Exact cost minimization of a series-parallel system

F. Castro, J. Gago, I. Hartillo, J. Puerto, J.M. Ucha

June 22, 2018

In memory of our fellow Alejandro Fernández-Margarit

Abstract

The redundancy allocation problem is formulated minimizing the design cost for a series-parallel system with multiple component choices whereas ensuring a given system reliability level. The obtained model is a nonlinear integer programming problem with a non linear, non separable constraint. We propose an algebraic method, based on Gröbner bases, to obtain the exact solution of the problem. In addition, we provide a closed form for the required Gröbner bases, avoiding the bottleneck associated with the computation, and promising computational results.

1 Introduction

System reliability is considered an important measure in the engineering design process. A series system is like a chain composed of links, each of them representing a subsystem. The failure of one of these components means the failure of the whole system. In order to avoid this, it is usual to use redundant components in parallel to guarantee a certain level of reliability. These systems are called series-parallel systems.

Determining the optimal number of components in each subsystem is the so called reliability optimization problem. Two different approaches are usual:

- maximize system reliability subject to system budget constraint, or
- minimize system cost subject to a required level of reliability.

Both problems are nonlinear integer programming problems, and they are NP-hard [5]. There are very few papers looking for their exact solutions, due to the difficulty of the problems. Those works use essentially Dynamic Programming [14], branch and bound methods [9], or Lagrangian relaxation [11], among others techniques.

On the contrary, in the literature there are many heuristics and metaheuristic algorithms; such as those based in Genetic Algorithms [6], Tabu Search [10] or Ant Colony Optimization [2], among others.

In this paper we study the exact solution of one of the versions of the problem that minimizes the cost function of the chosen design, subject to a non linear constraint which describes the reliability of the considered system. For a fixed subsystem its inner components can be considered equal, as in [9], or different, as in [11]. If the components are equal the reliability function is separable and convex, and the problem can be reduced to a linear knapsack problem [9]. In the case of multiple component choices the reliability function is no longer separable. In [14], the solution is found using dynamic programming methods. That approach presents two stages; in the first one the problem is restricted to each subsystem, with a level of reliability. Under this assumption the reliability function is separable, and the optimization problem can be reduced to a knapsack problem. Then the reliability levels of the subsystems are determined by a new dynamic programming process.

The solution method shown in [11] uses an algorithm based on Lagrangian relaxations over two linear relaxations of the original problem. The first relaxation consists of deleting the non-linear reliability function, and adding certain linear constraints, one for each subsystem. The second relaxation assumes that the same type of component is going to be used in every subsystem, so that the problem has the form as in [9].

Mainly, the algorithm of [11] is a what their authors called a *cut and partition scheme* (a geometric branch and bound). The solution space is partitioned in boxes, which are divided and discarded for certain conditions. The cuts are built from the best bound feasible solution of some Lagrangian relaxations. Such bounds allow to remove certain boxes depending on the improvement with respect to the current best point.

We address the problem via a different approach based on Gröbner bases. As introduction on this subject, we recommend the text books [1], [3] and [8].

Gröbner bases were applied to Integer Linear Programming, by the first time, in [7]. Later, Tayur et al. [13] introduced a new application framework, which solves nonlinear integer programming problems, with a linear objective function. This is exactly our framework, as in [4].

First, we consider a relaxed integer programming problem where all the restrictions are linear. Then we find the solution of the relaxed problem by computing a test set. By using the so called reverse test set, we can solve the complete problem, generating paths from the solution of the relaxed problem to a solution of the complete one. These paths increase the cost function at each step.

A test set for a linear integer programming problem is a set of directions that

can be used to design descending algorithms with respect to a linear cost function. A test set can be computed from a Gröbner basis of the toric ideal associated with the linear restrictions, with respect to an order given by the cost function.

One of the main tasks in the process described before is usually the calculation of the Gröbner basis. We construct a linear programming problem from the original one, removing the reliability function, and adding a new linear restriction. This constraint is obtained computing a feasible solution with a greedy algorithm. For the relaxed linear programming problem obtained in this way we explicitly give the associated Gröbner basis, and so the test set to solve the main problem. We point out here that any Gröbner basis computation is done using closed formulas, thus avoiding the hard computation burden of reduction algorithms to compute Gröbner bases.

The organization of the paper is as follows. In Section 2 we introduce the notation and describe the model of a parallel-series system with multiple component choices. In Section 3, a greedy algorithm is described to compute a feasible point. Section 4 is devoted to a brief introduction to the essential facts about Gröbner bases. Section 5 contains the main result about the closed formula for the test set of the integer linear problem. Our computational experiments are reported in Section 6. Finally we draw some concluding remarks in Section 7.

2 General model

In order to formulate the problem, some notation is first introduced.

- n number of subsystems.
- k_i number of different types of available components for the i-th subsystem, $i = 1,\dots,n$.
- r_{ij} reliability of the j-th component for the i-th subsystem, $i = 1,\dots,n,\ j = 1,\dots,k_i$.
- c_{ij} cost of the j-th component for the i-th subsystem, $i = 1,\dots,n,\ j = 1,\dots,k_i$.
- l_{ij}, u_{ij} lower/upper bounds of number of j components for the i-th subsystem, $i = 1,\dots,n,\ j = 1,\dots,k_i$.
- R_0 admissible level of reliability of the whole system.
- x_{ij} number of j components used in the i-th subsystem, $i = 1,\dots,n,\ j = 1,\dots,k_i$.

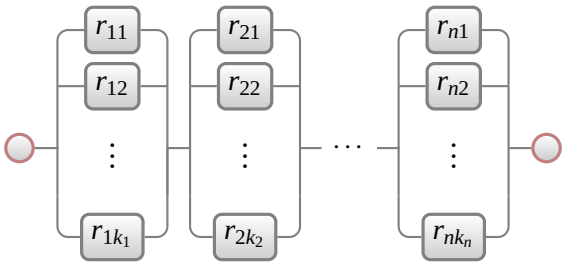

Figure 1: A series-parallel system with multiple choice components

In our model, some assumptions are considered:

- Components have two states: working or failed.
- The reliability of each component is known and is deterministic.
- Failure of individual components are independent.
- Failed components do not damage other components or the system, and they are not repaired.

This model is illustrated in Figure 1. It is a system with n subsystems with the notation introduced before. The optimization problem can be formulated as:

$$
\begin{array}{lll}
(RP) \quad \min & \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} & \\
\text{s. t.} & & \\
& R(x) \geq R_0, & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1,\ldots,n, \\
& 0 \leq l_{ij} \leq x_{ij} \leq u_{ij}, & i = 1,\ldots,n, \\
& & j = 1,\ldots,k_i, \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i,j,
\end{array}
\tag{1}
$$

where $R(x) = \prod_{i=1}^{n}(1-\prod_{j=1}^{k_i}(1-r_{ij})^{x_{ij}})$. The first n linear inequalities in (1) assert that each subsystem must have, at least, one component.

As usual, we can make a change of variables $y_{ij} = x_{ij} - l_{ij}$, so that we can assume $l_{ij} = 0$. This does not alter the equations of (RP), and some of the last equations can be redundant. Hence it can be assumed $l_{ij} = 0$ without loss of gen-

erality, and:

$$
\begin{array}{lll}
(RP) \quad \min & \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij} & \\
\text{s. t.} & & \\
& R(x) \geq R_0, & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1, \ldots, n. \\
& 0 \leq x_{ij} \leq u_{ij}, & i = 1, \ldots, n, \\
& & j = 1, \ldots, k_i, \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i, j.
\end{array}
$$

A feasible solution is sometimes called a reliable solution because it ensures a reliability greater than or equal to R_0.

3 Computing a reliable system with a greedy procedure

The main step used in the algebraic algorithm described in this article is to consider an integer linear programming problem (LRP), relaxed from the original problem (RP). There is only one nonlinear constraint in (RP), the equation which ensures the reliability of the whole system. Removing the nonlinear constraint we get an integer linear programming problem:

$$
\begin{array}{lll}
(LRP1) \quad \min & \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij} & \\
\text{s. t.} & & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1, \ldots, n. \\
& 0 \leq x_{ij} \leq u_{ij}, & i = 1, \ldots, n, \\
& & j = 1, \ldots, k_i, \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i, j.
\end{array}
$$

In our solution technique, we start from the solution of the linear programming problem $(LRP1)$, and following the directions given by the test set of $(LRP1)$ we follow a descent path to the solution of the complete problem (RP). If the linear relaxation is too weak, the paths to be followed to get to the optimal solution of (RP) will be very long, and the number of points to be processed is huge.

To avoid this problem we add a new linear equation. We need a feasible point y^0 of (RP), and there are lots of heuristic methods to obtain such a point. In our case, we use a greedy algorithm similar to [9] or [11].

At the beginning of the greedy algorithm, y^0 describes the system with the maximum number of components of every type. If the reliability of that system is less than R_0, the problem is unfeasible. We consider I the set of all pairs (i, j), which describes the j-th component for the i-th subsystem. For each (i, j), we calculate

the rate $t_{ij} = \frac{c_{ij}}{-\log(1-r_{ij})}$ between cost and reliability, and order I non increasingly by these rates (ties are solved by lex order, for example). For the first index (i_0, j_0) in I, we subtract components of type (i_0, j_0) from y^0 until it is non reliable, there is no such component or the i_0-th subsystem is empty. If the solution obtained by this process is non reliable, or the i_0-th subsystem is empty, one (i_0, j_0) component is added. Then we take the next index in the set I and repeat the procedure, until the index set I has been completely processed.

Data: r_{ij}, vector $\boldsymbol{c}$
Result: $\boldsymbol{y}^0$ feasible point
$\boldsymbol{y}^0 = (u_{11}, \ldots, u_{nk_n})$
$t = \left(\frac{c_{11}}{-\log(1-r_{11})}, \ldots, \frac{c_{nk_n}}{-\log(1-r_{nk_n})}\right)$
$I = \{(1,1), \ldots, (1,k_1) \ldots, (n,k_n)\}$
Order I non increasingly by $t_{i,j}$
forall the $(i,j) \in I$ **do**
  Reliable=TRUE
  SubsystemNonEmpty=TRUE
  while *Reliable* **and** *SubsystemNonEmpty* **and** $y^0_{i,j} > 0$ **do**
    $y^0_{i,j} = y^0_{i,j} - 1$
    if $\sum_k y^0_{ik} < 1$ **then**
      SubsystemNonEmpty=FALSE
    end
    if $R(y^0) < R_0$ **then**
      Reliable=FALSE
    end
    if *Reliable=FALSE* **or** *SubsystemNonEmpty=FALSE* **then**
      $y^0_{ij} = y^0_{ij} + 1$
    end
  end
end

Algorithm 1: Greedy algorithm

Using the above greedy algorithm, we obtain a feasible point y^0, with a cost $\sum_{ij} c_{ij} y^0_{ij} = c^0$. The optimal solution of (RP) has a cost less than or equal to c^0, so we can add to (RP) a valid inequality stating this condition and the problem has an

equivalent form:

$$
\begin{array}{lll}
(RP) \quad \min & \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} & \\
\text{s. t.} & & \\
& R(x) \geq R_0, & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1,\ldots,n, \\
& 0 \leq x_{ij} \leq u_{ij}, & i = 1,\ldots,n, \\
& & j = 1,\ldots,k_i, \\
& \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} \leq c^0, & \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i,j.
\end{array}
$$

From this formulation, we have the new integer linear problem

$$
\begin{array}{lll}
(LRP) \quad \min & \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} & \\
\text{s. t.} & & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1,\ldots,n, \\
& 0 \leq x_{ij} \leq u_{ij}, & i = 1,\ldots,n, \\
& & j = 1,\ldots,k_i, \\
& \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} \leq c^0, & \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i,j.
\end{array}
$$

4 A review on integer programming and Gröbner bases

In this section, we recall the concepts and algorithms used to solve Integer Linear Programming problems from an algebraic point of view, and the walk back procedure for nonlinear integer programming problems based on test sets. To this end, we have followed [12] and [13].

4.1 Gröbner bases

Denote by $k[\boldsymbol{x}] = k[x_1,\ldots,x_N]$ the ring of polynomial with coefficients in a field k. In our case, k will be $\mathbb{R}$. The ideal generated by a subset $\mathscr{F} \subset k[\boldsymbol{x}]$ is the set $\langle \mathscr{F} \rangle$ consisting of all linear combinations:

$$\langle \mathscr{F} \rangle = \{h_1 f_1 + \cdots + h_r f_r \ : \ f_1,\ldots,f_r \in \mathscr{F}, h_1,\ldots,h_r \in k[\boldsymbol{x}]\}.$$

A term order on $\mathbb{N}^N$ is a total order $\prec$ satisfying the following properties:

- $\prec$ is compatible with sums, i.e., $\alpha \prec \beta \Rightarrow \alpha+\gamma \prec \beta+\gamma$, for all $\alpha,\beta,\gamma \in \mathbb{N}^N$.

- $\prec$ is a well-ordering, i.e., $0 \prec \alpha$ for all $\alpha \in \mathbb{N}^N$, $\alpha \neq 0$.

If we fix a term order $\prec$, then every non zero polynomial f has a unique initial term $\text{in}_\prec(f) = a\boldsymbol{x}^\alpha$. It is the monomial $a\boldsymbol{x}^\alpha$ where α is the largest term appearing in f for the term order $\prec$. We are particularly interested in two term orders:

1. The lexicographic order $<_{\text{lex}}$. For every $\alpha, \beta \in \mathbb{N}^N$, we say $\alpha >_{\text{lex}} \beta$ if, in the vector difference $\alpha - \beta \in \mathbb{Z}^N$, the leftmost nonzero entry is positive.

2. The vector induced order $<_{\boldsymbol{c}}$. We consider a vector $\boldsymbol{c} \in \mathbb{N}^N$. Given $\alpha, \beta \in \mathbb{N}^N$, we say $\alpha >_{\boldsymbol{c}} \beta$ if

$$\boldsymbol{c}^t\alpha > \boldsymbol{c}^t\beta \text{ or } \boldsymbol{c}^t\alpha = \boldsymbol{c}^t\beta, \text{ and } \alpha >_{\text{lex}} \beta.$$

For example, consider the polinomial $f = 6x_1x_2^2x_3 + 7x_3^2 - 5x_1^3 + 4x_1^2x_3^2$ and the vector $\boldsymbol{c} = (3,2,2)^t$. Then

$$\text{in}_{<_{\text{lex}}}(f) = -5x_1^3, \text{in}_{<_{\boldsymbol{c}}}(f) = 4x_1^2x_3^2.$$

Of course, we can reorder the variables x_i, and get a new term order. In general, the notation $<_{\boldsymbol{c}}$ means a term order which respect the partial order defined by the vector $\boldsymbol{c}$ and then a tie-break term order, so if we change the lexicographic order in the definition on $<_{\boldsymbol{c}}$, we get another vector induced order.

Suppose that J is an ideal in $k[\boldsymbol{x}]$, and $\prec$ is a given term order. Then its initial ideal is the ideal generated by the initial terms of the polynomials in J:

$$\text{in}_\prec(J) = \langle \text{in}_\prec(f) \ : \ f \in J \rangle.$$

A finite subset $\mathscr{G}$ of J is a Gröbner basis with respect to the term order $\prec$ if the initial terms of the elements in $\mathscr{G}$ suffice to generate the initial ideal:

$$\text{in}_\prec(J) = \langle \text{in}_\prec(g) \ : \ g \in \mathscr{G} \rangle.$$

Fixed an ideal and a term order, a Gröbner basis is not unique. Adding two more conditions, the uniqueness is guaranteed. The reduced Gröbner basis of J with respect to $\prec$ is a Gröbner basis $\mathscr{G}_\prec$ of J such that:

- $\text{in}_\prec(g_i)$ has unit coefficient for each $g_i \in \mathscr{G}_\prec$.

- For each $g_i \in \mathscr{G}_\prec$, no monomial in g_i lies in $\langle \text{in}_\prec(\mathscr{G}_\prec \backslash g_i) \rangle$.

Every ideal J has a unique reduced Gröbner basis for each term order.

4.2 Test set

Consider a linear programming problem:

$$\begin{array}{lll} LP(\boldsymbol{b}) & \min & \boldsymbol{c}^t \cdot \boldsymbol{x} \\ & \text{s. t.} & \\ & & A \cdot \boldsymbol{x} = \boldsymbol{b}, \\ & & \boldsymbol{x} \in \mathbb{Z}_+^N, \end{array}$$

where $A \in \mathbb{Z}^{d \times N}, \boldsymbol{b} \in \mathbb{Z}^d, \boldsymbol{c} \in \mathbb{R}^N$. The notation $LP(\boldsymbol{b})$ denotes the linear programming problem with the right-hand-side restrictions fixed to $\boldsymbol{b}$. When we write (LP), we note all integer programming problems, obtained by varying the right-hand-side vector $\boldsymbol{b}$, fixing A and the cost function $\boldsymbol{c}$. Consider the map $\pi : \mathbb{N}^N \to \mathbb{Z}^d$ defined by $\pi(\boldsymbol{x}) = A\boldsymbol{x}$. Given a vector $\boldsymbol{b} \in \mathbb{Z}^d$, the set $\pi^{-1}(\boldsymbol{b}) = \{\boldsymbol{u} \in \mathbb{N}^N \ : \ \pi(\boldsymbol{u}) = \boldsymbol{b}\}$ is the fiber of (LP) over $\boldsymbol{b}$.

We group points in $\mathbb{N}^N$ according to increasing cost value $\boldsymbol{c}^t\boldsymbol{x}$, and refine this order to a total order $<_{\boldsymbol{c}}$ breaking ties among points with the same cost value by adopting some term order (lexicographic, for example, as defined in the previous section). It is the vector induced order.

A set $G_{<_{\boldsymbol{c}}} \subset \mathbb{Z}^N$ is a test set for the family of integer problems (LP) with respect to the matrix A and the order $<_{\boldsymbol{c}}$ if

- for each nonoptimal point α in each fiber of (LP), there exists $g \in G_{<_{\boldsymbol{c}}}$ such that $\alpha - g$ is a feasible solution in the same fiber and $\alpha - g <_{\boldsymbol{c}} \alpha$,
- for the optimal point β in a fiber of (LP), $\beta - g$ is unfeasible for every $g \in G_{<_{\boldsymbol{c}}}$

A test set for (LP) gives an obvious algorithm to solve an integer program, provided we know a feasible solution to this problem. At every step of this algorithm, we have two different cases:

- There exists an element in the test set which, when subtracted from the current point, yields an improved point. We are then in a nonoptimal point, but we get a better one.
- There will not exist such an element in the set, so we are in the optimum of the fiber.

4.3 Toric ideal

We define I_A the toric ideal associated with A as

$$I_A = \langle \boldsymbol{x}^\alpha - \boldsymbol{x}^\beta \ : \ A\alpha = A\beta, \alpha, \beta \in \mathbb{N}^N \rangle.$$

Given an integral vector $\gamma \in \mathbb{Z}^N$, we can write it uniquely as $\gamma = \gamma^+ - \gamma^-$, where $\gamma^+, \gamma^- \in \mathbb{N}^N$ and have disjoint supports. It is well known ([12]) that

$$I_A = \langle \boldsymbol{x}^{\alpha^+} - \boldsymbol{x}^{\alpha^-} \ : \ A\alpha = \mathbf{0}, \alpha \in \mathbb{Z}^N \rangle.$$

The relationship between the previous concepts is that the reduced Gröbner basis $\mathscr{G}_{<_{\boldsymbol{c}}}$ of I_A with respect to the order $<_{\boldsymbol{c}}$ allows us to compute a uniquely defined minimal test set $G_{<_{\boldsymbol{c}}}$ for (LP). The reduced Gröbner basis is formed by binomials

$$\mathscr{G}_{<_{\boldsymbol{c}}} = \{\boldsymbol{x}^{\alpha_i} - \boldsymbol{x}^{\beta_i}, i = 1, 2, \ldots, r\}, \text{ with } \mathrm{in}_{<_{\boldsymbol{c}}}(\boldsymbol{x}^{\alpha_i} - \boldsymbol{x}^{\beta_i}) = \boldsymbol{x}^{\alpha_i},$$

and then the test set is expressed as

$$G_{<_{\boldsymbol{c}}} = \{\alpha_i - \beta_i, i = 1, 2, \ldots, r\}.$$

4.4 Walk back procedure

Basically the walk back procedure gives an algorithm which computes the optimum for a nonlinear integer programming problem under some conditions. The integer programming problem (RP) introduced in Section 2 is not linear. It has a nonlinear constraint (the reliability condition), while the rest of the restrictions are linear and the cost function is also linear. These are the conditions required to use the walk back procedure, introduced in [13]. In Algorithm 2, it is used the directed graph defined by the Gröbner basis over the feasible points, but directions are reversed in the skeleton. In each step, elements $w = \alpha + g$ in the reverse skeleton are computed, where $A\alpha = \mathbf{0}$ and g is an element in the Gröbner basis.

In general, Algorithm 2, uses the following notation. We denote by (RP) the entire non linear integer programming problem, (LRP) the relaxed linear integer programming problem which arises from (RP). Let β be the optimum of (LRP). If β is feasible for (RP), then it is the solution to (RP). If it is non feasible, then the reverse skeleton is needed.

Let $P(\alpha)$ denote the path, in the directed graph (reversed) of the linear integer programming Problem (LRP), from the optimum β for (LRP) to a feasible point α for (LP). There is always one. Any solution of (RP) is feasible for (LRP), so the objective is to find such a path, in an ordered way. In each reversed step the cost function increases, so the minimum cost feasible points for (RP) are found first.

5 The test set for the relaxed linear problem

Once the (LRP) problem is reinforced by means of the linear constraint that comes after a feasible solution of (RP) is found by the greedy algorithm, the relaxed linear problem is:

Data: Matrix A, vectors $\boldsymbol{b}, \boldsymbol{c}$, non-linear restrictions
Result: Optimum
$\mathscr{P} = \{P(\beta)\}$
$y^0 = \texttt{greedy}(RP)$
$Y = \{y^0\}$
$\mathscr{G}_{<_{\boldsymbol{c}}} = \texttt{groebner}(I_A)$ with respect to $<_{\boldsymbol{c}}$
$\beta =$ optimum for relaxed *LRP*
repeat
  forall the $P(\alpha) \in \mathscr{P}$ **do**
    forall the $g \in \mathscr{G}_{<_{\boldsymbol{c}}}$ **do**
      $w = \alpha + g$
      if *w is a feasible point of (LRP)* **then**
        if *w is feasible for (RP)* **then**
          $Y = Y \cup \{w\}$
          Prune $P(w)$
        else
          if $y <_{\boldsymbol{c}} w$ *for some* $y \in Y$ **then**
            Prune $P(w)$
          end
          $\mathscr{P} = \mathscr{P} \cup \{P(w)\}$
        end
      end
    end
  end
  Delete $P(\alpha)$ from $\mathscr{P}$
until *all paths in* $\mathscr{P}$ *are pruned*
Optimum = Select minimum $<_{\boldsymbol{c}}$ element from Y

Algorithm 2: Walk back procedure

$$
\begin{array}{lll}
(LRP) \quad \min & \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} & \\
\text{s. t.} & & \\
& \sum_{j=1}^{k_i} x_{ij} \geq 1, & i = 1,\ldots,n, \\
& 0 \leq x_{ij} \leq u_{ij}, & i = 1,\ldots,n, \\
& & j = 1,\ldots,k_i, \\
& \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} \leq c^0, & \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i,j.
\end{array}
$$

Each inequality must be converted to an equality, so we must introduce a new slack variable for each inequality:

$$
\begin{array}{lll}
(LRP) \quad \min & \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} & \\
\text{s.t.} & & \\
& \sum_{j=1}^{k_i} x_{ij} - d_i = 1, & i = 1,\ldots,n, \\
& x_{ij} + t_{ij} = u_{ij}, & i = 1,\ldots,n, \\
& & j = 1,\ldots,k_i, \\
& \sum_{i=1}^{n}\sum_{j=1}^{k_i} c_{ij}x_{ij} + b = c^0, & \\
& x_{ij} \in \mathbb{Z}_+ & \text{for all } i,j.
\end{array}
$$

If we put $N = k_1 + \ldots + k_n$ and

$$
D_{n\times N} = \begin{pmatrix}
\overbrace{1\ldots1}^{k_1} & \overbrace{0\ldots0}^{k_2} & \ldots & \overbrace{0\ldots0}^{k_n} \\
0\ldots0 & 1\ldots1 & \ldots & 0\ldots0 \\
 & & \ddots & \\
0\ldots0 & 0\ldots0 & \ldots & 1\ldots1
\end{pmatrix}
$$

the restrictions in matrix form can be written as

$$
\begin{pmatrix}
D & -I_n & \mathbf{0}_{n\times N} & \mathbf{0}_{n\times 1} \\
I_N & \mathbf{0}_{N\times n} & I_N & \mathbf{0}_{N\times 1} \\
\mathbf{c}_{1\times N} & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times N} & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
\boldsymbol{x}_{N\times 1} \\ \boldsymbol{d}_{n\times 1} \\ \boldsymbol{t}_{N\times 1} \\ b
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{1}_n \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_n \\ c^0
\end{pmatrix},
$$

where

$$\mathbf{c}_{1\times N} = \begin{pmatrix} c_{11} & \dots & c_{1k_1} & \dots & c_{n1} & \dots & c_{nk_n} \end{pmatrix},$$

$$\boldsymbol{x} = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1k_1} \\ \vdots \\ x_{n1} \\ \vdots \\ x_{nk_n} \end{pmatrix}, \boldsymbol{t} = \begin{pmatrix} t_{11} \\ \vdots \\ t_{1k_1} \\ \vdots \\ t_{n1} \\ \vdots \\ t_{nk_n} \end{pmatrix}, \boldsymbol{d} = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix}, \boldsymbol{u}_i = \begin{pmatrix} u_{i1} \\ \vdots \\ u_{ik_i} \end{pmatrix}, i = 1, \dots, n,$$

and $\mathbf{1}_n$ denotes the n-vector with all the componentes equal to 1. We can assume that, for each $i = 1, \dots, n$, the costs c_{ij} are ordered in descending order: $c_{iq} \geq c_{ip}$ if $q < p$. Let

$$\boldsymbol{z} = (x_{11}, \dots, x_{nk_n}, d_1, \dots, d_n, t_{11}, \dots, t_{nk_n}, b) = (\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{t}, b),$$

and consider the following set of binomials in $k[\boldsymbol{z}]$:

$$\mathscr{G} = \{\underline{x_{ik} d_i} - t_{ik} b^{c_{ik}}, \underline{x_{iq} t_{ip}} - x_{ip} t_{iq} b^{c_{iq} - c_{ip}}\},$$

for $i = 1, \dots, n, k = 1, \dots, k_i, 1 \leq q < p \leq k_i$. Let $>$ be a term order in $k[\boldsymbol{z}]$ such that $\boldsymbol{x} > \boldsymbol{d} > \boldsymbol{t} > b$. Within each block, the variables are sorted lexicographically as follows:

$$x_{11} > \cdots > x_{1k_1} > x_{21} > \cdots > x_{nk_n}, t_{11} > \cdots > t_{1k_1} > t_{21} > \cdots > t_{nk_n}, d_1 > \cdots > d_n.$$

Theorem 1. *The set $\mathscr{G}$ is the reduced Gröbner basis of the toric ideal I_A with respect to the term orden $>$. Moreover, $\mathscr{G}$ is the reduced Gröbner basis with respect to the order $<_{\boldsymbol{c}}$ induced by the cost vector* $\mathbf{c}$.

Proof. The proof follows similar steps and notation that [13, Thm. 4]. First of all, the set $\mathscr{G}$ is a subset of I_A, because all the binomials $\boldsymbol{z}^{\alpha} - \boldsymbol{z}^{\beta}$ in $\mathscr{G}$ verify $A\alpha = A\beta$.

The initial term of every binomial in $\mathscr{G}$ with respect to $>$ is the underlined term. It is enough to show that for every binomial $\boldsymbol{z}^{\alpha} - \boldsymbol{z}^{\beta} \in I_A$, with initial term $\boldsymbol{z}^{\alpha}$, there is some $g \in \mathscr{G}$ whose initial term divides $\boldsymbol{z}^{\alpha}$. By definition of toric ideal, $\boldsymbol{z}^{\alpha} - \boldsymbol{z}^{\beta} \in I_A$ if and only if $\alpha - \beta \in K = \{\boldsymbol{y} \in \mathbb{Z}^s \ : \ A\boldsymbol{y} = \mathbf{0}\}, s = n + 2N + 1$. We denote an element $\boldsymbol{y}$ in K by $\boldsymbol{y} = (y_x, y_d, y_t, y_b)$ to indicate the correspondence between components of $\boldsymbol{y}$ and the columns of A. In addition, we denote the components of y_x by $(X_{11}, \dots, X_{nk_n})$, and similarly for the others. We classify the elements in K in the following manner:

1. Let $K_1 = \{\boldsymbol{y} \in K \,:\, y_x = 0\}$. Now $\boldsymbol{y} \in K_1$ if and only if $(y_d, y_t, y_b) \in \mathbb{Z}^{s_1}, s_1 = n+N+1$ belongs to the lattice $S' = \{\boldsymbol{w} \in \mathbb{Z}^{s_1} \,:\, A'\boldsymbol{w} = 0\}$ where

$$A' = \begin{pmatrix} -I_n & & \\ & I_N & \\ & & 1 \end{pmatrix}.$$

   But $S' = 0$ since A' is a non singular matrix. Therefore $K_1 = 0$. This implies that there are no binomials of the form $\boldsymbol{z}^{\alpha} - \boldsymbol{z}^{\beta}$ that do not contain the variables x_{ij}.

2. Let $K_2 = \{\boldsymbol{y} \in K \,:\, y_d = 0\}$. Again $\boldsymbol{y} \in K_2$ if and only if $(y_x, y_t, y_b) \in \mathbb{Z}^{s_2}, s_2 = 2N+1$ belongs to the lattice $S'' = \{\boldsymbol{w} \in \mathbb{Z}^{s_2} \,:\, A''\boldsymbol{w} = 0\}$, where

$$A'' = \begin{pmatrix} D & 0 & 0 \\ I_N & I_N & 0 \\ \mathbf{c} & 0 & 1 \end{pmatrix}.$$

   Let X_{iq} be the left most nonzero component of y_x. We may assume that $X_{iq} > 0$ since S is the set of integer points in a vector space which implies that it contains the negative of every element in it. The i-th row of matrix D in A'' implies that there exists some $p > q$ such that $X_{ip} < 0$. Therefore,

$$x_{iq} \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^+} \text{ and } x_{ip} \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^-}.$$

   Consider now the rows given by the block $\begin{pmatrix} I_N & I_N & 0 \end{pmatrix}$ in A''. These rows imply that $T_{iq} = -X_{iq} < 0$ and $T_{ip} = -X_{ip} > 0$. Therefore,

$$x_{iq}t_{ip} \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^+} \text{ and } x_{ip}t_{iq} \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^-}.$$

   The initial term of $\boldsymbol{z}^{\boldsymbol{y}^+} - \boldsymbol{z}^{\boldsymbol{y}^-}$ with respect to $>$ is $\boldsymbol{z}^{\boldsymbol{y}^+}$ since x_{iq} divides $\boldsymbol{z}^{\boldsymbol{y}^+}$ and x_{iq} is the greatest variable that appears in this binomial. But this implies that the initial term of $x_{iq}t_{ip} - x_{ip}t_{iq}b^{c_{iq}-c_{ip}} \in \mathscr{G}$ divides the initial term of $\boldsymbol{z}^{\boldsymbol{y}^+} - \boldsymbol{z}^{\boldsymbol{y}^-}$. Therefore, the initial term of all binomials associated with K_2 is divisible by the initial term of an element in $\mathscr{G}$.

3. Consider now a general element in $S = \{\boldsymbol{y} \in \mathbb{Z}^s \,:\, A\boldsymbol{y} = 0\}, s = n+2N+1$, with no variables restricted to be zero. By the previous cases we may assume $y_x \neq 0, y_d \neq 0$. Let D_i be the first nonzero component of y_d. As before, we may assume that $D_i > 0$. Therefore,

$$d_i \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^+}.$$

Then there exists $X_{ik} > 0$, so

$$x_{ik}d_i \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^+}.$$

Similarly, there exists $T_{ik} < 0$ and

$$t_{ik} \text{ divides } \boldsymbol{z}^{\boldsymbol{y}^-},$$

so the initial term of $\boldsymbol{z}^{\boldsymbol{y}^+} - \boldsymbol{z}^{\boldsymbol{y}^-}$ is $\boldsymbol{z}^{\boldsymbol{y}^+}$ because $d_i > t_{ik}$. This initial term is divisible by $x_{ik}d_i$, which is the initial term of $x_{ik}d_i - t_{ik}b^{c_{ik}}$. Therefore

$$\text{in}_<(I_A) = \langle \text{in}_<(\mathscr{G}) \rangle,$$

which proves that $\mathscr{G}$ is a Gröbner basis of I_A with respecto to the term order $<$. Clearly, it is reduced.

Moreover, with respect to the term order $<_{\boldsymbol{c}}$,

$$\text{in}_{<_{\boldsymbol{c}}}(x_{ik}d_i - t_{ik}b^{c_{ik}}) = x_{ik}d_i,$$

because the weight of the first monomial is equal to $c_{ik} > 0$, and the weight of the second monomial is equal to zero. Similarly,

$$\text{in}_{<_{\boldsymbol{c}}}(x_{iq}t_{ip} - x_{ip}t_{iq}b^{c_{iq}-c_{ip}}) = x_{iq}t_{ip}, 1 \leq q < p < k_i,$$

because the weight of the first monomial is $c_{iq} \geq c_{ip}$, which is the weight of the second monomial. If $c_{iq} = c_{ip}$, the tie is broken with the lexicographical order $x_{iq} > x_{ip}$.

□ □

The above theorem gives a *reduced* Gröbner basis with respect to the term order induced by the objective function of (LR). On the contrary, [13, Thm. 4] only provides a Gröbner basis with respect to a lexicographical order, and not with respect to the term order needed for the computation of the test set. Therefore, in order for that Gröbner basis to be applied to solve their problem one more computational step is required whereas our construction gives directly the answer with its consequent saving.

6 Computational results

The previous construction of the test set is used in our computational experiments. In order to gain some insights of its efficiency, if a program like `4ti2`

([4ti2 team(2008)]) were used to do the computation of the test set, a simple configuration of 10 subsystems with 3 components would take more than 60 minutes. Therefore, it is very important to apply the result in Theorem 1 to be able to construct the test set.

Our algorithm has been coded in MATLAB and run on a AMD Opteron 252 (2.6 GHz) with 5 GB RAM. For Table 1, all the data in the test problems are randomly generated from uniform distributions, with $l_{ij} = 0, u_{ij} = 4$ and $r_{ij} \in [0.99, 0.998]$, as in [11]. The linear cost function $\sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij}$ has values $c_{ij} \in [10, 20]$.

The number n is the number of subsystems, and k is the number of different components in each subsystem.

Table 1: $R_0 = 0.90, r_{ij} \in [0.99, 0.998]$

n	k	Nodes	Iter. R-S	Avg. CPU time (s)
10	2	0	696	0.0
10	3	0	5797	0.0
10	5	0	26427	0.0
15	2	0	15184	0.0
15	3	0.4	85103	0.1
20	2	7041	294747	276.0

The average CPU time, and the average number of generated nodes during the algorithm has been obtained by running the program for 10 instances.

The column "Iter. R-S" contains the number of iterations according to the results given in [11, Table 1]. Comparing our results with [11], not only the CPU time is improved, but also the effort measured by the number of processed nodes by the walk back procedure is less than the number of iterations in [11]. We also point out that the iterations in [11] compute two Lagrangian discrete relaxations and their corresponding solutions for the best value, each time. After that, to discard remaining boxes in their branch and bound tree, reliability of that solution is needed. In our method in each iteration we only compute a node by adding a vector, and then compute its reliability.

In order to better illustrate the results, new tests have been done with the additional hypothesis that a greater reliability in a component implies a greater cost. Note that if there is no correlation between cost and reliability of a component (as in [11]), then it is likely that certain components are not going to be used, because only more reliable components are going to be chosen regardless of their cost. Hence the dimensionality of the problem is artificially reduced. The results of this more realistic case appear in Table 2. It is clear the increasing computational effort

Table 2: $R_0 = 0.90, r_{ij} \in [0.99, 0.998]$ (**ordered**)

n	k	Nodes	Avg. CPU time (s)
10	2	0	0.0
10	3	0	0.0
10	5	0	0.0
15	2	45	0.2
15	3	661	4.4
15	4	28023	10571
17	2	12578	1355

showed by rows $n = 15, k = 3$ and $n = 15, k = 4$. From the above, we conclude that the computational experiments for this model should be done with this additional hypothesis of correlation between cost and reliability of each component.

We also note that the algorithm is very sensitive to changes in the value of the reliability parameters r_{ij}. For example, for less reliable components, $r_{ij} \in [0.980, 0.990]$ and the same value, $R_0 = 0.90$, for the overall reliability, we have obtained the results in Table 3. The reader may observe that the system sizes that

Table 3: $R_0 = 0.90, \mathbf{r}_{ij} \in [\mathbf{0.98}, \mathbf{0.99}]$ (**ordered**)

n	k	Nodes	Avg. CPU time (s)
6	4	14	0
6	5	39	0.1
7	4	1186	7.4
7	5	5662	140
8	4	46709	7010

can be solved are smaller. However, we have to point out that an exact solution has been found in all the examples. An interesting remark is that the elapsed time is significantly reduced if the algorithm is stopped with the first best point found in the walk back procedure. Obviously, this approach does not guarantee optimality but it gives very accurate approximations. From this observation, we think that a promising open field is the combination of this technique with heuristic methods to get a good approximation of the optimal solution.

7 Conclusion

We have presented in this paper an exact method for solving a nonlinear integer programming problem arising from the design of series-parallel reliability systems. The method is based on the construction of a test set of an integer linear problem through the theory of Gröbner bases. We provide an explicit formula of the test set, avoiding the high cost of this computation. Computational tests show that this approach improves existing methods already applied for this problem.

This paper deepens the challenge given in [13] to yield efficient algorithms for problems in integer problems based on attractive bases.

References

[4ti2 team(2008)] 4ti2 team, 2008. 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces. Available at www.4ti2.de.

[1] W.W. Adams, P. Loustaunau, An introduction to Gröbner bases, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, Providence, RI, 1994.

[2] F. Ahmadizar, H. Soltanpanah, Reliability optimization of a series system whith multiple-choice and budget constraints using an efficient ant colony approach, Expert systems with Applications, 38, (2011), 3640–3646.

[3] D. Bertsimas, R. Weismantel, Optimization over integers, Dynamic ideas, 2005.

[4] Castro, F., Gago, J., Hartillo, I., Puerto, J., Ucha, J.M., 2011. An algebraic approach to integer portfolio problems. European Journal of Operational Research, 210, 647–659.

[5] M.S. Chern, On the computational-complexity of reliability redundancy allocation in a series system, Oper. Res. Lett., 11, (1992), 309–315.

[6] D.W. Coit, A.E. Smith, Reliability optimization of series-parallel systems using a genetic algorithm, IEEE Trans. Reliab., 45, (1996) 254–260.

[7] P. Conti, C. Traverso, Buchberger algorithm and integer programming. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Lect. Notes Comput. Sci., 539, (1991), 130–139., second ed.

[8] D.A. Cox, J. Little, D. O'Shea, Using Algebraic Geometry, Graduate Texts in Mathematics, vol. 185, Springer, New York, 2005.

[9] M. Djerdjour, K. Rekab, A branch and bound algorithm for designing reliable systems at a minimum cost, Appl. Math. Comput, 118,(2001) 247–259.

[10] M. Ouzineb, M. Nourelfath, M. Gendreau, Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems, Reliab. Eng. Syst. Saf, 93, (2008), 1257–1272.

[11] N. Ruan, XL. Sun, An exact algorithm for cost minimization in series reliability systems with multiple component choices, Appl. Math. Comput, 181, (2006) 732–741.

[12] B. Sturmfels, Gröbner Bases and Convex Polytopes, University Lecture Series, vol. 8, American Mathematical Society, Providence, Rhode Island, 1996.

[13] S.R. Tayur, R.R. Thomas, N.R. Natraj, An algebraic geometry algorithm for scheduling in presence of setups and correlated demands, Math. Program., 69, (1995) 369–401.

[14] A. Yalaoui, E. Chatelet, C.B. Chu, A new dynamic programming method for reliability & redundancy allocation in a parallel-series system, IEEE Trans. Reliab., 54, (2005), 254–261.