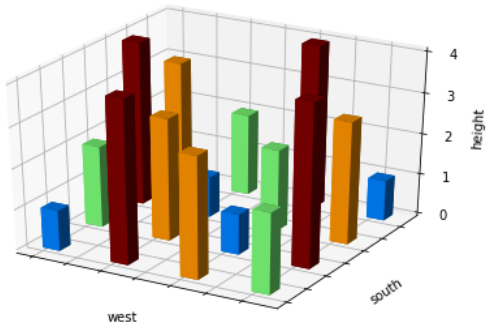
The Skyscrapers Puzzle

Integer Programming vs. Constraint Programming

Julius Barth

June 2, 2021



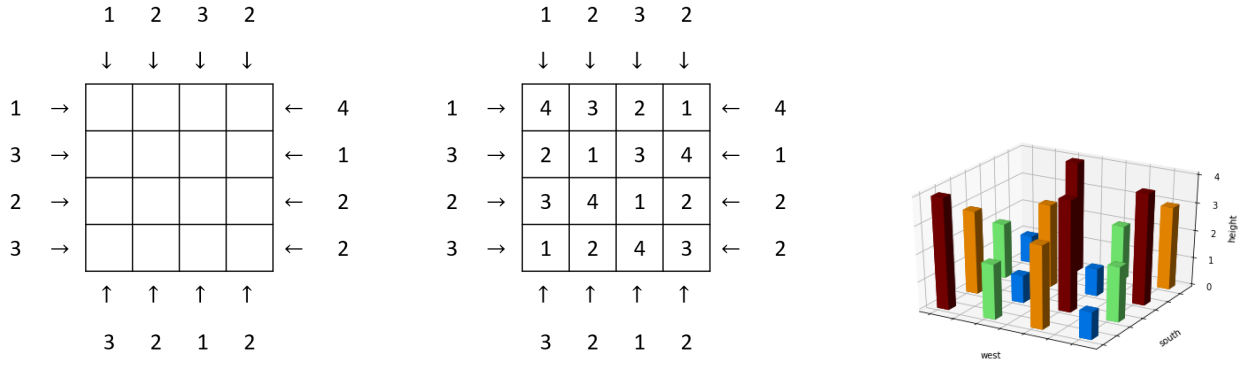
Abstract

An Integer Programming formulation for the skyscrapers puzzle (Chlond 2013), is presented, strengthened with valid inequalities and compared with a Constraint Programming formulation. A brief numerical study shows that while the valid inequalities greatly improve the performance of the basic Integer Programming formulation, the Constraint Programming formulation is still significantly more powerful than even the strengthened Integer Programming formulation.

Contents

1 Problem Definition

The skyscrapers puzzle is combinatorial puzzle and can be seen as a special case of a Latin Square. The objective of a $n \times n$ Latin Square is to fill each cell of the square with one of n elements, e.g. an integer from 1 to n, so that each row and each column may contain any given element at most once. For the special case of a Skyscraper Square, we can interpret the assigned value of a cell as the height of a skyscraper placed in that cell. Additionally, there exist requirements on the number of visible skyscrapers for each row and columns from all directions (North, South, East West). A skyscraper is deemed visible, if it is not hidden behind a taller skyscraper. Some additional clues may be given in the form of pre-filled cell values, i.e. skyscraper heights. Figure 1 presents an example of a skyscraper puzzle, the corresponding solution in form of a Skyscraper Square and a 3D visualization of the result.



(a) Example puzzle setup (b) Example puzzle solution (c) Example puzzle 3D

Figure 1: Skyscrapers puzzle example.

2 Methodology

2.1 Integer Programming vs. Constraint Programming

Integer Programming (IP) deals with problems of the form $max\{cx\, s.t.\, Ax \leq b, x \in \mathbb{Z}_+^n\}$, where both the objective function and the set of constraints are linear in terms of the decision variables which may only take on discrete integer values. While IP is generally based on optimality or finding near-optimal solutions, Constraint Programming (CP) is based on finding a feasible solution to a problem stated with arbitrary constraints. For CP, constraints need not be linear and are generally more flexible. For example, CP allows us to natively express logical implications, e.g. $A \rightarrow B$, which

require further treatment in IP formulations, i.e. auxiliary variables and constraints. Additionally, CP allows for the use of specialized relationships such as $allDifferent()$ or $noOverlap$ constraints. Thus, it can be said that the theoretical foundations of IP lie in algebra while the theoretical foundations of CP lie in logical inferences. IP models are typically solved by means of problem relaxations, e.g. in the Branch-and-Bound algorithm, or by using cutting planes to tighten the formulation. CP models are solved by fixing certain decision variable values and subsequently reducing the remaining decision variables' domains by means of logical inferences, i.e. constraint propagation.

2.2 Basic Integer Programming Formulation

Let n denote the size (height and width) of the square to be filled. Thus, let $N = \{1, \ldots, n\}$ be the set of rows, columns and feasible digits which can be used to fill the square. Each cell has a unique reference label (i, j) defined by its row i and column j. There may be certain clues for pre-filled cells which are denoted by g_{ij}. The set of clue cells is denoted by C. Let $D = \{North, South, West, East\}$ denote the set of directions from which the square is viewed. The visibility requirements/clues are then denoted by the parameters $a_i^{North}, a_i^{South}, a_j^{West}$ and a_j^{East}. These parameters could be under specified, i.e. some values may be missing. We denote the sets of indices with for which a visibility requirement/clue exists for direction d by C^d.

The height of the skyscraper in cell (i, j) is encoded by the binary decision variables x_{ij}^k, where a value of 1 indicates the skyscraper in cell (i, j) is k units tall, i.e. digit k is placed in cell (i, j). For notational simplicity it is sometimes convenient to use an auxiliary integer-valued decision variable $z_{ij} = \sum_{k \in N} k x_{ij}^k$ to denote the height of the skyscraper in cell (i, j). Interestingly, using this more explicit description of cell values can sometimes help the Solver to completely solve the model in the pre-solve phase by using the various reductions, implications, and propagations. Let the binary decision variable y_{ij}^d indicate whether the skyscraper in cell (i, j) is visible from direction d. Finally, let the binary decision variable v_{ijkl} denote whether the skyscraper in cell (i, j) is shorter than the skyscraper in cell (k, l). Using the above notation, the skyscrapers puzzle can be stated as a constraint satisfaction problem using an IP formulation (see Chlond 2013) as stated by constraints (1) - (26).

$$\sum_{k \in N} x_{ij}^k = 1 \qquad \forall \quad i \in N, j \in N \tag{1}$$

$$\sum_{k\in k} k x_{ij}^k = z_{ij} \quad \forall \quad i \in N, j \in N \tag{2}$$

$$x_{ij}^k = 1 \quad \forall \quad (i,j) \in C : g_{ij} = k \tag{3}$$

$$z_{ij} = g_{ij} \quad \forall \quad (i,j) \in C \tag{4}$$

$$\sum_{j\in N} x_{ij}^k = 1 \quad \forall \quad i \in N, k \in N \tag{5}$$

$$\sum_{i\in N} x_{ij}^k = 1 \quad \forall \quad j \in N, k \in N \tag{6}$$

$$\sum_{i\in N} y_{ij}^d = a_j^d \quad \forall \quad d \in \{North, South\}, j \in C^d \tag{7}$$

$$\sum_{j\in N} y_{ij}^d = a_i^d \quad \forall \quad d \in \{West, East\}, i \in C^d \tag{8}$$

$$y_{1j}^{North} = 1 \quad \forall \quad j \in N \tag{9}$$

$$y_{nj}^{South} = 1 \quad \forall \quad j \in N \tag{10}$$

$$y_{i1}^{West} = 1 \quad \forall \quad i \in N \tag{11}$$

$$y_{in}^{East} = 1 \quad \forall \quad i \in N \tag{12}$$

$$z_{kl} \leq z_{ij} + M v_{ijkl} \quad \forall \quad i \in N, j \in N, k \in N, l \in N \tag{13}$$

$$z_{kl} \geq z_{ij} - M(1 - v_{ijkl}) \quad \forall \quad i \in N, j \in N, k \in N, l \in N \tag{14}$$

$$\sum_{k=1}^{i-1} v_{ijkj} \geq 1 - M y_{ij}^{North} \quad \forall \quad j \in N, i \in \{2, \ldots, n\} \tag{15}$$

$$\sum_{k=1}^{i-1} v_{ijkj} \leq M(1 - y_{ij}^{North}) \quad \forall \quad j \in N, i \in \{2, \ldots, n\} \tag{16}$$

$$\sum_{k=i+1}^{n} v_{ijkj} \geq 1 - M y_{ij}^{South} \quad \forall \quad j \in N, i \in \{1, \ldots, n-1\} \tag{17}$$

$$\sum_{k=i+1}^{n} v_{ijkj} \leq M(1 - y_{ij}^{South}) \quad \forall \quad j \in N, i \in \{1, \ldots, n-1\} \tag{18}$$

$$\sum_{l=1}^{j-1} v_{ijil} \geq 1 - M y_{ij}^{West} \quad \forall \quad i \in N, j \in \{2, \ldots, n\} \tag{19}$$

$$\sum_{l=1}^{j-1} v_{ijil} \leq M(1 - y_{ij}^{West}) \quad \forall \quad i \in N, j \in \{2, \ldots, n\} \tag{20}$$

$$\sum_{l=j+1}^{n} v_{ijil} \geq 1 - M y_{ij}^{East} \quad \forall \quad i \in N, j \in \{1, \ldots, n-1\} \tag{21}$$

$$\sum_{l=j+1}^{n} v_{ijil} \leq M(1 - y_{ij}^{East}) \quad \forall \quad i \in N, j \in \{1, \ldots, n-1\} \tag{22}$$

$$x_{ij}^{k} \in \{0,1\} \quad \forall \quad i \in N, j \in N, k \in N \tag{23}$$

$$z_{ij} \in \{1, \ldots, n\} \quad \forall \quad i \in N, j \in N \tag{24}$$

$$y_{ij}^{d} \in \{0,1\} \quad \forall \quad i \in N, j \in N, d \in D \tag{25}$$

$$v_{ijkl} \in \{0,1\} \quad \forall \quad i \in N, j \in N, k \in N, l \in N \tag{26}$$

Constraints (1) and (2) ensure that each cell is filled with only one value and define the auxiliary decision variables z_{ij}. Either constraint (3) or (4) can be imposed to utilize cell clues if $|C| \neq \emptyset$. Constraints (5) and (6) define the Latin Square requirements, i.e. any element occurs at most once in each row and each column. Constraints (7) and (8) ensure that the visibility requirements are met for all rows, columns and directions. Constraints (9) - (12) define the visibility of the cells in the first row/column for any given viewing direction. Obviously the first skyscraper is always visible regardless of its height. Constraints (13) and (14) define the relative ordering of skyscraper heights which are needed to compute the visibility variables as explained below. Constraint (13) forces $s_{ijkl} = 1$ whenever the skyscraper in cell (i, j) is shorter than the skyscraper in cell (k, l). Constraint (14) forces $s_{ijkl} = 0$ whenever the skyscraper in cell (i, j) is taller than the skyscraper in cell (k, l). Using the above defined relative ordering of skyscraper heights, the visibility state of skyscrapers which are not in the first row/column of sight are defined by constraints (15) - (22). Since the four pairs of constraints are analogous for the four directions, explanations are provided only for the case of the viewing direction from the North (constraints (15) and (16)). Constraint (15) ensure that if there are no taller skyscrapers in between the viewing position and the skyscraper in cell (i, j) then this skyscraper is recorded as visible. Similarly, constraint (16) ensures that whenever the skyscraper in cell (i, j) is recorded as visible, then there can be no taller skyscrapers in between that cell and the viewing position. Finally, constraints (23) - (26) define the domains of all decision variables.

2.3 Valid Inequalities

The formulation of Chlond (2013) can be strengthened by adding several families of valid inequalities, which arise naturally when one manually solves skyscrapers puzzles. First, consider the case where only a single skyscraper is visible for a given row/column. In this case, it is clear that the

tallest skyscraper must be placed in the closest cell of the row/column under consideration as shown in (27) - (34).

$$x^n_{1j} = 1 \quad \forall \quad j \in N : a^{North}_j = 1 \tag{27}$$

$$z_{1j} = n \quad \forall \quad j \in N : a^{North}_j = 1 \tag{28}$$

$$x^n_{nj} = 1 \quad \forall \quad j \in N : a^{South}_j = 1 \tag{29}$$

$$z_{nj} = n \quad \forall \quad j \in N : a^{South}_j = 1 \tag{30}$$

$$x^n_{i1} = 1 \quad \forall \quad i \in N : a^{West}_i = 1 \tag{31}$$

$$z_{i1} = n \quad \forall \quad i \in N : a^{West}_i = 1 \tag{32}$$

$$x^n_{in} = 1 \quad \forall \quad i \in N : a^{East}_i = 1 \tag{33}$$

$$z_{in} = n \quad \forall \quad i \in N : a^{East}_i = 1 \tag{34}$$

Similarly, in the case where all n skyscrapers are visible for a given row/column, it is clear that the skyscrapers must be arranged in ascending order as shown in (35) - (42).

$$x^i_{ij} = 1 \quad \forall \quad i \in N, j \in N : a^{North}_j = n \tag{35}$$

$$z_{ij} = i \quad \forall \quad i \in N, j \in N : a^{North}_j = n \tag{36}$$

$$x^i_{n-i+1,j} = 1 \quad \forall \quad i \in N, j \in N : a^{South}_j = n \tag{37}$$

$$z_{n-i+1,j} = i \quad \forall \quad i \in N, j \in N : a^{South}_j = n \tag{38}$$

$$x^j_{ij} = 1 \quad \forall \quad j \in N, i \in N : a^{West}_i = n \tag{39}$$

$$z_{ij} = j \quad \forall \quad j \in N, i \in N : a^{West}_i = n \tag{40}$$

$$x^j_{i,n-j+1} = 1 \quad \forall \quad j \in N, i \in N : a^{East}_i = n \tag{41}$$

$$z_{i,n-j+1} = j \quad \forall \quad j \in N, i \in N : a^{East}_i = n \tag{42}$$

Additionally, we can place bounds on the position of the different sized skyscrapers based on the number visible skyscrapers as shown in (43) - (50).

$$\sum_{i=1}^{a^{North}_j - n + k - 1} x^k_{ij} = 0 \quad \forall \quad j \in N : a^{North}_j \geq 2, k \in \{n - a^{North}_j + 2, \ldots, n\} \tag{43}$$

$$z_{ij} \leq k - 1 \quad \forall \quad j \in N : a^{North}_j \geq 2, k \in \{n - a^{North}_j + 2, \ldots, n\}, \; i \in \{1, \ldots, a^{North}_j - n + k - 1\} \tag{44}$$

$$\sum_{i=2n-a_j^{South}-k+2}^{n} x_{ij}^k = 0 \quad \forall \quad j \in N : a_j^{South} \geq 2, k \in \{n - a_j^{South} + 2, \ldots, n\} \tag{45}$$

$$z_{ij} \leq k-1 \quad \forall \quad j \in N : a_j^{South} \geq 2, k \in \{n - a_j^{South} + 2, \ldots, n\}, \; i \in \{2n - a_j^{South} - k + 2, \ldots, n\} \tag{46}$$

$$\sum_{j=1}^{a_i^{West}-n+k-1} x_{ij}^k = 0 \quad \forall \quad i \in N : a_i^{West} \geq 2, k \in \{n - a_i^{West} + 2, \ldots, n\} \tag{47}$$

$$z_{ij} \leq k-1 \quad \forall \quad i \in N : a_i^{West} \geq 2, k \in \{n - a_i^{West} + 2, \ldots, n\}, \; j \in \{1, \ldots, a_i^{West} - n + k - 1\} \tag{48}$$

$$\sum_{j=2n-a_i^{East}-k+2}^{n} x_{ij}^k = 0 \quad \forall \quad i \in N : a_i^{East} \geq 2, k \in \{n - a_i^{East} + 2, \ldots, n\} \tag{49}$$

$$z_{ij} \leq k-1 \quad \forall \quad i \in N : a_i^{East} \geq 2, k \in \{n - a_i^{East} + 2, \ldots, n\} \; j \in \{2n - a_i^{East} - k + 2, \ldots, n\} \tag{50}$$

These inequalities are best explained by means of an example. Consider constraints (43) and (44) for the case of $n = 5$, $j = 1$ and $a_1^{North} = 4$. Then, we know that tallest skyscraper (size 5) can only be placed in positions 4 and 5, but not positions $1, 2$ or 3 of column 1. Thus, we can impose a constraint stating that $x_{11}^5 + x_{21}^5 + x_{31}^5 = 0$. Similarly, for the second tallest skyscraper (size 4) cannot be placed in positions 1 or 2. Thus, we can impose a constraint stating that $x_{11}^4 + x_{21}^4 = 0$. Finally, for the third tallest skyscraper (size 3), we can conclude that it cannot be placed in position 1 and thus $x_{11}^3 = 0$ is valid. Alternatively, we express the three valid inequalities above as $z_{31} \leq 4$, $z_{21} \leq 3$ and $z_{11} \leq 2$. Analogous arguments can be made for the other three directions leading to the formulations stated in (45) - (50).

2.4 Constraint Programming Formulation

For the following CP formulation we will reuse some of the notation introduced in Section 2.2. However, since CP allows us to model constraints in a more flexible manner than IP (e.g. logical constraints with implications), the resulting model can be stated more succinctly as shown in (51) - (69).

$$allDifferent(z_{i1}, \ldots, z_{in}) \quad \forall \quad i \in N \tag{51}$$

$$allDifferent(z_{1j}, \ldots, z_{nj}) \quad \forall \quad j \in N \tag{52}$$

$$z_{ij} = g_{ij} \quad \forall \quad (i,j) \in C \tag{53}$$

$$\sum_{i \in N} y_{ij}^{d} = a_{j}^{d} \quad \forall \quad d \in \{North, South\}, j \in C^{d} \tag{54}$$

$$\sum_{j \in N} y_{ij}^{d} = a_{i}^{d} \quad \forall \quad d \in \{West, East\}, i \in C^{d} \tag{55}$$

$$y_{1j}^{North} = 1 \quad \forall \quad j \in N \tag{56}$$

$$y_{nj}^{South} = 1 \quad \forall \quad j \in N \tag{57}$$

$$y_{i1}^{West} = 1 \quad \forall \quad i \in N \tag{58}$$

$$y_{in}^{East} = 1 \quad \forall \quad i \in N \tag{59}$$

$$z_{ij} \geq \max_{l \in \{1,\ldots,i-1\}} z_{lj} \rightarrow y_{ij}^{North} = 1 \quad \forall \quad i \in \{2,\ldots,n\}, j \in N \tag{60}$$

$$z_{ij} \leq z_{lj} \rightarrow y_{ij}^{North} = 0 \quad \forall \quad i \in \{2,\ldots,n\}, j \in N, \; l \in \{1,\ldots,i-1\} \tag{61}$$

$$z_{ij} \geq \max_{l \in \{i+1,\ldots,n\}} z_{lj} \rightarrow y_{ij}^{South} = 1 \quad \forall \quad i \in \{1,\ldots,n-1\}, j \in N \tag{62}$$

$$z_{ij} \leq z_{lj} \rightarrow y_{ij}^{South} = 0 \quad \forall \quad i \in \{1,\ldots,n-1\}, j \in N, \; l \in \{i+1,\ldots,n\} \tag{63}$$

$$z_{ij} \geq \max_{l \in \{1,\ldots,j-1\}} z_{il} \rightarrow y_{ij}^{West} = 1 \quad \forall \quad j \in \{2,\ldots,n\}, i \in N \tag{64}$$

$$z_{ij} \leq z_{il} \rightarrow y_{ij}^{West} = 0 \quad \forall \quad j \in \{2,\ldots,n\}, i \in N, \; l \in \{1,\ldots,j-1\} \tag{65}$$

$$z_{ij} \geq \max_{l \in \{j+1,\ldots,n\}} z_{il} \rightarrow y_{ij}^{East} = 1 \quad \forall \quad j \in \{1,\ldots,n-1\}, i \in N \tag{66}$$

$$z_{ij} \leq z_{il} \rightarrow y_{ij}^{East} = 0 \quad \forall \quad j \in \{2,\ldots,n\}, i \in N, \; l \in \{j+1,\ldots,n\} \tag{67}$$

$$z_{ij} \in \{1,\ldots,n\} \quad \forall \quad i \in N, j \in N \tag{68}$$

$$y_{ij}^{d} \in \{0,1\} \quad \forall \quad i \in N, j \in N, d \in D \tag{69}$$

Constraints (51) and (52) ensure the Latin Square requirements using logical $allDifferent()$ constraints. Constraints (53) - (59), (68) and (69) are exactly the same as stated in the IP formulation. Finally, constraints (60) - (67) define the visibility status of each cell from each viewing direction by using logical implications. Specifically, if there are no taller skyscrapers in front of the cell under consideration, then the skyscraper is visible; else the skyscraper is not visible.

3 Numerical Study

The basic IP formulation and the strengthened IP formulation with the added valid inequalities were both implemented in Gurobi using the Pyhton API. The standard solution procedure of Gurobi for IPs employs a sophisticated Branch-and-Cut algorithm. Since, we are not concerned with finding an optimal solution, but rather just a feasible solution, a 'Maximum Infeasibility Branching' strategy was chosen for the branching variable selection step, in order to try to quickly generate a feasible solution. The CP model was implemented in OPL and runs on the ILOG CPLEX CP Optimizer engine. First, to compare the performance of the basic IP and the strengthened IP with valid inequalities, tests on 50 random problem instances of size $n \in \{4, 5, 6, 7\}$ were run. All same size instances were of similar difficulty in the sense that they all contained a fully defined set of visibility requirements for all directions, i.e. $C^d = N \quad \forall d \in D$, but no direct clues for cell values, i.e. $C = \emptyset$.

A summary of the solution runtimes for both IP models is presented in Table 1. Even though problems up to size $n = 5$ are solved almost instantaneously by the basic IP, problems of size $n = 6$ and $n = 7$ already require on in the worst case over one minute and eleven minutes respectively to be solved. The strengthened IP model performs significantly better delivering average speed-up factors of well over 100 for both $n = 5$ and $n = 6$. For problems of size $n = 6$ the strengthened IP shows quite considerable performance variation with some instances being solved almost instantaneously while others take almost three minutes to solve.

model	basic IP				strengthened IP			
n	4	5	6	7	4	5	6	7
min	0.01	0.02	0.06	62.81	0.01	0.01	0.03	0.13
max	0.03	0.57	79.72	679.54	0.03	0.68	19.38	175.41
mean	0.01	0.08	21.21	293.81	0.01	0.06	1.71	33.68
median	0.01	0.03	17.75	275.95	0.01	0.02	0.18	18.27

Table 1: Runtime [s] summary statistics: IP vs. strengthened IP.

In order to compare the performance of the strengthened IP model and the CP model, it is reasonable to run both models on problem instances regarded 'hard' on popular puzzle websites, e.g. brainbashers.com. Larger puzzles are considered significantly more difficult to solve than smaller puzzles. Within a given problem size the clue sparsity, i.e. the number of visibility requirements and cell value clues, leads to minor variations in the level of difficulty. The effects of sparsity are generally outweighed by the effect of problem size. For example a sparse puzzle of size $n = 8$ is considered to be much harder than a dense puzzle of size $n = 5$, but still significantly easier

than a sparse puzzle of size $n = 8$. Therefore, we can sample sparse, i.e. 'hard', problems of sizes $n = 8$ from such websites to find a suitable pool of testing instances to compare the strengthened IP formulation with the CP formulation. The ten sampled puzzles instances are the daily 8×8 'hard' puzzles from brainbashers.com from 2020-08-13 to 2020-08-22. These instances have a minimum/mean/maximum visibility clue density of 41%/51%/63% and a minimum/mean/maximum cell clue density of 8%/13%/16%. A summary of the solution runtimes for these test instances is presented in Table 2. It is evident that the CP model vastly outperforms the strengthened IP model, with average runtimes below half a second. In addition to the extremely fast performance on average, the CP model also delivers excellent worst-case performance of well below two seconds for the sample problems. Across all ten test instances, the CP model delivers an average/median speed-up factor of over 1000/500 compared to the strengthened IP.

model	strengthened IP	CP
min	4.49	0.08
max	916.02	1.49
mean	384.79	0.44
median	320.72	0.39

Table 2: Runtime [s] summary statistics: strengthened IP vs. CP.

4 Conclusion

We have explored two solution paradigms for the skyscrapers puzzle, IP and CP. While IP is perhaps the more well know methodology, CP delivers vastly better solution times for this problem. Even after strengthening the basic IP formulation with helpful valid inequalities, CP still outperforms IP. This is likely due to the effects of constraint propagation as well as the smaller number of constraints/variables necessary to model the problem since CP offers more flexibility in terms of model formulation. Based on the excellent performance of the CP model, it would be interesting to explore how well this method fares for the Skyscraper Sudoku problem, a 9×9 skyscraper puzzle which must also satisfy the requirements of Sudoku.

References

[1] Martin J. Chlond. "Puzzle—Latin Square Puzzles". In: *INFORMS Transactions on Education* 13.2 (2013), 126–128. doi:10.1287/ited.1120.0103. eprint: https://doi.org/10.1287/ited.1120.0103. https://doi.org/10.1287/ited.1120.0103.