# AN IMPROVED TEST SET APPROACH TO NONLINEAR INTEGER PROBLEMS WITH APPLICATIONS TO ENGINEERING DESIGN

J. GAGO-VARGAS, M.I. HARTILLO-HERMOSO, J. PUERTO-ALBANDOZ, AND J.M. UCHA-ENRÍQUEZ

Abstract. Many problems in engineering design involve the use of nonlinearities and some integer variables. Methods based on test sets have been proposed to solve some particular problems with integer variables, but they have not been frequently applied because of computation costs. The walk-back procedure based on a test set gives an exact method to obtain an optimal point of an integer programming problem with linear and nonlinear constraints, but the calculation of this test set and the identification of an optimal solution using the test set directions are usually computationally intensive.

In problems for which obtaining the test set is reasonably fast, we show how the effectiveness can still be substantially improved. This methodology is presented in its full generality and illustrated on two specific problems: (1) minimizing cost in the problem of scheduling jobs on parallel machines given restrictions on demands and capacity, and (2) minimizing cost in the series parallel redundancy allocation problem, given a target reliability. Our computational results are promising and suggest the applicability of this approach to deal with other problems with similar characteristics or to combine it with mainstream solvers to certify optimality.

Non-linear Integer Programming and test set and Gröbner basis and chance constrained programming

## 1. Introduction

The ever-increasing demand on operations researchers to lower production costs or to increase benefits has prompted the specialized community to look for rigorous methods of decision making, such as optimization methods, to design and process both economically and efficiently most engineering systems. Optimization techniques, having reached a degree of maturity over the last years, are being used in a wide spectrum of engineering applications, including reliability of systems, finance, scheduling, as well as various contributions in aerospace, automotive, chemical, electrical or manufacturing industries. For some of those applications Operations Research has developed efficient algorithms, especially when the problems can be modeled either as continuous (linear or nonlinear) programs or discrete linear programs. However, in many occasions the resulting models contain both nonlinearities and discrete variables which make their resolution challenging.

The purpose of this paper is to improve a general technique to handle nonlinear integer problems and to show some applications to engineering optimization problems in a simple manner. This technique combines three essential elements: 1) it borrows some tools from Computational Algebra which are applied to generate

*Date*: Received: date / Accepted: date.

*test sets* for linear integer programs, 2) it uses relaxed problems to obtain seeds for a dual search procedure, and 3) it defines an order on the search tree of feasible solutions that uses the classical idea of a penalty strategy that should take into account both the objective function value and the distance to the feasible region. This third ingredient is specifically the new one that makes the general dual search (or *walk-back procedure*) much more efficient in the experiments.

The general theory of penalty functions usually considers the transformation of a constrained problem into unconstrained one(s) and afterward, starting from some convenient seed, uses suitable directions in order to reach an optimum. Our approach is a natural generalization in this context: once the optimum for a relaxed integer linear problem of the original problem is reached, the process starts from this optimum and chooses the most promising nodes balancing the cost and the distance to the whole feasible region. The main advance of our algorithm with respect to the one described in [30] is the insertion in the pending nodes list by the ascending penalized cost. We will see in Section 4 the effectiveness of this change in some examples.

The method is described in its full generality and then it is tested in two engineering design problems taken from the literature, namely, the minimization of the cost of scheduling jobs on parallel machines given restrictions on demands and capacity and in the minimization of the cost in the series parallel redundancy allocation problem given a target reliability. For the first problem, with stochastic restrictions, it is remarkable that our method can manage the examples tested whereas other mainstream nonlinear solvers cannot deal with the nonlinear formulation. The results obtained in the second problem are competitive with the nonlinear solvers, with a remarkable performance in the time needed to reach the optimum.

The paper is organized as follows. In Section 2 the basics about the walk-back procedure using a test set associated with an integer linear program are presented. Section 3 contains the main technique followed to improve the pure walk-back procedure. In this section it is also introduced the penalized cost function that can be employed to order, in an efficient way, the feasible points of the relaxed integer linear problem. Section 4 is devoted to the examples where the method has been tested. In all the examples tables with execution data (CPU time, number of processed nodes) and comparison with previous works ([30], [25], [17]) and other nonlinear integer programming solvers are included. Additionally, we show how to exploit a combined strategy with other solvers to speed up certification of optimality. Finally, Section 5 contains the conclusions.

## 2. Preliminaries

This section contains a brief summary of the concepts and algorithms used to solve integer linear programming problems from an algebraic point of view. In addition it is recalled the walk back procedure for nonlinear integer programming problems based on test sets. To this end, we have followed [28] and [30].

### 2.1. **Test sets.** Consider an integer linear programming problem:

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

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 integer linear programming problem with right-hand-side fixed to $\boldsymbol{b}$. (LP) denotes the set of all the integer linear programming problems obtained by varying the right-hand-side vector $\boldsymbol{b}$, fixed $A$ and the cost function $\boldsymbol{c}$. Let $\pi$ be the map defined by $\pi(\boldsymbol{x}) = A\boldsymbol{x}$. Given a vector $\boldsymbol{b} \in \mathbb{Z}^d$, the set $\pi^{-1}(\boldsymbol{b}) = \{\boldsymbol{u} \in \mathbb{Z}^N_+ \ : \ \pi(\boldsymbol{u}) = \boldsymbol{b}\}$ is the fiber of (LP) over $\boldsymbol{b}$.

Fixed a term order $<_{\boldsymbol{c}}$, associated with the cost, defined as usual in [28], there exists a unique optimum $\boldsymbol{\beta}$ for (LP($\boldsymbol{b}$)). A set $G_{<_{\boldsymbol{c}}} \subset \mathbb{Z}^N$ is a *test set* for the family of integer linear problems (LP) with respect to the matrix $A$ and the order $<_{\boldsymbol{c}}$ if

- for each nonoptimal point $\boldsymbol{\alpha}$ of (LP($\boldsymbol{b}$)), there exists $\boldsymbol{g} \in G_{<_{\boldsymbol{c}}}$ such that $\boldsymbol{\alpha} - \boldsymbol{g}$ is a feasible solution for (LP($\boldsymbol{b}$)) and $\boldsymbol{\alpha} - \boldsymbol{g} <_{\boldsymbol{c}} \boldsymbol{\alpha}$,
- for the optimal point $\boldsymbol{\beta}$, $\boldsymbol{\beta} - \boldsymbol{g}$ is not a feasible point for any $\boldsymbol{g} \in G_{<_{\boldsymbol{c}}}$.

In this way, a test set for (LP) connects the fiber as a directed graph with a unique sink. It gives an obvious algorithm to solve an integer program, starting from 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.
- There does not exist such an element in the set, so that the point is the optimum of the fiber.

2.2. **Toric ideal.** The toric ideal associated with $A$, denoted as $I_A$ is defined as

$$I_A = \langle \boldsymbol{x}^{\boldsymbol{\alpha}} - \boldsymbol{x}^{\boldsymbol{\beta}} \ : \ A\boldsymbol{\alpha} = A\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}^N_+ \rangle \subset \mathbb{Q}[x_1, \ldots, x_N].$$

Given a term order $<_{\boldsymbol{c}}$ compatible with the cost function $c(\boldsymbol{x})$, a reduced Gröbner basis $\mathcal{G}_{<_{\boldsymbol{c}}}$ of $I_A$ yields a test set $G_{<_{\boldsymbol{c}}}$ for (LP) (cf. [28]). If the reduced Gröbner basis is formed by binomials

$$\mathcal{G}_{<_{\boldsymbol{c}}} = \{\boldsymbol{x}^{\boldsymbol{\alpha}_i} - \boldsymbol{x}^{\boldsymbol{\beta}_i}, i = 1, 2, \ldots, r\}, \text{ with } \mathrm{in}_{<_{\boldsymbol{c}}}(\boldsymbol{x}^{\boldsymbol{\alpha}_i} - \boldsymbol{x}^{\boldsymbol{\beta}_i}) = \boldsymbol{x}^{\boldsymbol{\alpha}_i},$$

where $\mathrm{in}_{<_{\boldsymbol{c}}}$ stands for the greatest monomial with respect to $<_{\boldsymbol{c}}$, then the test set is expressed as

$$G_{<_{\boldsymbol{c}}} = \{\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i, i = 1, 2, \ldots, r\}.$$

2.3. **Walk back procedure.** The walk back procedure is an algorithm which computes the optimum for a nonlinear integer programming problem under some conditions. Our problem is of the form:

$$\begin{aligned} \text{(P)} \quad \min \quad & c(\boldsymbol{x}) = \boldsymbol{c}^t \cdot \boldsymbol{x} \\ & \boldsymbol{x} \in \mathcal{A} \subset \mathbb{Z}^n_+, \\ & \boldsymbol{x} \in \mathcal{B} \subset \mathbb{Z}^n_+, \end{aligned}$$

where

- the constraints $\boldsymbol{x} \in \mathcal{A}$ are linear, and a test set can be obtained in practice with respect to the linear cost function and these restrictions;
- the constraints $\boldsymbol{x} \in \mathcal{B}$ can be linear and nonlinear.

We can assume without loss of generality that $c(\boldsymbol{x})$ is a linear cost function because if it were nonlinear we would transform the problem introducing the new constraint $c(\boldsymbol{x}) \leq z$, that would be included in the set $\boldsymbol{x} \in \mathcal{B}$, and the objective function $\min z$. The problem would now fit within the original hypothesis, but the test set will change because the cost function that defines the term order is different.

Let (LIP) denotes the relaxed linear integer problem, namely:

$$\text{(LIP)} \quad \min \quad c(\boldsymbol{x}) \\ \boldsymbol{x} \in \mathcal{A} \subset \mathbb{Z}_+^n.$$

Let $G$ be the test set associated with the relaxed linear problem (LIP). Starting from some solution of (LIP) and adding vectors from $G$, every feasible point of the relaxed problem (LIP) can be reached. So, in particular, every feasible point of problem (P) can be reached as well. But we do not need to complete an exhaustive enumeration of the feasible points of (P): if $\boldsymbol{p}'$ is a feasible point for (P), with cost $c(\boldsymbol{p}')$, all the paths starting from points whose cost is greater than $c(\boldsymbol{p}')$ can be pruned. Remember that subtracting the elements of the test set always reduces the cost, so adding them (or subtracting the elements of the *reverse* test set) provides points that do not improve the value $c(\boldsymbol{p}')$. When a path is pruned, all its points are discarded. Points with negative components are deleted too. Along this process new points are generated, called pending nodes, and the paths starting from them must be analyzed. At the end all the possibilities will have been processed, and the final result will be an optimum of (P), or a certificate that problem (P) has an empty feasible region if no feasible point could be reached. This is the *walk-back procedure*, described in [30]. The above argument can be summarized in the following result:

**Theorem 2.1.** *The walk-back procedure solves the problem* (P)*, that is, it obtains an optimum or shows that the problem has no feasible solution.*

To the best of our knowledge, there have been few practical examples where the walk-back procedure has been successfully applied, as in [30], [17], [9]. This lack of applicability may be due to its two main drawbacks: the computation of the test set and the time required for visiting the points $\boldsymbol{x} \in \mathcal{A}$ to eventually obtain an optimum for (P).

- The computation of the test set has, in general, a high cost. We have used the toric ideal approach [28], which is implemented in a very efficient way in the software 4ti2 [1]. In some cases, there is no need of such computation, because it is possible to give a closed form of a Gröbner basis of the problem, as it is made for example in [17].
- The points are ordered taking into account their cost. As soon as a feasible point for (P) is reached, new points are found quickly, and one of them is often an optimum. However, the processing time to reach such feasible points is usually long and the list of pending nodes is huge.

The question is whether it is possible to order the points that have to be visited in an alternative way, so that new feasible better points are obtained earlier. This approach is explained in Section 3.

## 3. Combining test sets and a penalized strategy

3.1. **Penalty functions.** Traditionally, methods using penalty functions transform a constrained problem into a single unconstrained problem or into a sequence of unconstrained problems. Unconstrained problems are easier to manage in general. The general idea is to place the constraints into the objective function with a penalty parameter that *penalizes* properly any violation of these constraints.

Given a problem (P),

$$\begin{array}{lll} \text{(P)} & \min & c(\boldsymbol{x}) \\ & & \boldsymbol{x} \in \mathcal{A} \subset \mathbb{Z}_+^n, \\ & & \boldsymbol{x} \in \mathcal{B} \subset \mathbb{Z}_+^n, \end{array}$$

a *penalty function* $p(\boldsymbol{x})$ is a function such that $p(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathcal{A} \cap \mathcal{B}$ and $p(\boldsymbol{x}) \geq \gamma > 0$ if $\boldsymbol{x} \in \mathcal{A} \setminus \mathcal{B}$. Moreover, the statement $\gamma > 0$ comes from the discrete nature of (P).

For a problem (P) the *penalty problem* ($\mathrm{P}_{\mathrm{pen}}(\mu)$) is defined as

$$\min_{\boldsymbol{x} \in \mathcal{A}} T(\boldsymbol{x}, \mu) = c(\boldsymbol{x}) + \mu p(\boldsymbol{x}), \text{ for a certain } \mu > 0.$$

Since $T(\boldsymbol{x}, \mu) = c(\boldsymbol{x})$ if $\boldsymbol{x} \in \mathcal{A} \cap \mathcal{B}$, it is clear that

$$\mathrm{opt(P)} \geq \mathrm{opt}(\mathrm{P}_{\mathrm{pen}}(\mu)).$$

In the problems that we deal with in this work the region $\mathcal{B}$ is possibly an enormous, but finite, subset of $\mathbb{Z}^n$. It can be proved that, in this situation the problem is equivalent to a penalized one.

**Theorem 3.1** (cf. [20] Th. 2.11)**.** *Suppose that $\mathcal{A} \setminus \mathcal{B} \neq \emptyset$. Let $c_0$ be a lower bound of $c(\boldsymbol{x})$ in $\mathcal{A}$ and $\rho > 0$ be a lower bound of $\min_{\boldsymbol{x} \in \mathcal{A} \setminus \mathcal{B}} p(\boldsymbol{x})$. Then, for any $\mu > \mu_0$ with*

$$\mu_0 = \frac{\mathrm{opt(P)} - c_0}{\rho}$$

*we have*

$$\mathrm{opt(P)} = \mathrm{opt}(\mathrm{P}_{\mathrm{pen}}(\mu)).$$

3.2. **Ordering nodes by penalized cost.** Following the notation of [30] we describe in this subsection our algorithm. The walk-back procedure provides a list of feasible points, noted $\mathcal{P}$, for the relaxed linear problem (LIP) and a guided search procedure that allows us to get an exact optimum for the original problem (P). However, the way in which the points are scanned is fundamental to create an algorithm that performs the computations in a reasonable time. A first approach is to order the pending nodes list according to their cost ([30], [17]). A second approach, presented here, consists of ordering the pending nodes by a penalized cost function of the form

$$T(\boldsymbol{x}, \mu) = c(\boldsymbol{x}) + \mu p(\boldsymbol{x}),$$

where the cost of the point is added to a value associated with the distance to the region defined by the constraints of $\mathcal{B}$ in problem (P). Instead of ordering the points to be visited by the original cost function $c$, the penalized cost function $T(\boldsymbol{x}, \mu)$ is used, expecting to reach faster some points that are feasible for the whole problem (P). This is the main advance of our algorithm compared to the one described in [30]: the insertion in the pending nodes list ordered by the new penalized cost function.

It is worthwhile to emphasize that this procedure is *exact* and *certifies* the optimality as proved in Theorem 2.1, because *it is only a different way of visiting the points required to solve the problem.* It starts from the optimum $\boldsymbol{\beta}$ of (LIP) (that we assume infeasible for (P) since otherwise we are done), and produces recursively new points $\boldsymbol{x}$ that are in two possible situations:

(1) $\boldsymbol{x}$ is feasible for (P), so we include it in the set $Y$ of feasible points for (P) obtained so far. It is not necessary to follow paths from $\boldsymbol{x}$ since the cost would be worse;
(2) $\boldsymbol{x}$ is not feasible for (P) and we do not have points in $Y$ with better cost, then $\boldsymbol{x}$ is inserted according to the function $T(\boldsymbol{x}, \mu)$ in the set $\mathcal{P}$ of nodes to examine.

## 4. Applications

4.1. **Correlated setup.** The walk back procedure was introduced in [30] and applied in a problem of assignment of jobs to machines, with given production and correlated setup costs, capacity constraints and probability to reach a given demand. We deal with the same problem by adding the improvement of the penalized cost and a set of new cuts that does not alter the optimum.

The notation for the model is the following:

- $n$ number of job types, indexed by $i$,
- $m$ number of machines, indexed by $j$,
- $(D_1, \ldots, D_n)$ random vector of demands,
- $N$ size of the sample set used to estimate the probability,
- $C_j$ capacity (time) for each machine,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ means vector of the probability distribution of demand,
- $S_{ij}$ setup time for job type $i$ on machine $j$,
- $K_{ij}$ setup cost for job type $i$ on machine $j$,
- $M_i$ lot splitting,
- $L'_{ij}$ the cost of producing a unit of product type $i$ on machine $j$,
- $L_{ij} = (\hat{D}_i/M_i)L'_{ij}$,
- $p_{ij}$ processing time for a unit of job type $i$ on machine $j$,
- $\gamma$ probability of no shortfall.
- $z_{ij}$ equals 1 if job type $i$ is scheduled on machine $j$, 0 otherwise,
- $y_{ij}$ multiples of $1/M_i$ of demand of product $i$ are scheduled on machine $j$.

The model presented in [30] is:

$$\text{(SP) minimize } \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

subject to

$$(1) \quad \sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n,$$

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

$$(3) \quad g_j(\boldsymbol{z}, \boldsymbol{y}) = \sum_{i=1}^{n} p_{ij} \left( \hat{D}_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m,$$

$$(4) \quad g_0(\boldsymbol{z}, \boldsymbol{y}) = \text{Prob} \left\{ \sum_{i=1}^{n} p_{ij} \left( D_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m \right\} \geq \gamma,$$

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

The condition (4) will be computed in a sample dataset of size $N$. As usual we note (LSP) the linear relaxed problem defined by constraints (1), (2) and (5).

The model (SP) admits a family of optimality cuts:

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \tag{6}$$

It is clear that if the job $i$ is assigned to the machine $j$, then something is produced in it. These constraints reduce the feasible region, but do not alter the optimum of problem (SP). We call problem (ISP) the resulting model obtained from (SP) after adding the constraints (6).

The test set is associated with the linear constraints (1), (2), (5) and (6), which defines a relaxed problem (LISP). The equations (3) and (4) are checked for every point and test feasibility for Problem (ISP).

Similarly to [30], the linear problem with restrictions (1), (2), (5) and (6) can be expressed with equalities by adding slack variables:

$$\text{(LISP} - \text{II) minimize } \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

subject to

$$\sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n, \tag{7}$$

$$y_{ij} + a_{ij} - M_i z_{ij} = 0, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{8}$$

$$z_{ij} + b_{ij} = 1, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{9}$$

$$-y_{ij} + z_{ij} + c_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{10}$$

$$y_{ij}, a_{ij}, b_{ij}, c_{ij} \in \mathbb{Z}_+, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m. \tag{11}$$

**Theorem 4.1.** *The set of binomials*

$$\begin{aligned}
y_{ip} a_{iq} c_{ip} - y_{iq} a_{ip} c_{iq}, &\qquad i = 1, 2, \ldots, n, 1 \leq p < q \leq m, \\
b_{ip} c_{ip} - z_{ip} a_{ip}^{M_i}, &\qquad i = 1, 2, \ldots, n, p = 1, 2, \ldots, m, \\
b_{ip} y_{iq} c_{iq} - z_{ip} y_{ip} a_{ip}^{M_i - 1} a_{iq}, &\qquad i = 1, 2, \ldots, n, 1 \leq p \neq q \leq m, \\
b_{ip} y_{iq} z_{iq} a_{iq}^{M_i - 1} - z_{ip} y_{ip} b_{iq} a_{ip}^{M_i - 1}, &\qquad 1 \leq p < q \leq m,
\end{aligned}$$

*is the reduced Gröbner basis of the toric ideal corresponding to* (LISP-II) *with respect to the lexicographical order $b > y > z > a > c$. The letter $y$ denotes the set of variables $(y_{11}, \ldots, y_{mn})$ and similarly $a, z, b$ and $c$ denote the other sets of variables. Within a block, the variables are sorted lexicographically according to index.*

The proof is written in Appendix A.

However, this basis is not the one needed to build the test set with respect to the cost function. We have used 4ti2 to get the basis in a straightforward way, and the computation cost has been always under one second for all the considered instances, which are of the same size of those proposed in [30].

4.1.1. *New penalized cost function.* Following the notation of Section 3, let $\boldsymbol{x}$ be the set of variables $y_{ij}, z_{ij}$. Multiplying by a suitable constant, we can assume that all the coefficients in $g_j(\boldsymbol{x})$ for $j = 1, 2, \ldots, n$ are integers. We consider the functions

$$G_0(\boldsymbol{x}) = \gamma - g_0(\boldsymbol{x}), \quad G_j(\boldsymbol{x}) = g_j(\boldsymbol{x}) - C_j.$$

Let $\mathcal{A}$ be the finite set in $\mathbb{Z}^{n+m}$ defined by the linear constraints (1), (2), (5) and (6). We call

$$\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^{n+m} \mid G_j(\boldsymbol{x}) \leq 0, j = 0, 1, \ldots, m\},$$

and define the penalty function

$$p(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

Theorem 3.1 gives a constant $\mu_0$ which frames a parameter region where the same objective value for the nonlinear problem and the penalized version is obtained.

Let $c(\boldsymbol{x}) = \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$ be the cost function and $\hat{\boldsymbol{x}}$ the optimal solution of the relaxed problem (LISP).

**Proposition 4.1.** *Let $c_1 := \sum_i \sum_j (K_{ij} + L_{ij})$, $N$ be the size of the sample set and $\rho_0 = \frac{1}{2N}$. Let* (ISP)$'$ *be the problem that results from* (ISP) *after replacing constraint (4) by $g_0(\boldsymbol{x}) \geq \gamma'$, with $\gamma' = \frac{\lceil N\gamma \rceil}{N} - \frac{1}{2N}$.*

*The objective values of problems* (ISP) *and* (ISP)$'$ *coincide. Moreover, for any $\mu \geq \mu_0 = \frac{c_1 - c(\hat{x})}{\rho_0}$, Problem* (ISP) *and the penalized version of* (ISP)$'$ *with objective function $c(\boldsymbol{x}) + \mu p(\boldsymbol{x})$ have the same optimal value.*

*Proof.* First of all, we observe that $g_0(\boldsymbol{x})$ is a stepwise function that can only assume the values $\frac{k}{N}$ for $k = 1, \ldots, N$. Therefore, the set of feasible points of (ISP) that satisfy $g_0(\boldsymbol{x}) \geq \frac{\lceil N\gamma \rceil}{N}$ for any $\gamma \in (0, 1)$ is the same that those that satisfy $g_0(\boldsymbol{x}) \geq \frac{\lceil N\gamma \rceil}{N} - \frac{1}{2N}$. This proves the first statement of the proposition.

From now on we only consider (ISP)$'$ and its penalized version. In order to apply Theorem 2, we need to prove that $\rho_0$ is a valid lower bound of $p(\boldsymbol{x})$ on $\mathcal{A} \setminus \mathcal{B}$. Indeed, we note that if $\boldsymbol{x} \notin \mathcal{B}$ then either $G_j(\boldsymbol{x}) > 1$ for some $j = 1, \ldots, m$ or $g_0(\boldsymbol{x}) < \frac{\lceil N\gamma \rceil}{N} - \frac{1}{2N}$. In the former case, we have $p(\boldsymbol{x}) \geq \sum_{j=1}^{m} G_j(\boldsymbol{x}) \geq 1 > \rho_0$. In the latter case, due to $\frac{\lceil N\gamma \rceil - 1}{N} < \frac{\lceil N\gamma \rceil}{N} - \frac{1}{2N} < \frac{\lceil N\gamma \rceil}{N}$ and taking into account that $g_0(\boldsymbol{x})$ only assumes values $\frac{k}{N}$, then the constraint $g_0(\boldsymbol{x}) < \frac{\lceil N\gamma \rceil}{N} - \frac{1}{2N}$ is equivalent to $g_0(\boldsymbol{x}) \leq \frac{\lceil N\gamma \rceil - 1}{N}$. So $\gamma' - g_0(\boldsymbol{x}) \geq \frac{1}{2N}$ and hence $p(\boldsymbol{x}) \geq G_0(\boldsymbol{x}) \geq \gamma' - g_0(\boldsymbol{x}) \geq \frac{1}{2N}$.

Theorem 2 provides a constant $\mu_0$ which depends on the unknown optimal value of (ISP)$'$. To avoid this inconvenience, we shall replace $\mu_0$ by something greater. We substitute the unknown value of (ISP)$'$ by $c_1$, a valid upper bound of that value. Hence, we can apply Theorem 3.1 to (ISP)$'$ and its penalized version to conclude the second statement of the proposition. □

□

In spite of the above result, our algorithm does not solve the penalized version. Rather than that we use the order induced by its values to guide the search strategy in our branching tree. However, the above estimation gives a good estimate of $\mu$ in our penalty term because a big value of $\mu$ yields a poor performance of our algorithm.

It is clear that $c_0 = c(\hat{\boldsymbol{x}})$ is a lower bound of $\min_{\boldsymbol{x} \in \mathcal{A}} c(\boldsymbol{x})$. In our computational results, after some experimentation, we have used a different estimate of the value of $\min_{\boldsymbol{x} \in \mathcal{A} \setminus \mathcal{B}} p(\boldsymbol{x})$ than the one given in Proposition 4.1 because of its better performance.

Indeed, the values of the functions $G_j(\boldsymbol{x})$ for $j = 1, 2, \ldots, m$ are greater than one, while the values of $G_0(\boldsymbol{x})$ are less than 1. We let $\bar{\rho} = G_0(\hat{\boldsymbol{x}})$, and take

$$\bar{\mu}_0 = \frac{c_1 - c_0}{\bar{\rho}}, \quad \alpha = \lfloor \log(\frac{c_0}{\bar{\mu}_0}) \rfloor, \text{ and } \mu = \alpha \bar{\mu}_0.$$

This quotient is intended to avoid that the coefficient $\bar{\mu}_0$ takes on values very large with respect to $c_0$. The penalized cost is then $T(\boldsymbol{x}, \mu) = c(\boldsymbol{x}) + \mu p(\boldsymbol{x})$.

We have considered other penalized cost functions, as the adaptive function described in [11], or the oracle function of [27], but the computational results have been slightly worse than the experiments with the previously defined function $T(\boldsymbol{x}, \mu)$.

4.1.2. *Computational results.* In order to evaluate the performance of the new ordering in the list $\mathcal{P}$ of pending nodes, several families of instances have been considered and compared with the previous walk-back procedure. Both algorithms have been coded in MATLAB and run on a Intel Xeon X5660 (2.8 GHz) with 32 GB RAM. The data have been randomly generated with the conditions described in [30, Example 4], that is, $n = 7$ jobs and $m = 4$ machines, with capacities $(20, 20, 40, 40)$ (there is a misprint in the original article), mean demands equal to $\boldsymbol{\mu} = (20, 16, 12, 8, 8, 4, 4)$, lots $M = 2$ and covariance matrix

$$V = \begin{pmatrix} 36 & 0 & -10.8 & 0 & 0 & 0 & 7.34 \\ 0 & 23 & 0 & 5.9 & 11.5 & 0 & 0 \\ -10.8 & 0 & 13 & 0 & 0 & -4.4 & -4.4 \\ 0 & 5.9 & 0 & 6 & 0 & 0 & 0 \\ 0 & 11.5 & 0 & 0 & 23 & 0 & 0 \\ 0 & 0 & -4.4 & 0 & 0 & 6 & 0 \\ 7.34 & 0 & -4.4 & 0 & 0 & 0 & 6 \end{pmatrix}.$$

All setup times $S_{ij}$ and production times $p_{ij}$ are equal to 1. The setup and production costs $(K_{ij}, L'_{ij})$ are provided in Table 1, similarly to [30, Fig. 3], but with the errors corrected (the values were swapped).

TABLE 1. Setup and production costs for $n = 7$ jobs, $m = 4$ machines

| | job 1 | job2 | job 3 | job 4 | job 5 | job 6 | job 7 |
|---|---|---|---|---|---|---|---|
| machine 1 | 5, 3 | 3, 4 | 3, 3 | 1, 2 | 1, 3 | 3, 3 | 5, 4 |
| machine 2 | 1, 4 | 1, 4 | 4, 2 | 5, 2 | 3, 4 | 1, 3 | 2, 2 |
| machine 3 | 4, 3 | 2, 3 | 5, 2 | 4, 2 | 4, 4 | 2, 2 | 4, 4 |
| machine 4 | 5, 3 | 4, 4 | 4, 2 | 4, 3 | 1, 3 | 5, 4 | 2, 2 |

Different sample sets have been considered, with 250 records, mean $\boldsymbol{\mu}$ and covariance matrix $V$. There are no differences among the different samples, we have chosen the more descriptive one. The results appear in Table 2 for different values of the probability $\gamma$. We compare the runs for the original walk-back procedure of [30], ordered only by the cost (column "Walk-back" in tables), and the runs with the previously defined penalized cost function (column "WB new ordering" for function $T(\boldsymbol{x}, \mu)$). The column "Total" groups the number of visited nodes and CPU time in seconds. The maximum number of points to be considered has been set to 120000, (a proxy for a CPU time approximately equivalent to 90 minutes of computation), and the label "Max" denotes that the run has reached this limit and it has been terminated. The column "Optimum" groups the number of visited nodes and CPU time to get an optimum. When the process has reached the maximum number of points, the value of the best point found is shown, but it is not possible to guarantee that it is an optimum. If the process has not even found a feasible point we write "NNP". The computation of the Gröbner basis is done

TABLE 2. 4 machines, 7 jobs, $M = 2$, improved model, 250 records, Time in seconds

| | Walk-back | | | | WB new ordering | | | |
|---|---|---|---|---|---|---|---|---|
| | Total | | Optimum | | Total | | Optimum | |
| $\gamma$ | Time | Nodes | Time | Nodes | Time | Nodes | Time | Nodes |
| 0.696 | 0.3 | 125 | 0.1 | 121 | 0.2 | 63 | 0.1 | 44 |
| 0.710 | 0.3 | 125 | 0.1 | 121 | 0.2 | 63 | 0.1 | 44 |
| 0.750 | 0.4 | 190 | 0.2 | 179 | 0.3 | 99 | 0.1 | 61 |
| 0.770 | 0.5 | 226 | 0.3 | 215 | 0.3 | 99 | 0.1 | 61 |
| 0.850 | 1.1 | 396 | 0.4 | 311 | 1.1 | 278 | 0.3 | 130 |
| 0.888 | 3.4 | 934 | 1.3 | 873 | 2.8 | 643 | 0.2 | 100 |
| 0.900 | 3.4 | 934 | 1.3 | 873 | 2.7 | 643 | 0.2 | 100 |
| 0.932 | 5.4 | 1299 | 1.4 | 893 | 4.1 | 913 | 0.1 | 63 |
| 0.956 | 176.9 | 15220 | 130.2 | 14670 | 64.8 | 8511 | 2.1 | 1145 |
| 0.960 | 534.9 | 28575 | 429.3 | 27799 | 263.7 | 19072 | 62.5 | 7973 |
| 0.980 | | Max | | NNP | 6355.7 | 98294 | 11.7 | 3360 |

under one second, for all the examples. It is worthwhile noting that the test set for the original formulation (LSP) contains 178 vectors and the test set for model (LISP) contains 245 vectors.

We can remark the following facts:

- The walk-back procedure with a new order given by a penalized cost function ("WB new ordering") is, in general, more competitive than the original method ("Walk-back") in all the values of $\gamma$.
- As we expected, the instances under "Walk-back" find an optimum after a big number of nodes with respect to the total number of nodes. The instances with a penalized cost function usually find the optimum very quickly. Although we have not included the data in the tables, we must remark that the first feasible point found in "WB new ordering" appears very early in the run, if we compare it with "Walk-back".
- In the last row of "WB new ordering", with $\gamma = 0.98$ a very good feasible point is reached much earlier than for the case of $\gamma = 0.96$ (one to the last row), but it is harder to certify that it is the optimum. We do not order points by their costs, the trade-off done by the penalty cost function can give such a result, but its performance is much better than the original method.
- Certifying a point to be optimum is a heavy task, and all the methods give similar results.
- We have expanded the range of values of the probability $\gamma$ with respect to [30, Example 4] to show that the new ordering method not only increases the speed of the process, but it also allows to solve more difficult cases.
- The certificate of optimality is very sensitive to small variations in the value of $\gamma$, as Table 2 shows in the high increment of processed nodes. Finding an optimum is more robust to this change, and the time stays under 4 minutes. The ordering of pending nodes by a penalized cost function can be used then as a search tool, as long as a good point can be reached very quickly.

Obviously, the choice of the value of $\gamma$ should be made thoroughly, because it is easy to create infeasible problems.

4.1.3. *Other formulation to run solvers.* This joint chance constraint problem with sampling can be solved as an MILP with the use of big-M notation (see [26], [22], [4]), but with *another completely different integer linear formulation.* Each record in the sample generates $m$ constraints in the new linear problem. This approach has been implemented in the model language GAMS and solved with Gurobi ([13]), Mosek ([23]), Cbc ([8]) and Scip ([2]) in the Neos Server site. The results for the treated levels of probability in Table 2 appear in Table 3. For each solver, the column "Total" contains the CPU time to certify the optimum and the column "Optimum" the time when the solver found the solution. The label "Max" denotes that the run has reached the limits in Neos Server, and "NNP" means optimum not found.

Table 3. Linear formulation

| | Gurobi | | Mosek | | Cbc | | Scip | |
|---|---|---|---|---|---|---|---|---|
| $\gamma$ | Total | Optimum | Total | Optimum | Total | Optimum | Total | Optimum |
| 0.696 | 0.7 | 0.1 | 0.8 | 0.7 | 10.3 | 7.5 | 3.8 | 3.6 |
| 0.71 | 0.9 | 0.1 | 0.9 | 0.9 | 16.0 | 8.6 | 3.7 | 3.5 |
| 0.750 | 1.6 | 1.0 | 2.7 | 2.6 | 9.5 | 8.0 | 7.4 | 6.3 |
| 0.770 | 1.1 | 1.0 | 1.6 | 1.5 | 13.6 | 7.0 | 5.3 | 5.0 |
| 0.850 | 7.7 | 6.0 | Max | 117.2 | 41.9 | 23.3 | 5.8 | 2.2 |
| 0.888 | 5.1 | 4.0 | Max | 337.4 | 31.7 | 31.3 | 16.6 | 7.4 |
| 0.900 | 17.4 | 17.0 | Max | 265.6 | 18.1 | 17.0 | 8.5 | 4.6 |
| 0.932 | 12.9 | 2.0 | 48.1 | 3.7 | 17.8 | 12.3 | 8.1 | 6.3 |
| 0.956 | 12.7 | 1.0 | Max | 2.7 | 48.8 | 41.5 | 23.9 | 21.3 |
| 0.960 | 41.3 | 34.0 | Max | 294.0 | 241.4 | 229.4 | 23.9 | 17.5 |
| 0.980 | 39.6 | 16.0 | Max | NNP | 75.7 | 38.8 | 32.5 | 30.4 |

As it is expected, linear solvers are in general more robust than nonlinear ones. For example, the results for Mosek are quite erratic. For easy cases ($\gamma \leq 0.9$), our approach is faster in both finding the optimal solution and proving optimality. For hard cases ($\gamma \geq 0.95$), our approach is competitive with the best MILP codes regarding finding optimal solutions, but vastly inferior to the best MILP codes regarding proving optimality.

4.2. **Reliability.** Optimization of the design of engineering systems is an important problem. We deal with the case of a series-parallel system, minimizing its cost, subject to a desired level of reliability. Usually redundancy allocation of components is used to achieve the reliability goal. We assume that for each subsystem, there are different types of components. The constraint which calculates the system reliability is a nonlinear and non-separable constraint, so we have a nonlinear integer programming problem. Even considering only one type of component for each subsystem, the problem is NP-hard [10].

For real problems, the size of the search space is huge. In the specialized literature most papers give heuristic methods to find efficient approaches. They are based on Tabu Search [24], Genetic Algorithms [12] or Ant Colony Optimization [3], among many other techniques.

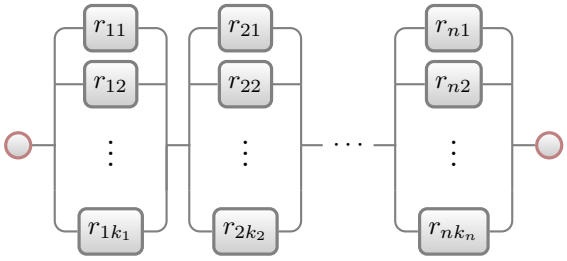


FIGURE 1. A parallel-series system with multiple choice components

4.2.1. *Statement of the problem.* We use the following notation:

- $n$ number of subsystems.
- $k_i$ number of different types of available components for the $i$-th subsystem.
- $r_{ij}$ reliability of the $j$-th component for the $i$-th subsystem.
- $c_{ij}$ cost of the $j$-th component for the $i$-th subsystem.
- $u_{ij}$ upper bound of number of the $j$-th component for the $i$-th subsystem.
- $R_0$ admissible level of reliability of the whole system.
- $x_{ij}$ number of the $j$-th component used in the $i$-th subsystem.

We consider a model of series-parallel with $n$ subsystems which is illustrated in Figure 1. Some assumptions are made in the following:

- Components have two states: operative or failed.
- The component reliabilities are known and deterministic.
- Failure of individual components are independent.
- Failed components do not damage other components, and they are not repaired.

The nonlinear integer redundancy allocation problem can be formulated as:

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

4.2.2. *Associated test set.* Consider an integer linear programming problem, denoted as (LIRP), which is the relaxed redundancy allocation problem without the reliability constraint, that is:

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

Each inequality can be converted to an equality with a new slack variable:

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

Usually the computation of the Gröbner basis is the bottleneck phase of this approach, but in our case (LIRP) is described by a totally unimodular matrix. Therefore, the set of circuits is known to be its Universal Gröbner basis [19]. However, for our purpose a Gröbner basis for a specific monomial order is needed. This basis can be computed explicitly.

For each $i = 1,\ldots,n$, the costs $c_{ij}$ can be sorted in descending order: $c_{iq} \geq c_{ip}$ if $q < p$. Let $N$ be the sum $k_1 + \cdots + k_n$ and define the matrices

$$D_{n\times N} = \begin{pmatrix} \overbrace{1\ldots1}^{k_1} & \overbrace{0\ldots0}^{k_2} & \ldots & \overbrace{0\ldots0}^{k_n} \\ 0\ldots0 & 1\ldots1 & \ldots & 0\ldots0 \\ & & \ddots & \\ 0\ldots0 & 0\ldots0 & \ldots & 1\ldots1 \end{pmatrix}, \quad C_{k'_i\times k_i} = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & -1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & -1 & \cdots & 0 \\ \vdots & & \vdots & & \ddots & \vdots \\ 1 & 0 & 0 & 0 & \cdots & -1 \\ 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & 0 & \cdots & -1 \\ \vdots & & \vdots & & \vdots & \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

where $k'_i = \binom{k_i}{2}$. Under these conditions, we have the following result:

**Theorem 4.2.** *The rows of the matrix*

$$\left(\begin{array}{cccc|c|cccc} \multicolumn{4}{c|}{I_N} & (D_{n\times N})^t & \multicolumn{4}{c}{-I_N} \\ \hline C_{k'_1\times k_1} & 0 & 0 & 0 & 0 & -C_{k'_1\times k_1} & 0 & 0 & 0 \\ 0 & C_{k'_2\times k_2} & 0 & 0 & 0 & 0 & -C_{k'_2\times k_2} & 0 & 0 \\ \vdots & & \ddots & & \vdots & & & \ddots & \\ 0 & 0 & \cdots & C_{k'_n\times k_n} & 0 & 0 & 0 & 0 & -C_{k'_n\times k_n} \end{array}\right),$$

*form a test set of* (LIRP) *with respect to the monomial order defined by the cost function, where the leftmost block corresponds to the variables*

$$x_{11}, \ldots, x_{1k_1}, x_{21}, \ldots, x_{nk_n},$$

*the center block to the variables* $d_1, \ldots, d_N$ *and the rightmost block to the variables* $t_{11}, \ldots, t_{1k_1}, t_{21}, \ldots, t_{nk_n}$.

For a complete proof of this result see [17].

4.2.3. *Penalized cost function.* The walk-back process for this problem is greatly improved by the knowledge of a feasible point $\boldsymbol{y}^0$ of (RP), which provides a good initial upper bound. There are lots of heuristic methods to obtain such a point. In this case, a greedy algorithm similar to [15] or [25] is used, which obtains a feasible point $\boldsymbol{y}^0$, with a cost $\sum_{ij} c_{ij} y^0_{ij} = c^0$ and a reliability $R_0$. Obviously, the optimal solution of (RP) has a cost less than or equal to $c^0$. Let $\boldsymbol{\beta}$ be the optimum of (LIRP) with reliability $R_{\boldsymbol{\beta}}$.

The penalized cost function can be defined in the following way, inspired by Theorem 3.1:

$$T(\boldsymbol{x}, \mu) = \boldsymbol{c}^t \cdot \boldsymbol{x} + \mu \cdot \max\{0, (R_0 - R(\boldsymbol{x}))\}, \quad \text{where} \quad \mu = \frac{c_Y - \boldsymbol{c}^t \cdot \boldsymbol{\beta}}{R_0 - R_{\boldsymbol{\beta}}},$$

with $c_Y$ the cost of the best point found for (RP). Initially, $c_Y = c^0$.

An improvement of the method consists in changing after a short period of time the penalized function. The list $\mathcal{P}$ of pending nodes is ordered according to the penalized cost function for a fixed amount of nodes $L$. After that, when it is likely a good feasible point has been found, the ordering is changed to $T_2(\boldsymbol{x}) = -\boldsymbol{c}^t \cdot \boldsymbol{x}$, that is, a "Depth first search" [5]. This assures a fast search for the optimum and a set $\mathcal{P}$ with a medium size. The parameter used is described in the next subsection.

4.2.4. *Computational results.* In all cases $n$ and $k$ refer to the number of subsystems, and the number of different components in each subsystem, respectively. For simplicity, the same number of components has been considered in each system $k_i = k$. The complexity of the problem depends on $\sum_i k_i$, different values of $k_i$ do not increase the problem size. The term "Walk-back" collects the results obtained with the algorithm published in [17]. The term "WB new ordering" summarizes the data for the new ordering given by the penalized cost function. For all the examples a summary table reports the average of a total of 30 instances run for each case.

The columns show the average CPU time in seconds and nodes visited, both data computed for the cases when both algorithms certified optimality visiting 80000 nodes or less. The parameter $L$ described in the previous subsection for the improvement is set to 2000. The column "> Limit" counts how many cases did not reach optimality.

4.2.5. *No correlation between cost and reliability, high reliability values.* In the first case the sample is randomly generated from a uniform distribution, as in [25], with $u_{ij} = 4$, $c_{ij} \in [10, 20]$, and values of reliability very close to 1: $r_{ij} \in [0.99, 0.998]$. The reliability objective is set to $R_0 = 0.90$. There is no correlation between cost and reliability of components, so it is likely that certain components are easily discarded. In Table 4 the results of the experiment are summarized.

TABLE 4. No correlation, high values of $r_{ij} \in [0.99, 0.998], c_{ij} \in [10, 20], u_{ij} = 4, R_0 = 0.90$, average time in seconds and nodes

| | | Walk-back | | | WB new ordering | | |
|---|---|---|---|---|---|---|---|
| $n$ | $k$ | Time | Nodes | $>$ Limit | Time | Nodes | $>$ Limit |
| 17 | 4 | 0.2 | 58 | 1 | 0.1 | 23 | 1 |
| 19 | 2 | 40.1 | 3389 | 1 | 9.1 | 2255 | 1 |
| 20 | 2 | 173.7 | 7254 | 6 | 72.7 | 7177 | 5 |

4.2.6. *Correlation between cost and reliability, high reliability values.* In this case we use the hypothesis that the greater the reliability component, the greater cost. The size of the systems have changed, due to the increased difficulty of the cases. The ranges of reliability and cost are similar to the first family of examples. Table 5 contains the summarized data.

TABLE 5. Correlation, high values of $r_{ij} \in [0.99, 0.998], c_{ij} \in [10, 20], u_{ij} = 4, R_0 = 0.90$, average time in seconds and nodes

| | | Walk-back | | | WB new ordering | | |
|---|---|---|---|---|---|---|---|
| $n$ | $k$ | Time | Nodes | $>$ Limit | Time | Nodes | $>$ Limit |
| 15 | 3 | 322.8 | 6965 | 1 | 42.3 | 4795 | 0 |
| 15 | 4 | 571.1 | 17629 | 13 | 432.6 | 21843 | 8 |
| 17 | 2 | 92.7 | 6465 | 1 | 15.4 | 3813 | 1 |

4.2.7. *Correlation between cost and reliability, smaller reliability values.* The last family of examples has the reliability and cost also correlated. When the range of the reliability of components is changed, the size of the problem increases very quickly. However if the range of cost is changed, the difficulty of the problem is the same. Here we consider smaller reliability values of components $r_{ij}$ in the range $[0.90, 0.99]$. Table 6 summarizes the collected data.

TABLE 6. Correlation, low values $r_{ij} \in [0.90, 0.99], c_{ij} \in [10, 20], u_{ij} = 4, R_0 = 0.90$, average time in seconds and nodes

| | | Walk-back | | | WB new ordering | | |
|---|---|---|---|---|---|---|---|
| $n$ | $k$ | Time | Nodes | $>$ Limit | Time | Nodes | $>$ Limit |
| 7 | 5 | 23.6 | 6157 | 0 | 17.9 | 5301 | 0 |
| 8 | 4 | 593.5 | 37728 | 7 | 373.4 | 34754 | 1 |

4.2.8. *Comments about the tables.* We highlight the following facts about the improvements:

- In all the cases when the walk-back algorithm did not find any improvement to the initial estimation, the new ordering procedure gives a better point.
- In almost the entire set of examples the optimum has been found with the new ordering algorithm in less than one second.

- When the walk-back procedure certifies optimality, the optimum is usually found after a large number of nodes explored during the search. The new ordering method finds it very quickly and most part of the time is used to certify optimality.
- The required time to certify optimality has highly decreased in all the cases. This fact is due to the decrease in the time needed to reach an initial feasible point. It contributes to speeding up the process through pruning paths. As an illustrative example in Table 5 the average time needed in the walk-back algorithm for $n = 15$, $k = 4$ is 571.1 s, and the new ordering method requires 432.6 s, an improvement of 25%.
- The performance is better than that appeared in [25].

4.2.9. *Comparison with other solvers.* In order to perform further comparisons with standard integer nonlinear solvers as BARON ([29]), COUENNE ([6]) or BONMIN ([7]), the problem (RP) has been modeled in GAMS format. This encoding has allowed us to launch our data sets in the Neos Server ([14], [18], [16]) under these solvers. We have chosen the 30 instances for systems with $n = 8, k = 4$ from Table 6, because it contains a good assortment of cases. A summary of the results are reported in Table 7 that shows average time to certify optimality and in Table 8 that shows the time required to reach the optimum (Couenne is not considered in the last Table).

TABLE 7. Comparison of solvers for the case $n = 8, k = 4$, time to certify optimality

| | WB new ordering | Baron | Couenne | Bonmin |
|---|---|---|---|---|
| Avg. Time | 373.4 | 53.9 | 1020.9 | 272.8 |
| N/F | 0 | 0 | 9 | 0 |

TABLE 8. Comparison of solvers for the case $n = 8, k = 4$, time to rearch the optimum

| | WB new ordering | Baron | Bonmin |
|---|---|---|---|
| Avg. Time | 2.8 | 25.7 | 209.4 |

The columns identify the solvers. The row "Avg. Time" contains the average time of CPU reported by the outputs for the 30 instances. The row "N/F" indicates the number of examples that the system has not been able to finish.

We note the following facts:

- The packages BONMIN and BARON have, in general, a better performance to certify optimality.
- In some cases we have observed that BARON returns a nonoptimal point, although it is very close to the minimum cost. The walk-back procedure and BONMIN always return an actual optimal point.
- The walk-back with new ordering found the optimum very quickly, usually under 3 seconds, as can be seen in Figure 2. This chart compares the times used by walk-back, BONMIN and BARON to find the optimum. COUENNE reports much larger times.

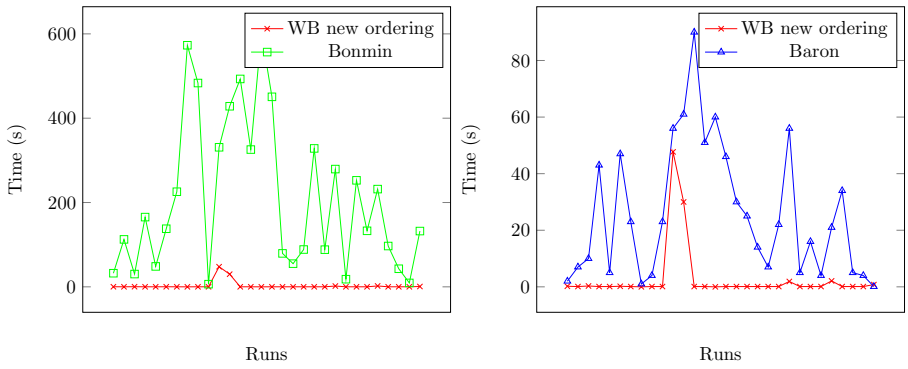


FIGURE 2. Time to find an optimum: WB new ordering, Bonmin and Baron

A natural question is how to combine the speed of the walk-back procedure with new ordering to find the optimum, and the good approach of a nonlinear solver like BARON to certify optimality of the point. This has been done including the point found by the walk-back procedure in the GAMS code as an initial feasible point. In all the cases, the point is certified with the right optimal cost, and the CPU time is significantly reduced, as it can be seen in Figure 3.

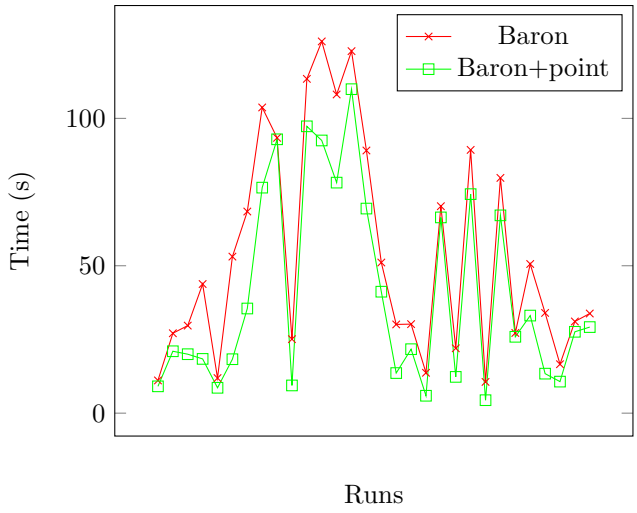


FIGURE 3. Time to certify: Baron and Baron with initial point given by WB

## 5. CONCLUSIONS

The walk-back procedure is based on extracting certain linear constraints from a nonlinear integer problem for which a test set can be easily computed. Starting from an optimum for the relaxed problem, and using the reverse test set, all the feasible points for the original problem can be reached, and an optimum can be found. The method has been improved by giving a criterion to sort the set of

points to be inspected by a penalized cost function. This new cost function trades off the cost of a point against its distance to the feasible region. As a result, new better points are obtained faster than with the order only by cost. Two real design engineering applications have been presented: the scheduling of jobs on parallel machines given restrictions on demands and capacity to minimize costs, and the series parallel redundancy allocation problem given a target reliability. The first problem includes a probabilistic constraint with random variables in the technology matrix, that can not be treated with nonlinear solvers. Our approach deals with any computable constraint. This problem can be reformulated as a linear integer program, and still in certain cases our approach is competitive with the best MILP solvers. The second application considered contains a nonlinear restriction given by the reliability function. This nonlinear model can be compared with other solvers with remarkable results. We conclude, through extensive computations, that both applications show that the combination of the walk-back procedure and an improved ordering is a promising tool as an exact method to solve nonlinear integer programs.

## Acknowledgments

This paper has been partially supported by Junta de Andalucía under grant FQM-5849, and Ministerio de Ciencia e Innovación MTM2010-19336, MTM2010-19576, MTM2013-46962-C2-1-P and FEDER. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

## References

[1] 4ti2 team. 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces. Available at www.4ti2.de.
[2] Achterberg, T. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, (2009).
[3] Ahmadizar, F. and Soltanpanah, H. Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach. *Expert Systems with Applications*, 38(4):3640–3646, (2011).
[4] Ahmed, S. and Shapiro, A. Solving Chance-Constrained Stochastic Programs via Sampling and Integer Programming. Available at www2.isye.gatech.edu/people/faculty/Shabbir-Ahmed/cctutorial.pdf. Accessed 30 April 2014.
[5] Avis, D. and Fukuda, K. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, (1996).
[6] Belotti, P. Couenne. Available at www.coin-or.org/Couenne/.
[7] Bonami, P. and contributors. Bonmin — basic open-source nonlinear mixed integer programming. Available at https://projects.coin-or.org/Bonmin.
[8] Cbc (Coin-or branch and cut). Available at https://projects.coin-or.org/Cbc.
[9] Castro, F., Gago, J., Hartillo, I., Puerto, J., and Ucha, J.M. An algebraic approach to integer portfolio problems. *European Journal of Operational Research*, 210(3):647–659, (2011).
[10] Chern, M.S. On the computational-complexity of reliability redundancy allocation in a series system. *Operations Research Letters*, 11(5):309–315, (1992).
[11] Coit, D. W., Smith, A.E. and Tate, D.M. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS J.Comput.*, 8(2):173–182, (1996).
[12] Coit, D.W. and Smith, A.E. Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, 45(2):254–260, (1996).
[13] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. Available at http://www.gurobi.com.
[14] Czyzyk, J., Mesnier, M.P. and Moré, J.J. The NEOS server. *IEEE Computational Science and Engineering*, 5(3):68–75, (1998).

[15] Djerdjour, M. and Rekab, K. A branch and bound algorithm for designing reliable systems at a minimum cost. *Applied Mathematics and Computation*, 118(2-3):247–259, (2001).
[16] Dolan, E. The NEOS server 4.0 administrative guide. Technical Report Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, (2001).
[17] Gago, J., Hartillo, I., Puerto, J., and Ucha, J.M. Exact cost minimization of a series-parallel system. *Computers and Operations Research*, 40(11):2752–2759, (2013).
[18] Gropp, W. and Moré, J.J. *Optimization environments and the NEOS server*, pages 167–182. Approximation theory and optimization (Cambridge, 1996). Cambridge Univ. Press, Cambridge, (1997).
[19] Hemmecke, R. and Schultz, R. Decomposition of test sets in stochastic integer programming. *Mathematical Programming*, 94(2-3):323–341, (2003).
[20] Li, D. and Sun, X. *Nonlinear integer programming*. Springer, New York, (2006).
[21] Li, Q., Guo, Y.K., Darlington, J. and Ida, T. Minimised geometric Buchberger algorithm for integer programming. *Annals of Operations Research*, 108(1-4), (2001).
[22] Luedtke, J., Ahmed, S., and Nemhauser, G. An integer programming approach for linear programs with probabilistic constraints. Lecture Notes in Computer Science 4513, pages 410–423, Berlin, 2007. Springer-Verlag.
[23] MOSEK ApS, Fruebjergvej 3, Boks 16, 2100 Copenhagen O, Denmark. The MOSEK optimization tools manual version 6.0, 2009. Available at http://www.mosek.com/.
[24] Ouzineb, M., Nourelfath, M., and Gendreau, M. Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems. *Reliability Engineering and System Safety*, 93(8):1257–1272, (2008).
[25] Ruan, N. and Sun, X. An exact algorithm for cost minimization in series reliability systems with multiple component choices. *Applied Mathematics and Computation*, 181(1):732–741, (2006).
[26] Ruszczyński, A. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93 (2), 195–215 (2002).
[27] Schlueter, M. and Gerdts, M. The oracle penalty method. *Journal of Global Optimization*, 47(2):293–325, (2010).
[28] Sturmfels, B. *Gröbner bases and convex polytopes*, volume 8. American Mathematical Society, Providence, RI, (1996).
[29] Tawarmalani, M. and Sahinidis, N.V. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, (2005).
[30] Tayur, S.R., Thomas, R.R., and Natraj, N.R. An algebraic-geometry algorithm for scheduling in presence of setups and correlated demands. *Mathematical Programming*, 69(3):369–401, (1995).

## Appendix A. Proof of Theorem 4.1

Let $A$ be the matrix associated with the restrictions for Problem (LSP-II). Then $A$ can be divided in five blocks according to the five sets of variables $y_{ij}, a_{ij}, z_{ij}, b_{ij}$ and $c_{ij}$. Let $I$ denote the identity matrix of size $mn$ and $\boldsymbol{O}$ denote a matrix with all entries zero. Let $D$ be the block diagonal $(0,1)$-coefficient matrix associated with constraints (7). Let $-MI$ denote the coefficient matrix of the variables $z_{ij}, i = 1,\ldots,n, j = 1,\ldots,m$ in constraints (8). We can then write $A$ as the $(n+3mn)\times 5mn$ integer matrix

$$A = \left(\begin{array}{c|c|c|c|c} D & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ I & I & -MI & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & I & I & \boldsymbol{O} \\ -I & \boldsymbol{O} & I & \boldsymbol{O} & I \end{array}\right)$$

We note $t = (t_y, t_a, t_z, t_b, t_c), S = \{t \mid At = 0\}$ and the block $t_y$ represents the variables $(Y_{11},\ldots,Y_{mn})$, noted in capital letters. Similar for the other blocks. Consider the following cases:

(1) $K_1 = \{t \in S \mid t_y = t_b = 0\}$. Let $S' = \ker A'$, where

$$A' = \begin{pmatrix} I & -MI & \boldsymbol{O} \\ \boldsymbol{O} & I & \boldsymbol{O} \\ \boldsymbol{O} & I & I \end{pmatrix} \to \begin{pmatrix} I & -MI & \boldsymbol{O} \\ \boldsymbol{O} & I & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & I \end{pmatrix},$$

which is not singular. Then $K_1 = 0$ and there is no binomial $x^\alpha - x^\beta$ such that contains only the variables $a_{ij}, z_{ij}, c_{ij}$.

(2) $K_2 = \{t \in S \mid t_b = 0\}$. Let $S'' = \ker A''$, where

$$A'' = \begin{pmatrix} D & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ I & I & -MI & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & I & \boldsymbol{O} \\ -I & \boldsymbol{O} & I & I \end{pmatrix}.$$

By the previous case, we can assume $t_y \neq 0$. The rows in the third block imply that $t_z = 0$. Then $t \in K_2$ if and only if $(t_y, t_a, t_c) \in S''' = \ker A'''$, for

$$A''' = \begin{pmatrix} D & \boldsymbol{O} & \boldsymbol{O} \\ I & I & \boldsymbol{O} \\ -I & \boldsymbol{O} & I \end{pmatrix}.$$

Let $Y_{ip}$ be the leftmost nonzero component in $t_y$. We can assume $Y_{ip} > 0$, because $S'''$ is a vector space. The row block corresponding to $D$ implies that there exists $q > p$ such that $Y_{iq} < 0$. Then

$$y_{ip} \text{ divides } x^{t^+}, y_{iq} \text{ divides } x^{t^-}.$$

The second block of rows in $A'''$ implies $A_{ip} = -Y_{ip} < 0, A_{iq} = -Y_{iq} > 0$. From the third block of rows, $C_{ip} = Y_{ip} > 0, C_{iq} = A_{iq} < 0$. Then

$$y_{ip}a_{iq}c_{ip} \text{ divides } x^{t^+}, y_{iq}a_{ip}c_{iq} \text{ divides } x^{t^-}.$$

The initial term in $x^{t^+} - x^{t^-}$ with respect to $>$ is $x^{t^+}$ because $y_{ip}$ divides $x^{t^+}$ and $y_{ip}$ it the greatest monomial in the binomial. Then the initial term of

$$y_{ip}a_{iq}c_{ip} - y_{iq}a_{ip}c_{iq} \tag{12}$$

divides the initial term of $x^{t^+} - x^{t^-}$.

(3) Let $S = \ker A$. By (a) and (b) we suppose $t_b \neq 0$. Let $B_{rs}$ the leftmost nonzero component in $t_b$. As in the previous case, take $B_{rs} > 0$. Because $b$ is the set of greatest variables for our term order, $x^{t^+}$ is the leading term. We also have $Z_{rs} = -B_{rs} < 0$.

(a) If $Y_{rs} \geq 0$ then

$$-Y_{rs} + Z_{rs} + C_{rs} = 0$$

then

$$b_{rs}c_{rs} \text{ divides } x^{t^+}$$

and the initial term of

$$b_{rs}c_{rs} - z_{rs}a_{rs}^{M_r}. \tag{13}$$

divides $x^{t^+}$.

(b) $Y_{rs} < 0$. Then there exists $Y_{rq}$, $s \neq q$ with $Y_{rq} = -Y_{rs} > 0$ .

(i) If $C_{rq} > 0$, then $b_{rs}y_{rq}c_{rq}$ divides $x^{t^+}$. The initial term of

$$b_{rs}y_{rq}c_{rq} - z_{rs}y_{rs}a_{rs}^{M_r-1}a_{rq}, s \neq q. \tag{14}$$

divides the initial term of $x^{t^+} - x^{t^-}$.

(ii) If $C_{rq} \leq 0$ then $0 < Y_{rq} \leq Z_{rq}$, which implies $B_{rq} < 0$ so $s < q$, because $B_{rs}$ was the first nonzero element. The rows in the second block imply that

$$Y_{rq} + A_{rq} - M_r Z_{rq} = 0, \text{ so } A_{rq} = M_r Z_{rq} - Y_{rq} \geq M_r Y_{rq} - Y_{rq} = (M_r - 1)Y_{rq}.$$

Then the initial term of

$$b_{rs}y_{rq}z_{rq}a_{rq}^{M_r-1} - z_{rs}y_{rs}b_{rq}a_{rs}^{M_r-1}, s < q \tag{15}$$

divides the initial term of $x^{t^+} - x^{t^-}$.

We have reviewed all the cases, so the sets (12), (13), (14) and (15) form the reduced Gröbner basis.

Depto. de Álgebra, Universidad de Sevilla. Apdo. 1160, E-41080 Sevilla (Spain)
*E-mail address*: gago@us.es

Dpto. de Matemática Aplicada I, E.T.S. de Ingeniería Informática, Av. Reina Mercedes, s/n, 41012 Sevilla, Spain
*E-mail address*: hartillo@us.es

Dpto. de Estadística e I.O., Facultad de Matemáticas, apdo. 1160, 41080 Sevilla, Spain
*E-mail address*: puerto@us.es

Depto. de Álgebra, Universidad de Sevilla. Apdo. 1160, E-41080 Sevilla (Spain)
*E-mail address*: ucha@us.es