Gröbner Bases

Victor Adamchik

Carnegie Mellon University

Sudoku with Gröbner bases

Sudoku is a famous word puzzle. Players are given a 9×9 board and some numbers (typically at least 17), and are required to fill out the whole board while following 3 rules:

- Each row should contain digit 1 to 9 exactly once.
- Each column should contain digit 1 to 9 exactly once.
- Each 3×3 blocks should contain digit 1 to 9 exactly once

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

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

Algebraic Representation.

The general idea is to set one or more variables for each grid, and express the constraints by polynomial equations. Typically, there are three kinds of algebraic systems representing a Sudoku:

Roots-Of-Unity System

- For each grid (i, j), set a variable x_{ij}, so there are 81 variables in total.
- Use roots of $z^9 == 1$ to represent numbers from 1 to 9.
- For each grid (i, j), add an equation $x_{ij}^9 - 1 = 0$
- For two x_{ij} in each row, column or block, add an equation $\frac{x_{ij}^9 - x_{kl}^9}{x_{ij} - x_{kl}} = 0$

The total number of equation in the original system is

1053 = 81 + 3 (block, row, column) x 9 (board size) x 36 (pairwise combinations)

Sum - Product System

- For each grid (i, j), set a variable x_{ij}, so there are 81 variables in total.
- For each grid (i, j), add an equation $\prod_{s=1}^{9}(x_{ij}-s)=0$
- For two x_{ij} in each row, column or block, add an equation $\frac{\prod_{s=1}^{9}(x_{ij}-s)-\prod_{s=1}^{9}(x_{kl}-s)}{x_{ij}-x_{kl}}=0$

■ **Demo (Sum - Product System)**

In[1]:=

```
SudokuInteger[given_List] :=
 Module[{pos, i, j, k, l, first, m = 3, n = 9},
  Basis = {}; GB = {};

  (* Every grid is assigned a digit *)
  For[i = 1, i ≤ n^2, i++,
   AppendTo[Basis, (x[i] - 1) * (x[i] - 2) * (x[i] - 3) * (x[i] - 4) *
       (x[i] - 5) * (x[i] - 6) * (x[i] - 7) * (x[i] - 8) * (x[i] - 9)];
  ];

  diff :=
   Cancel[
     1
    ---
    a - b
     ((-9 + a) (-8 + a) (-7 + a) (-6 + a) (-5 + a) (-4 + a) (-3 + a) (-2 + a) (-1 + a) -
       (-9 + b) (-8 + b) (-7 + b) (-6 + b) (-5 + b) (-4 + b) (-3 + b)
        (-2 + b) (-1 + b))];
  (* and add constraints with a block *)
  For[i = 1, i ≤ m, i++,
   For[j = 1, j ≤ m, j++,
    first = (i - 1) * n * m + (j - 1) * m + 1;
    pos = {first, first + 1, first + 2, first + n, first + n + 1,
      first + n + 2, first + 2 n, first + 2 n + 1, first + 2 n + 2};
    For[k = 1, k ≤ n, k++,
     For[l = k + 1, l ≤ n, l++,
      AppendTo[Basis, diff /. {a → x[pos[[k]]], b → x[pos[[l]]]}];
     ]
    ]
```

```
  ]
 ];

 (* Constrains within a row *)
 For[i = 1, i ≤ n, i++,
  first = (i - 1) * n + 1;
  pos = {first, first + 1, first + 2, first + 3, first + 4, first + 5,
    first + 6, first + 7, first + 8};
  For[k = 1, k ≤ n, k++,
   For[l = k + 1, l ≤ n, l++,
    AppendTo[Basis, diff /. {a → x[pos[[k]]], b → x[pos[[l]]]}];
   ]
  ]
 ];

 (* Constrains within a column *)
 For[i = 1, i ≤ n, i++,
  first = i;
  pos = {first, first + n, first + 2 n, first + 3 n, first + 4 n,
    first + 5 n, first + 6 n, first + 7 n, first + 8 n};
  For[k = 1, k ≤ n, k++,
   For[l = k + 1, l ≤ n, l++,
    AppendTo[Basis, diff /. {a → x[pos[[k]]], b → x[pos[[l]]]}];
   ]
  ]
 ];

 (* Given numbers *)
 For[i = 1, i ≤ Length[given], i++,
  AppendTo[Basis, x[ (given[[i, 1]] - 1) n + given[[i, 2]] ] - given[[i, 3]] ]
 ];

 (* Solve the Sudoku *)
 GB = GroebnerBasis[Basis, Table[x[i], {i, 1, 81}] ]
]
```

Sudoku board:

In[2]:=
```
G28 = {{1, 1, 9}, {1, 9, 8}, {2, 1, 5}, {2, 4, 2}, {2, 6, 8}, {2, 8, 6},
    {3, 3, 3}, {3, 4, 7}, {3, 5, 1}, {3, 9, 9}, {4, 5, 7}, {4, 6, 3},
    {4, 8, 5}, {5, 1, 2}, {5, 9, 4}, {6, 2, 5}, {6, 4, 1}, {6, 5, 6},
    {7, 1, 8}, {7, 5, 2}, {7, 6, 7}, {7, 7, 3}, {8, 2, 4}, {8, 4, 3},
    {8, 6, 9}, {8, 9, 1}, {9, 1, 7}, {9, 9, 2}};
```

where first two digits represent $(i,\ j)$ grid, and the third digit is a value stored at that location.

In[3]:=
```
SudokuInteger[G28] // Timing
```

Out[3]=
```
{9.79686, {-2 + x[81], -9 + x[80], -6 + x[79], -1 + x[78],
  -4 + x[77], -8 + x[76], -5 + x[75], -3 + x[74], -7 + x[73],
  -1 + x[72], -7 + x[71], -8 + x[70], -9 + x[69], -5 + x[68], -3 + x[67],
  -2 + x[66], -4 + x[65], -6 + x[64], -5 + x[63], -4 + x[62], -3 + x[61],
  -7 + x[60], -2 + x[59], -6 + x[58], -9 + x[57], -1 + x[56], -8 + x[55],
  -7 + x[54], -8 + x[53], -9 + x[52], -2 + x[51], -6 + x[50], -1 + x[49],
  -4 + x[48], -5 + x[47], -3 + x[46], -4 + x[45], -3 + x[44], -1 + x[43],
  -5 + x[42], -8 + x[41], -9 + x[40], -7 + x[39], -6 + x[38], -2 + x[37],
  -6 + x[36], -5 + x[35], -2 + x[34], -3 + x[33], -7 + x[32], -4 + x[31],
  -8 + x[30], -9 + x[29], -1 + x[28], -9 + x[27], -2 + x[26], -5 + x[25],
  -6 + x[24], -1 + x[23], -7 + x[22], -3 + x[21], -8 + x[20], -4 + x[19],
  -3 + x[18], -6 + x[17], -4 + x[16], -8 + x[15], -9 + x[14], -2 + x[13],
  -1 + x[12], -7 + x[11], -5 + x[10], -8 + x[9], -1 + x[8], -7 + x[7],
  -4 + x[6], -3 + x[5], -5 + x[4], -6 + x[3], -2 + x[2], -9 + x[1]}}
```

Boolean System

- For each grid $(i,\ j)$, set 9 variables $x_{ij1},..., x_{ij9}$ so there are 729 variables in total.
- For each x_{ijk} add a contrain $x_{ijk}\,(x_{ijk} - 1) = 0$
- For each grid $(i,\ j)$, add an equation $\sum_{k=1}^{9} x_{ijk} = 1$
- For two x_{ijk} in each row, column or block, add an equation $\sum_{s=1}^{9} x_{ijs}\, x_{kls} = 0$

SAT Solver

Another way to solve Sudoku is to use the SAT solver. Here is a setup:

- For each grid $(i,\ j)$, set 9 variables $x_{ij1},..., x_{ij9}$ so there are 729 variables in total.
- For each grid $(i,\ j)$, add $\bigvee_k x_{ijk}$, one of nine digis
- For two x_{ijk} in each row, add $\neg\left(x_{ijs} \wedge x_{iks}\right)$, no same digits, 36 clauses

- For two x_{ijk} in each column, add $\neg\left(x_{jis} \wedge x_{kis}\right)$
- For two x_{ijk} in each block, add $\neg\left(x_{ijs} \wedge x_{kls}\right)$

This gives us a total of 11745 clauses:

81 definedness clauses of length 9,

$81 \cdot 36 = 2916$ validity clauses, $x_{ijk} \neq x_{ijl}$

$3 \cdot 81 \cdot 36 = 8748$ uniqueness clauses of length 2 (constraints),

■ **Demo (SAT solver)**

In[4]:=

```
SudokuSAT9[given_List] := Module[
  {i, j, k, l, t, rows, cols, m = 3, n = 9, cur, ans, ansmatrix},
  SAT = True;
  ansmatrix = IdentityMatrix[n];

  (* Given numbers *)
  For[i = 1, i ≤ Length[given], i++,
   SAT = SAT && w[ given[[i, 1]]][given[[i, 2]]][given[[i, 3]]] ];

  (* Exactly one digit per grid *)
  For[i = 1, i ≤ n, i++,
   For[j = 1, j ≤ n, j++,
    SAT =
     SAT && (w[i][j][1] || w[i][j][2] ||w[i][j][3] || w[i][j][4] ||
        w[i][j][5] || w[i][j][6] ||w[i][j][7] || w[i][j][8] ||
        w[i][j][9]);
    For[k = 1, k ≤ n, k++,
     For[l = k + 1, l ≤ n, l++,
      SAT = SAT && (!w[i][j][k] || !w[i][j][l]);
     ]
    ]
   ]
  ];

  (* Constraints with a block *)
  For[i = 1, i ≤ m, i++,
   For[j = 1, j ≤ m, j++,
    rows = {(i - 1) m + 1, (i - 1) m + 1, (i - 1) m + 1, (i - 1) m + 2,
      (i - 1) m + 2, (i - 1) m + 2, (i - 1) m + 3, (i - 1) m + 3, (i - 1) m + 3};
```

```
  cols = {(j - 1) m + 1, (j - 1) m + 2, (j - 1) m + 3, (j - 1) m + 1,
    (j - 1) m + 2, (j - 1) m + 3, (j - 1) m + 1, (j - 1) m + 2, (j - 1) m + 3};
  For[k = 1, k ≤ n, k++,
   For[l = k + 1, l ≤ n, l++,
    For[t = 1, t ≤ n, t++,
     SAT = SAT && (! w[rows[[k]]][cols[[k]]][t] ||
           ! w[rows[[l]]][cols[[l]]][t]);
    ]
   ]
  ]
 ]
];

(* Constrains within a row *)
For[i = 1, i ≤ n, i++,
 For[j = 1, j ≤ n, j++,
  For[k = j + 1, k ≤ n, k++,
   For[l = 1, l ≤ n, l++,
    SAT = SAT && (! w[i][j][l] || ! w[i][k][l]);
   ]
  ]
 ]
];

(* Constrains within a column *)
For[i = 1, i ≤ n, i++,
 For[j = 1, j ≤ n, j++,
  For[k = j + 1, k ≤ n, k++,
   For[l = 1, l ≤ n, l++,
    SAT = SAT && (! w[j][i][l] || ! w[k][i][l]);
   ]
  ]
 ]
];

(* Solve the Sudoku *)
ans = SatisfiabilityInstances[SAT,
  Flatten[Table[w[i][j][k], {i, 1, n}, {j, 1, n}, {k, 1, n}]]];
cur = 0;
For[i = 1, i ≤ n, i++,
 For[j = 1, j ≤ n, j++,
   For[k = 1, k ≤ n, k++,
     cur++;
```

```
        If[TrueQ[ans[[1, cur]]], ansmatrix[[i, j]] = k];
       ];
     ];
   ];
   ansmatrix // MatrixForm
 ]
```

In[5]:=
```
SudokuSAT9[G28] // Timing
```

Out[5]=
$$\left\{3.91563,\ \begin{pmatrix} 9 & 2 & 6 & 5 & 3 & 4 & 7 & 1 & 8 \\ 5 & 7 & 1 & 2 & 9 & 8 & 4 & 6 & 3 \\ 4 & 8 & 3 & 7 & 1 & 6 & 5 & 2 & 9 \\ 1 & 9 & 8 & 4 & 7 & 3 & 2 & 5 & 6 \\ 2 & 6 & 7 & 9 & 8 & 5 & 1 & 3 & 4 \\ 3 & 5 & 4 & 1 & 6 & 2 & 9 & 8 & 7 \\ 8 & 1 & 9 & 6 & 2 & 7 & 3 & 4 & 5 \\ 6 & 4 & 2 & 3 & 5 & 9 & 8 & 7 & 1 \\ 7 & 3 & 5 & 8 & 4 & 1 & 6 & 9 & 2 \end{pmatrix}\right\}$$

Using the SAT solver, we can also count the number of solutions:

In[6]:=
```
SatisfiabilityCount[SAT,
  Flatten[Table[w[i][j][k], {i, 1, 9}, {j, 1, 9}, {k, 1, 9}]]] //
 Timing
```

Out[6]=
```
{200.758, 1}
```

In[7]:=
```
Extreme1 = {{1, 6, 3}, {1, 7, 7}, {1, 8, 5}, {2, 1, 6}, {2, 3, 2},
   {2, 4, 9}, {4, 2, 9}, {5, 2, 5}, {5, 3, 4}, {5, 8, 8}, {6, 4, 6},
   {6, 9, 1}, {7, 5, 5}, {7, 6, 8}, {8, 1, 7}, {8, 7, 6}, {8, 9, 2}};
```

In[8]:=
```
SudokuSAT9[Extreme1] // Timing
```

Out[8]=
$$\left\{3.27602,\ \begin{pmatrix} 9 & 1 & 8 & 2 & 6 & 3 & 7 & 5 & 4 \\ 6 & 7 & 2 & 9 & 4 & 5 & 3 & 1 & 8 \\ 5 & 4 & 3 & 8 & 1 & 7 & 2 & 6 & 9 \\ 3 & 9 & 6 & 5 & 8 & 1 & 4 & 2 & 7 \\ 1 & 5 & 4 & 3 & 7 & 2 & 9 & 8 & 6 \\ 8 & 2 & 7 & 6 & 9 & 4 & 5 & 3 & 1 \\ 2 & 6 & 9 & 4 & 5 & 8 & 1 & 7 & 3 \\ 7 & 8 & 5 & 1 & 3 & 9 & 6 & 4 & 2 \\ 4 & 3 & 1 & 7 & 2 & 6 & 8 & 9 & 5 \end{pmatrix}\right\}$$

In[76]:=
```
SatisfiabilityCount[SAT,
 Flatten[Table[w[i][j][k], {i, 1, 9}, {j, 1, 9}, {k, 1, 9}]]]
```

Out[76]=
```
$Aborted
```