,n} = t_{m,n}/N\). 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>
<thead>
<tr>
<th>$n$</th>
<th>gen approx</th>
<th>gen approx</th>
<th>gen approx</th>
<th>gen approx</th>
<th>gen approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.0000</td>
<td>0.0000</td>
<td>2.0210e-05</td>
<td>2.3952e-05</td>
<td>1.3517e-05</td>
</tr>
<tr>
<td>21</td>
<td>0.0007</td>
<td>0.0008</td>
<td>4.0743e-05</td>
<td>4.1004e-05</td>
<td>2.2294e-05</td>
</tr>
<tr>
<td>15</td>
<td>0.0410</td>
<td>0.0411</td>
<td>7.2870e-06</td>
<td>7.4300e-06</td>
<td>3.0082e-06</td>
</tr>
<tr>
<td>10</td>
<td>0.0040</td>
<td>0.0040</td>
<td>1.0000e-06</td>
<td>1.0000e-06</td>
<td>6.8390e-06</td>
</tr>
<tr>
<td>5</td>
<td>0.0211</td>
<td>0.0212</td>
<td>2.1255e-07</td>
<td>2.1261e-07</td>
<td>3.2171e-07</td>
</tr>
<tr>
<td>1</td>
<td>0.1511</td>
<td>0.1512</td>
<td>3.8308e-08</td>
<td>3.9177e-08</td>
<td>1.9192e-09</td>
</tr>
<tr>
<td>0.0030</td>
<td>0.0030</td>
<td>1.0000e-08</td>
<td>1.0000e-08</td>
<td>6.8390e-06</td>
<td></td>
</tr>
<tr>
<td>0.0103</td>
<td>0.0103</td>
<td>3.2050e-06</td>
<td>2.3541e-06</td>
<td>1.0449e-06</td>
<td></td>
</tr>
<tr>
<td>0.0193</td>
<td>0.0193</td>
<td>3.9220e-06</td>
<td>4.0789e-06</td>
<td>1.8561e-06</td>
<td></td>
</tr>
<tr>
<td>0.2000</td>
<td>0.2000</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.1200</td>
<td>0.1200</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0500</td>
<td>0.0500</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0100</td>
<td>0.0100</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0050</td>
<td>0.0050</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0010</td>
<td>0.0010</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0005</td>
<td>0.0005</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0001</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.00001</td>
<td>0.00001</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
<tr>
<td>0.000001</td>
<td>0.000001</td>
<td>7.2871e-06</td>
<td>7.3787e-06</td>
<td>3.1396e-05</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3:** The expected number of solutions computed exactly using the probability generating function (gen) and approximated by the normal distribution (approx)
Table 4: The probability of generating an infeasible instance

| m | n | E[♯$|$] | $\bar{p}$ | $q_{0.025}$ | $q_{0.975}$ | $l$ | $\bar{p}$ |
|---|---|---------|---------|-----------|-----------|-----|---------|
| 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


17


[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


