# SUDOKUS AND GRÖBNER BASES: NOT ONLY A *DIVERTIMENTO*

J. GAGO-VARGAS, M.I. HARTILLO-HERMOSO, J. MARTÍN-MORALES, AND J.M. UCHA-ENRÍQUEZ

Abstract. Sudoku is a logic-based placement puzzle. We recall how to translate this puzzle into a 9-colouring problem which is equivalent to a (big) algebraic system of polynomial equations. We study how far Gröbner bases techniques can be used to treat these systems produced by Sudokus. This general purpose tool can not be considered as a good solver, but we show that it can be useful to provide information on systems that are —in spite of their origin— hard to solve.

## 1. Introduction

During the last years some games called of 'Number place' type have become very popular. The target is to put some numbers or pieces on a board starting from some information given by other numbers. We have analyzed some of these puzzles and reduced them to equivalent algebraic systems of polynomial equations. We think that this modelling is itself a good motivation for students.

We have found that these systems —particularly in the case of Sudoku— are a good source of non-trivial examples to:

(1) Study the limits and applicability of the available solving methods.
(2) Compare the methods.

This work is a report on what can be expected of Gröbner bases as the natural first approach to this study. The reader is referred to the classical bibliography as [1], [4], [11] or [3] as excellent introductions to this subject.

## 2. Describing and modelling Sudoku

Sudoku is a puzzle that became very popular in Japan in 1986 and all around the world in 2005, although its origin happened in New York, under the name 'Number Place'. You have to fill in a $9 \times 9$ board divided in 9 regions of size $3 \times 3$ with the digits 1 to 9, starting from some numbers given on the board in such a way that two numbers cannot be repeated in any row, column or $3 \times 3$ region. A proper Sudoku has only one solution.

Sudoku can be expressed as a graph colouring problem:

- The graph has 81 vertices, one for each cell.
- You need 9 colours, one for each number.
- The edges are defined by the adjacency relations of Sudoku: where we want different numbers (taking into account rows, columns and regions) we need different colours.

All authors partially supported by MTM2004-01165 and FQM-333.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 9 | | | | 4 | | | 7 |
| | | | | | 7 | 9 | | |
| 8 | | | | | | | | |
| 4 | | 5 | 8 | | | | | |
| 3 | | | | | | | | 2 |
| | | | | | 9 | 7 | | 6 |
| | | | | | | | | 4 |
| | | 3 | 5 | | | | | |
| 2 | | | 6 | | | | 8 | |

FIGURE 1. A typical Sudoku

The resulting graph $\mathcal{G}$ is a regular graph with degree 20, so the number of edges of $\mathcal{G}$ is equal to $\frac{81 \cdot 20}{2} = 810$.

We can solve the colouring problem through a polynomial system ([2]. cf. [1], [11]) described by an ideal $\mathcal{I}$ of $\mathbb{Q}[x_1, \ldots, x_{81}]$ —a variable for each vertex— with the following generators $F(x_j), j = 1, \ldots, 81$ and $G(x_i, x_j), 1 \leq i < j \leq 81$:

- We will consider the colours numbered from 1 to 9. For each vertex $x_j$ we consider the polynomial $F(x_j) = \prod_{i=1}^{9}(x_j - i)$.
- If two vertexes are adjacent then $F(x_i) - F(x_j) = (x_i - x_j)G(x_i, x_j) = 0$, so the condition about different colours is given by adding the polynomial $G(x_i, x_j)$.

We number the cells in a Sudoku as in Figure 2.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 |
| 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 |
| 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 |

FIGURE 2. Cells enumeration

In addition, all the initial information of the Sudoku must be included. For example, if we want to solve the Sudoku in Figure 1, we have to add the following polynomials to the ideal $\mathcal{I}$:

$$x_2 - 9,\ x_6 - 4,\ x_9 - 7,\ x_{15} - 7,\ x_{16} - 9,\ x_{19} - 8,$$
$$x_{28} - 4,\ x_{30} - 5,\ x_{31} - 8,\ x_{37} - 3,$$
$$x_{45} - 2,\ x_{51} - 9,\ x_{52} - 7,\ x_{56} - 6,\ x_{63} - 4,$$
$$x_{66} - 3,\ x_{67} - 5,\ x_{73} - 2,\ x_{76} - 6,\ x_{80} - 8.$$

All the information about the solutions of a given Sudoku is contained in the set of zeros of $\mathcal{I}$ noted by $V(\mathcal{I})$:

$V(\mathcal{I}) = V_{\mathbb{Q}}(\mathcal{I}) = \{(s_1, \ldots, s_{81}) \in \mathbb{Q}^{81}$ such that $H(s_1, \ldots, s_{81}) = 0,$ for any $H \in \mathcal{I}\}$.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | | | | | | | | 8 |
| 5 | | | 2 | | 8 | | 6 | |
| | | 3 | 7 | 1 | | | | 9 |
| | | | | 7 | 3 | | 5 | |
| 2 | | | | | | | | 4 |
| | 5 | | 1 | 6 | | | | |
| 8 | | | | 2 | 7 | 3 | | |
| | 4 | | 3 | | 9 | | | 1 |
| 7 | | | | | | | | 2 |

FIGURE 3. Sudoku with 28 numbers

*Remark* 1. Once we have added the polynomials corresponding to initial data, it is easy to see that the polynomials $F(x_i)$ are redundant so we can delete them. The system of equations has 810 equations, one for each edge of the graph.

The following are elementary results:

**Proposition 1.** *A Sudoku has solution if and only if* $\mathcal{I} \neq \mathbb{Q}[x_1, \ldots, x_{81}]$ *if and only if any reduced Gröbner basis of* $\mathcal{I}$ *with respect to any term ordering is not* $\{1\}$.

**Proposition 2.** *If we start from a proper Sudoku then any reduced Gröbner basis of* $\mathcal{I}$ *with respect to any term ordering has the form* $G = \{x_i - a_i \mid i = 1, \ldots, 81\}$ *where every* $a_i$ *are numbers from* 1 *to* 9. *The numbers* $a_i$ *describe the solution.*

*Example* 1. Let us consider the problem from Figure 3. We have written the set of 810 equations and a program in a file available on the Internet[1] called `sudoku` that runs under SINGULAR [7]. The syntax is

```
<"sudoku";
intmat A[9][9] =
        9,0,0,0,0,0,0,0,8, 5,0,0,2,0,8,0,6,0,
        0,0,3,7,1,0,0,0,9,
        0,0,0,0,7,3,0,5,0, 2,0,0,0,0,0,0,0,4,
        0,5,0,1,6,0,0,0,0,
        8,0,0,0,2,7,3,0,0, 0,4,0,3,0,9,0,0,1,
        7,0,0,0,0,0,0,0,2;
def G = sudoku(A); vdim(G);
//used time: 1.65 sec
//-> 1
```

This Sudoku has a unique solution and is encoded in the reduced Gröbner basis $G$.

Unfortunately, in general the systems produced by Sudokus are not so friendly. Backtracking solvers (more or less guided by logic) are all over the web and are usually very fast. An interesting alternative method (which admits an algebraic approach too that has to be considered in the future) is that of the *dancing links* ([10]).

We think that writing down the equations and computing Gröbner bases is not a good solving method in general[2]. Nevertheless this approach has some advantages:

[1] `http://www.us.es/gmcedm/`

[2] This is something that perhaps could be expected: when you solve Sudokus by hand, you consider proper subsets of the initial data of the Sudoku that produce new values in the cells. The

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **9** | 2 | 6 | 5 | 3 | 4 | 7 | 1 | **8** |
| **5** | 7 | 1 | **2** | 9 | **8** | 4 | **6** | 3 |
| 4 | 8 | **3** | **7** | **1** | 6 | 5 | 2 | **9** |
| 1 | 9 | 8 | 4 | **7** | **3** | 2 | **5** | 6 |
| **2** | 6 | 7 | 9 | 8 | 5 | 1 | 3 | **4** |
| 3 | **5** | 4 | **1** | **6** | 2 | 9 | 8 | 7 |
| **8** | 1 | 9 | 6 | **2** | **7** | **3** | 4 | 5 |
| 6 | **4** | 2 | **3** | 5 | **9** | 8 | 7 | **1** |
| **7** | 3 | 5 | 8 | 4 | 1 | 6 | 9 | **2** |

FIGURE 4. Solution for Figure 3

if the Sudoku has many solutions the Gröbner bases allow us to obtain the number of solutions, as we will see in the next section.

## 3. Counting solutions

It is well known that, as ideals $\mathcal{I}$ produced by Sudokus are radical (cf. [4, Ch. 2, Prop. 2.7.]) the number of elements in $V(\mathcal{I})$ is equal to the dimension of the $\mathbb{Q}$-vector space $\mathbb{Q}[x_1, \ldots, x_{81}]/\mathcal{I}$, and that this number can be computed with *any* Gröbner basis $G$ with respect to any term ordering $<$:

**Proposition 3.** *(cf.* [1, Prop. 2.1.6.]*) A basis of the* $\mathbb{Q}$*-vector space* $\mathbb{Q}[x_1, \ldots, x_{81}]/\mathcal{I}$ *consists of the cosets of all the power products that are not divisible by* $lp_<(g_i)$ *for every* $g_i \in G$*.*

The Singular command ([7]) to obtain this invariant for a given ideal of a ring is `vdim`.

*Example* 2. Suppose now that we start from the Sudoku of Figure 3 but cells number 64 and 82 are empty.

```
A[6,4]=0; A[8,2]=0;
G=sudoku(A);
vdim(G);
//used time: 127.71 sec
//-> 53
```

Then there are 53 different solutions. To compute all of them

```
LIB "solve.lib";
def S = solve(G,5,0,"nodisplay");
setring S; size(SOL);
//-> 53
SOL[1];  //First solution in the list
```

In an analogous way we can see that if cell 26 is empty too there are 98 solutions.

*Example* 3. We have easily obtained that the number of different Sudokus $4 \times 4$ that has the same rules that the $9 \times 9$ but only four colours and four $2 \times 2$ regions is 288. The ideal to be considered is the one corresponding with no initial data.

---

polynomial approach in principle take into account *all the system at the same time* and does not take advantage of the subsystems, unless you choose ad hoc term orderings for each Sudoku.

The number of possible configurations for the case $9 \times 9$ is known ([6]) and it is a work in progress to apply a mixed approach between brute force and Gröbner bases computation to obtain the number of configurations. It should be pointed out that mixed volume might be another interesting approach.

*Example* 4. If the initial configuration of Figure 3 has the number 1 in cell number 4 then the problem has no solution: any reduced Gröbner basis is equal to $\{1\}$. To obtain that a given Sudoku has no solution is often in practice reasonably fast. So, although solving a given Sudoku can be very hard, it is not so hard in general to guess the value of a given cell trying the set of possible values.

## 4. Modelling more fashionable games

There exist many variants of the previous game. We briefly overview some of them and give their mathematical modelling with an algebraic system, above all because of their pedagogical interest. In general they are not colouring problems and in all cases Gröbner bases count the number of possible solutions.

4.1. **Variants of Sudoku.** The following games are variants of the classical Sudoku:

(1) *Killer Sudoku.* Instead of being given the values of a few individual cells, the sum of groups of cells are given. No duplicates are used within the groups. The algebraic system is built by adding to the 810 equations those that define the linear relations coming from the sums of groups of cells. For example, in Figure 5 the cells $x_1$ and $x_2$ give us the polynomial $x_1 + x_2 - 3$.

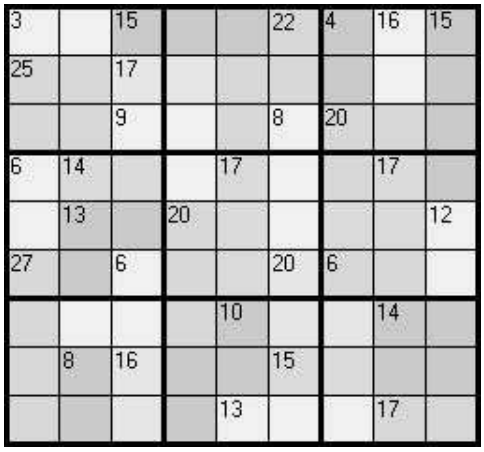


Figure 5. Killer Sudoku

(2) *Even-Odd Sudoku.* Fill in the grid so that every row, column, $3 \times 3$ box, contains the digits 1 through 9, with gray cells even, white cells odd. The grey cells bring out a polynomial of the form $\prod_{i \text{ even}}(x_j - i)$,
(3) 1-*way Disallowed number place.* All the places where orthogonally adjacent cells are consecutive numbers have been specially marked. If two cells $x_i$ y $x_j$ are adjacent we have to add the following equations. If they are specially

marked we have to write an equation of the form $(x_i - x_j - 1)(x_i - x_j + 1)$. If they don't then it is the negative proposition, so we can write it as $z(x_i - x_j - 1)(x_i - x_j + 1) - 1$, where $z$ is a new variable.

(4) *Greater than Sudoku.* It only appears "greater than" or "less than" signs in adjacent cells. The board is empty: we have not any data. Any relation of the form $x_i > x_j$ can be written as $x_i - x_j = b_{ij}$, where $b_{ij} \in \{1, 2, \ldots, 8\}$.
(5) *Geometry Sudoku.* The board is not rectangular, it can even be a torus. We only have to change the adjacency relations.
(6) *Factor Rooms.* It is similar to Killer Sudoku, but now with products and without $3 \times 3$ blocks.

4.2. **Kakuro.** The rules are

(1) Place a number from 1 to 9 in each empty cell.
(2) The sum of each vertical or horizontal block equals the number at the top or on the left of that block.
(3) Numbers may only be used once in each block.

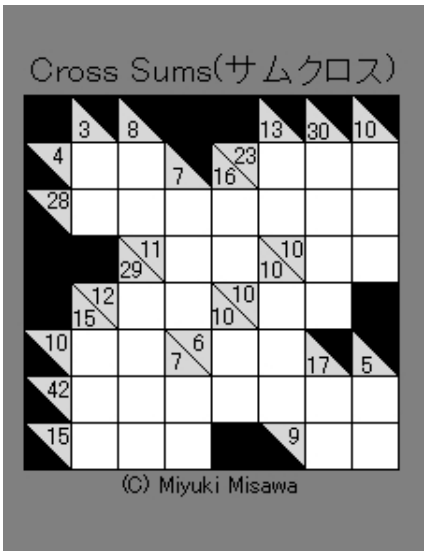


FIGURE 6. Kakuro

The equations are of the form

(1) $F(x_j) = \prod_{i=1}^{9}(x_j - i)$ for each cell.
(2) $G(x_j, x_k) = \frac{F(x_j) - F(x_k)}{x_j - x_k}$ for cells $(j, k)$ in the same block.
(3) Linear relations defined by the sums in each block.

For example, in Figure 6 we have the following linear relations for the first cell:

$$x_1 + x_2 - 4, x_1 + x_6 - 3.$$

4.3. **Bridges or Hashiwokakero.** This is another popular game in Japan. The rules are

(1) The number of bridges is the same as the number inside the island.
(2) There can be up to two bridges between two islands.

(3) Bridges cannot cross islands or other bridges.
(4) There is a continuous path connecting all the islands.

The unknowns are the bridges that cross from one island to other. For example, from the top left island in Figure 7, with value 3, we get variable $x_1$ (connection to right island) and $x_2$ (connection to down) (see Figure 8). Every $x_i$ can have the

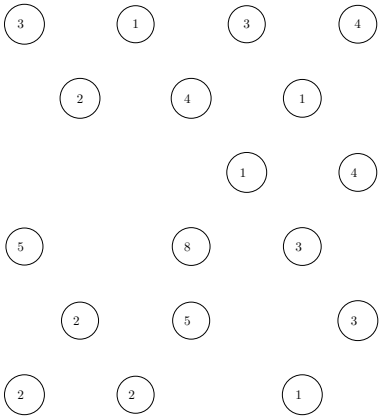


FIGURE 7. Bridges

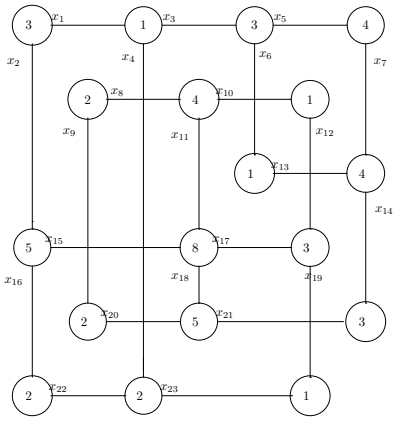


FIGURE 8. Bridges model

value $0, 1, 2$, so we include the polynomials $x_i(x_i - 1)(x_i - 2)$. The condition that the bridge $x_i$ cannot cross the bridge $x_j$ is equivalent to the equation $x_i x_j = 0$. Last, we have the linear relations given by the sum of bridges that start from an

island. In the previous example, we get

$$\begin{aligned} x_i(x_i-1)(x_i-2), i=1,\dots,23, \\ x_4x_8, x_4x_{15}, x_4x_{20}, x_6x_{10}, x_{12}x_{13}, x_9x_{15}, x_{19}x_{21}, \\ x_1+x_2-3, x_1+x_3+x_4-1, x_3+x_5+x_6-3, \\ x_5+x_7-4, x_8+x_9-2, x_8+x_{10}+x_{11}-4, \\ x_{10}+x_{12}-1, x_6+x_{13}-1, x_7+x_{13}+x_{14}-4, \\ x_2+x_{15}+x_{16}-5, x_{11}+x_{15}+x_{17}+x_{18}-8, \\ x_{12}+x_{17}+x_{19}-3, x_9+x_{20}-2, x_{18}+x_{20}+x_{21}-5, \\ x_{14}+x_{21}-3, x_{16}+x_{22}-2, x_4+x_{22}+x_{23}-2, x_{19}+x_{23}-1 \end{aligned}$$

It is easy to compute the solutions of this system. To extract those that define a connected graph is a different kettle of fish that we do not treat in this little section. The computation in SINGULAR is as follows.

```
ring r0=0,x(1..23),dp; option(redSB);

proc F (int i) { return(x(i)*(x(i)-1)*(x(i)-2)); };

ideal I;

for (i = 1; i<=23; i++) {I[i]=F(i); };

I = I, x(4)*x(8), x(4)*x(15), x(4)*x(20), x(6)*x(10), x(12)*x(13), x(9)*x(15), x(19)*x(21),
x(1)+x(2) -3, x(1)+x(3)+x(4)-1, x(3)+x(5)+x(6)-3, x(5)+x(7)-4, x(8)+x(9)-2, x(8)+x(10)+x(11)-4,
x(10) +x(12)-1, x(6)+x(13)-1, x(7)+x(13)+x(14)-4, x(2)+x(15)+x(16)-5, x(11)+x(15)+x(17)+x(18)-8,
x(12)+x(17)+x(19)-3, x(9)+x(20)-2, x(18)+x(20)+x(21)-5, x(14)+x(21)-3, x(16)+x(22)-2,
x(4)+x(22)+x(23)-2, x(19)+x(23)-1;

ideal Isol = std(I);

Isol;

Isol[1]=x(23)-1 Isol[2]=x(22)-1 Isol[3]=x(21)-1 Isol[4]=x(20)-2 Isol[5]=x(19) Isol[6]=x(18)-2
Isol[7]=x(17)-2 Isol[8]=x(16)-1 Isol[9]=x(15)-2 Isol[10]=x(14)-2 Isol[11]=x(13) Isol[12]=x(12)-1
Isol[13]=x(11)-2 Isol[14]=x(10) Isol[15]=x(9) Isol[16]=x(8)-2 Isol[17]=x(7)-2 Isol[18]=x(6)-1
Isol[19]=x(5)-2 Isol[20]=x(4) Isol[21]=x(3) Isol[22]=x(2)-2 Isol[23]=x(1)-1
```

4.4. **Minesweeper.** The target of this well-known Windows game is to uncover all the tiles that do not have a mine under them. When we click on a tile, if there is a mine under it, the game is over. If there is no mine under it, you will be given a number. The number will tell you how many mines are touching that tile (left, right, above and below). We assign variables to each unknown tile, with values 0 or 1. The relations between them are of the form $\sum_i x_i - k$.

*Remark* 2. There is a classical and interesting problem: given a board of positions with numbers, is it valid? In other words, is there any way in which the mines could be arranged in the hidden squares that would be consistent with those numbers? This problem is known to be NP-complete [8]. With the previous model, we have an algorithm to decide the consistency, through the computation of a Gröbner basis. A theoretical consequence of the polynomial modelling is that we obtain that the consistency of a system of polynomial equations of degree two (almost linear!) is NP-complete.

## 5. Fake shortcuts and experimental facts

Of course we have tried the following (a priori) tricks to speed up our implementation to manage Sudokus based in Gröbner bases:

- Work in a field of 9 elements instead of characteristic 0. No significant improvements.

- Work with the 9-th roots of the unit as colours. No significant improvements.
- Change the numbers of the colours to $-4, -3, -2, -1, 0, 1, 2, 3, 4$ to obtain nicer coefficients. No significant improvements.
- Use symmetric polynomials instead of the $G_i$ of section 2. No significant improvement.
- In the available options to compute Gröebner bases with Singular, the option `intStrategy` has been used: it avoids division of coefficients during standard basis computations. Without this option computations are often much slower.

On the other hand, we have tried to solve our Sudoku systems with some different available methods. Here are the initial experimental results:

- **Numerical methods**: In most examples, usual Newton-Rapshon (cf. [9]) methods —the way in which an engineer would possibly try to solve our systems— have not succeeded. It is a work in progress to show that for systems produced by Sudokus the usual numerical methods diverge for a big enough family of examples. It would mean that Sudoku systems could be regarded as ill-conditioned systems richer by far than the classical Wilkinson's monster (cf. [5]).
- **Numerical homotopy methods**: The numerical homotopy methods implemented in Jan Verschelde's software package `PHCpack` ([14]) are another way of solving algebraic systems of polynomial equations of great interest. They are known to be well suited to treat the multilinear case. They have been used, for example, to obtain totally mixed Nash equilibria (cf. [13]). Neither have they obtained correct solutions in most examples.

Sudoku systems seems to be somewhat resistant to a non-purely-symbolic approach, and we think that this pathological behavior demands itself a deeper understanding.

## References


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

[2] Bayer, Dave *The division algorithm and the Hilbert scheme.* Ph. D. Thesis. Harvard University, June 1982.

[3] T. Becker and V. Weispfenning. *Gröbner bases*, volume 141 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1993. A computational approach to commutative algebra, In cooperation with Heinz Kredel.

[4] Cox, D., J. Little, D. O'Shea. *Ideals , Varieties and Algorithms.* Springer, Berlin, 1997.

[5] Cox, D., J. Little, D. O'Shea. *Using Algebraic Geometry.* Springer, Berlin, 1998.

[6] B. Felgenhauer and F. Jarvis. Enumerating possible sudoku grids, 2005. `http://www.afjarvis.staff.shef.ac.uk/ sudoku/sudoku.pdf`. [Online; accessed 30-December-2005].

[7] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. `http://www.singular.uni-kl.de`. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005.

[8] R. Kaye. Minesweeper is NP-complete. *Math. Intelligencer*, 22(2):9–15, 2000.

[9] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. With separately available software.

[10] Knuth, D.E. *Dancing links.* Preprint.

[11] M. Kreuzer and L. Robbiano. *Computational commutative algebra. 1.* Springer-Verlag, Berlin, 2000.
[12] E. Pegg. Sudoku variations, 2005. `http://www.maa.org/editorial/mathgames/mathgames_09_05_05.html`. [Online; accessed 12-December-2005].
[13] Sturmfels, B. Solving Systems of Polynomial Equations. Amer.Math.Soc., CBMS Regional Conferences Series, No 97, Providence, Rhode Island, 2002.
[14] J. Verschelde. PHCcpack: A general-purpose solver for polynomial systems by homotopy continuation, 2005. ACM Transactions on Mathematical Software volume 25, number 2: 251–276, 1999.

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

Depto. de Matemáticas, Universidad de Cádiz. Apdo. 40, E-11510 Puerto Real (Spain)
*E-mail address*: `isabel.hartillo@uca.es`

Depto. de Matemáticas, Univ. de Zaragoza, Zaragoza, Spain
*E-mail address*: `jorge@unizar.es`

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