Finding multiplies solutions for non-linear integer programming

José Manuel Jiménez Cobano [2]

Instituto de Matemática
Universidad de Sevilla

José María Ucha Enríquez [1,3]

Departamento de Matemática Aplicada I
Universidad de Sevilla

Abstract

We explain how to compute all the solutions of a nonlinear integer problem using the algebraic test-sets associated to some linear subproblem. These test-sets are obtained using Gröbner bases. We compare our method with previous approaches .

Keywords: Integer non-lineal optimization, multiplies solutions, test-sets, Groebner basis

[1] Second author is partially funded by MTM2016-75024-P, P12-FQM-2696, MTM2016-74983-C2-1-R and FQM-333.

[2] Email: josjimcob@alum.us.es

[3] Email: ucha@us.es

1 Introduction

In many real-life combinatorial optimization problems is of great interest for the decision-maker to have not only one solution, but the set of *all* optimal solutions (see [10] or [7], for example). The information provided by this set can give some additional information about the solutions, and sometimes is a first step for multi-objective optimization. On the other hand, sometimes these problems requires non-linear constraints to be modeled properly.

A method for problems of the form

$$\begin{aligned} \min\ & cx^t \\ \text{s.t. } & Ax^t \leq b^t \\ & x \in \Omega \\ & x \in \mathbb{N}^n \end{aligned} \quad (P)$$

where $A \in \mathbb{Z}^{m\times n}, c \in \mathbb{Z}^n, b \in \mathbb{Z}^m$ and the region Ω is defined by linear and nonlinear constraints was proposed in [9]. This method makes use of the test-sets associated to the linear subproblem

$$\begin{aligned} \min\ & cx^t \\ \text{s.t. } & Ax^t \leq b^t \\ & x \in \mathbb{N}^n \end{aligned} \quad (P_L)$$

A set $T \subset \mathbb{Z}^n$ is a test-set associated to (P_L) if $T \subset \ker(A)$, and for any non optimal x feasible for P_L there exists a $t \in T$ such that $x - t$ is feasible and $c(x-t) < c(x)$. As a consequence, starting from a optimal point $\hat{x}$ of (P_L) you can *recover* the set of all the feasible points, adding elements of the test-set until you eventually complete all the feasible region. In this way you can obtain the optimal points of (P) *walking back* from the linear optimal point until you reach the region Ω. Technical details can be found in [9]. There are several ways of computing test-sets, and one of the most efficient is using Gröbner bases (see for example [3]) with the software `4ti2` (see [5]).

In [1] and [6] the method of [9] is applied to real-life size problems with very competitive results. In this work: 1) we explain how to modify the walk-back method to obtain all the optimal points and 2) we compare its performance with the natural generalization of the algorithm presented in [10] using the computational system COUENNE (see [2]).

2 An algebraic algorithm to find all optimal points.

The main idea of our method is quite simple: starting from optimal point of the linear problem (P_L), we add test-set vectors until we find the points inside the non-linear region Ω. We save *all* points that has the best cost into Ω. The pseudocode is the following one:

```
INPUT: c, A, b; Ω; optimal point β of P_L; T associated test-set of P_L.

Opt := ∅;
Leaves := {β + t|∀t ∈ T} ∩ ℕ^n
costOpt = ∞

IF β ∈ Ω
THEN Opt := {β};
        costOpt := cβ^t

WHILE (Leaves ≠ ∅) DO
    FOR h ∈ Leaves DO
        IF c(h) < costOpt
                Leaves = (Leaves \ {h}) ∪ ({h + t|∀t ∈ T} ∩ ℕ^n)
                IF h ∈ Ω
                THEN Opt = {h};
                        costOpt = ch^t;
                        Leaves = (Leaves \ {h}) ∪ ({h + t|∀t ∈ T} ∩ ℕ^n)

                        ♯ the list of old candidates is deleted
                        ♯ and updated with a new candidate

        ELSE IF c(h) > costOpt
        THEN Leaves = Leaves \ {h}
        ♯ these branches are discarded

        ELSE IF c(h) = costOpt
        THEN Leaves = (Leave \ {h}) ∪ ({h + t|∀t ∈ T} ∩ ℕ^n)
                IF h ∈ Ω
                THEN Opt = Opt ∪ {h};
                ♯ a new candidate to be an optimal point has been obtained

END WHILE

OUTPUT: Opt the set of all optimal points with cost costeOpt
```

3 Computational experiments: reliability of series-parallel systems

Reliability problems are considered an important measure in the design of engineering processes, in which a series of systems is similar to a chain in which all the components must operate, since the failure of one of these components will suppose the failure of the complete system.

A mathematical model to minimize the cost of the system is

$$
\begin{aligned}
&\min \sum_{i=0}^{n}\sum_{j=1}^{k} c_{ij}x_{ij} \\
&s.t.\ \ R(x) \geq R_0 \\
&\qquad \sum_{j=1}^{k} x_{ij} \geq 1,\ \forall i = 1,\dots,n \\
&\qquad 0 \leq x_{ij} \leq u_{ij},\ \forall i = 1,\dots,n,\ j = 1,\dots,k \\
&\qquad x_{ij} \in \mathbb{Z}^+
\end{aligned}
\tag{1}
$$

with $R(x) = \prod_{i=1}^{n}\left(1 - \prod_{j=1}^{k}(1-r_{ij})^{x_{ij}}\right)$. Where:

- n: number of subsystems
- k_i: number of different kinds of components in subsystem i.
- r_{ij}: reliability of component j in the subsystem i.
- c_{ij}: cost of component j in the subsystem i.
- l_{ij}, u_{ij}: upper and lower dimensions of the numbers of components j in the subsystem i. We can suppose that $l_{ij} = 0$.
- R_0: Minimum reliability required return for the whole system.
- x_{ij}: numbers of components j in the subsystem i.

In our experiments, we have considered that all subsystem have equal numbers of components, that is, $k_i = k,\ \forall i$.

- $r_{ij} \in (0.9, 0.99), \forall i = 1,\dots,n,\ j = 1,\dots,k$
- $c_{ij} \in \{1, 2, \dots, 10\}, \forall i = 1,\dots,n,\ j = 1,\dots,k$
- $R_0 = 0.9$
- $u_{ij} = 4,\ \forall i = 1,\dots,n,\ j = 1,\dots,k$
- $n \in \{3, 4\}$

- $k \in \{2, 3\}$

We have run about 120 examples to test our algorithm coded in Python in a computer with an Intel Core i5, 3.5 Ghz, 8 Gb of RAM, under Ubuntu. The General Cut row stands for the natural generalization of the method proposed in [10], we have been used COUENNE ([2]) for the example which have been sent to `neos-server.org`.

	Number of examples	% examples
Test-set	84	70 %
General cut	36	30 %

Table 1
Reliability examples $n = 3, 4$, $k = 2, 3$ with all solutions.

We can observe that:

	Number of examples	% examples
Test-set	42	97,67 %
General cut	1	2,33 %

Table 2
Reliability examples $n = 3, 4$, $k = 2, 3$ with multiplies solutions.

	Number of examples	% examples
Test-set	42	54,55 %
General cut	35	45,45 %

Table 3
Reliability examples $n = 3, 4$, $k = 2, 3$ an unique solution.

	3 Systems		4 Systems	
	2 components	3 components	2 components	3 components
Test-set	0,07	0,19	0,37	0,51
General cut	0,43	1,23	0,84	1,50

Table 4
Average CPU times Reliability examples $n = 3, 4$, $k = 2, 3$.

- The Test-Set method and the General Cut are both exact
- The General Cut is worse in CPU time than the test-set method.

References

[1] Castro, F. J.; Gago, J.; Hartillo, I.; Puerto, J.; Ucha, J. M. An algebraic approach to integer portfolio problems. European J. Oper. Res. 210 (2011), no. 3, 647–659.

[2] Belotti, Pietro. COUENNE: a user's manual. Document available at `https://www.coin-or.org/Couenne/`. https://www.coin-or.org/Couenne/

[3] Cox, David A.; Little, John; O'Shea, Donal. Using algebraic geometry. Second edition. Graduate Texts in Mathematics, 185. Springer, New York, 2005.

[4] Djerdjour M, Rekab K. A branch and bound algorithm for designing reliable systems at a minimum cost. Appl. Math. and Computation 118 (2001), 247–59.

[5] 4ti2 team. 4ti2 – A software package for algebraic, geometric and combinatorial problems on linear spaces. Available at www.4ti2.de, 2013.

[6] Gago, J.; Hartillo, I.; Puerto, J.; Ucha, J. M. Exact cost minimization of a series-parallel reliable system with multiple component choices using an algebraic method. Comput. Oper. Res. 40 (2013), no. 11, 2752–2759.

[7] Leão, Aline A. S.; Cherri, Luiz H.; Arenales, Marcos N. Determining the K-best solutions of knapsack problems. Comput. Oper. Res. 49 (2014), 71–82.

[8] Ruan N, Sun XL. An exact algorithm for cost minimization in series reliability systems with multiple component choices. Appl. Math. and Computation 181 (2006), 732–41.

[9] Tayur, Sridhar R.; Thomas, Rekha R.; Natraj, N. R. An algebraic geometry algorithm for scheduling in presence of setups and correlated demands. Math. Programming 69 (1995), no. 3, Ser. A, 369–401.

[10] Tsai, Jung-Fa; Lin, Ming-Hua; Hu, Yi-Chung Finding multiple solutions to general integer linear programs. European J. Oper. Res. 184 (2008), no. 2, 802–809.

[11] Han-Lin Li, Tsai, Jung-Fa. A distributed computation algorithm for solving portfolio problems with integer variables. European J. Oper. Res. 186 (2008), no. 2, 882–891.