Matrix Puzzle Formulations

Paul A. Rubin (rubin@msu.edu)

April 26, 2023

Introduction

This document describes two models, one a mixed integer linear program (MIP) and the other a constraint program (CP), to solve a question posed on Mathematics Stack Exchange. The question describes a type of mathematical puzzle. You are given an $N \times N$ matrix and a set of blocks (smaller matrices) collectively containing N^2 numbers. The problem is to reposition the blocks in a nonoverlapping way so that the resulting matrix is symmetric. In what follows, we assume the following.

- The contents of the blocks are integers. This is true of the examples in the original post, but is not necessary for the models. The models will work with real numbers, and conceptually should work with complex numbers or arbitrary symbols (by assigning them unique integer ID numbers and using the ID numbers).
- Blocks can be repositioned but cannot be rotated. (This was not explicitly stated in the original question.)
- Blocks are rectangular matrices of arbitrary dimension. This is actually a generalization of the original question, where all blocks in both examples were either row or column vectors.

Notation

Throughout this document indexing will be 0-based, to be consistent with the use of Java for testing the models. I will arbitrarily assume that the rows and columns of the full matrix are indexed from 0 to $N-1$ with location $(0,0)$ in the upper left. Locations in blocks will similarly be indexed with $(0,0)$ being the upper left corner of each block. Blocks will be positioned by specifying where their upper left cell goes in the full matrix.

The following notation applies to both models:

- K is the number of blocks;
- B^k is the k-th block ($k \in \{0, \ldots, K-1\}$);
- the dimensions of B^k are $m_k \times n_k$;
- $B^k_{i,j}$ is the entry in row i, column j of B^k ($i \in \{0, \ldots, m_k - 1\}, j \in \{0, \ldots, n_k - 1\}$); and
- $A = \bigcup_{k=0}^{K-1} B^k$ is the "alphabet" (the set of all possible values in the final matrix) with minimum $\underline{A}$ and maximum $\overline{A}$.

Constraints

Although constraints will be expressed differently in the MIP and CP models, the underlying logic is the same.

- Every block needs to be placed exactly once.
- Blocks must fit within the puzzle square (cannot extend beyond row/column $N-1$).
- Blocks cannot overlap.
- The entry in each matrix cell comes from the block covering that cell.
- The matrix must be symmetric.

MIP Model

We will need two extra sets of parameters to formulate the MIP model. For every block index $k \in \{0,\ldots,K-1\}$, every cell $(r,c) \in \{0,\ldots,N-1\}^2$ in the matrix and every cell $(i,j) \in \{0,\ldots,N-1\}^2$, we define the following two values:

- $\alpha_{k,r,c,i,j} = 1$ if placing block k at location (r,c) results in it covering cell (i,j), 0 if not; and
- $\beta_{k,r,c,i,j} = \begin{cases} B^k_{i-r,j-c} & \alpha_{k,r,c,i,j} = 1 \\ 0 & \alpha_{k,r,c,i,j} = 0 \end{cases}$.

$\beta_{k,r,c,i,j}$ is the value cell (i,j) inherits from block B^k if the block is placed at location (r,c) (0 if the block does not cover (i,j)).
The MIP model contains the following elements.

Variables

- $x_{k,r,c} \in \{0,1\}$ is 1 if block B^k has its upper left corner at location (r,c), for $k \in \{0,\ldots,K-1\}, r \in \{0,\ldots,N-1\}, c \in \{0,\ldots,N-1\}$.
- $y_{r,c} \in \left\{\underline{A},\ldots,\overline{A}\right\}$ is the entry at location (r,c) of the final matrix, for $k \in \{0,\ldots,K-1\}, r \in \{0,\ldots,N-1\}, c \in \{0,\ldots,N-1\}$.

Objective Function

Since we are just looking for a feasible solution, the default objective function (minimize 0) is used.

Constraints

- Every block needs to be placed exactly once.

$$\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} x_{k,r,c} = 1 \quad \forall k \in \{0,\ldots,K-1\}$$

- Blocks must fit within the puzzle square (cannot extend beyond row/column $N-1$).

$$x_{k,r,c} = 0 \quad \forall k,r,c : r + m_k > N \bigvee c + n_k > N$$

- Blocks cannot overlap.

$$\sum_{k=0}^{K-1}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}\alpha_{k,r,c,i,j}x_{k,r,c}=1 \quad \forall(i,j)\in\{0,\ldots,N-1\}^2$$

  (Every cell must be covered by exactly one block.)

- The entry in each matrix cell comes from the block covering that cell.

$$\sum_{k=0}^{K-1}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}\beta_{k,r,c,i,j}x_{k,r,c}=y_{i,j} \quad \forall(i,j)\in\{0,\ldots,N-1\}^2$$

- The matrix must be symmetric.

$$y_{i,j}=y_{j,i} \quad \forall(i,j)\in\{0,\ldots,N-1\}^2 : i<j$$

CP Model

Constraint programming languages tend to be a bit more expressive than MIP models, and in particular most (?) allow an integer variable to be used as an index to an array of constants or variables. The following is the conceptual version of our CP model. As with the MIP model, there is no objective function.

Variables

- $x_k \in \{0,\ldots,N-m_k\}$ is the row in which the upper left corner of block $k\in\{0,\ldots,K-1\}$ is placed.
- $y_k \in \{0,\ldots,N-n_k\}$ is the column in which the upper left corner of block $k\in\{0,\ldots,K-1\}$ is placed.
- $z_{r,c}\in\left\{\underline{A},\ldots,\overline{A}\right\}$ is the entry of the matrix cell at position $(r,c)\in\{0,\ldots,N-1\}^2$.
- $w_{r,c}\in\{0,\ldots,K-1\}$ is the index of the block covering cell $(r,c)\in\{0,\ldots,N-1\}^2$.

Constraints

- Every block needs to be placed exactly once. This is implicit in the use of x_k and y_k as the upper corner for block B^k.
- Blocks must fit within the puzzle square (cannot extend beyond row/column $N-1$). This is handled by the upper bounds on x_k and y_k.
- Blocks cannot overlap. This is implied by $w_{r,c}$ having a unique value for all (r,c) and being well defined.

$$w_{x_k+r,y_k+c}=k \tag{1}$$

- The entry in each matrix cell comes from the block covering that cell.

$$z_{x_k+r,y_k+c}=B^k_{r,c} \tag{2}$$

- The matrix must be symmetric.

$$z_{r,c}=z_{c,r} \quad \forall(r,c)\in\{0,\ldots,N-1\}^2 : r<c$$

Note that constraints (1) and (2) are enforced for all combinations of $k\in\{0,\ldots,K-1\}$, $r\in\{0,\ldots,m_k-1\}$ and $c\in\{0,\ldots,n_k-1\}$, and rely on the ability of a CP model to use variables (x_k and y_k) to index other variables (w and z), something that MIP models cannot do.

The associated Java code demonstrating the use of both models on the examples from the Mathematics Stack Exchange post use CPLEX for the MIP model and CP Optimizer for the CP model. Unfortunately, the use of a variable to index another variable (via the `IloCP.element()` method) is limited to one dimensional arrays of variables. So, in the Java implementation, z and w are flattened to one dimensional arrays indexed by $\left\{0,\ldots,N^2-1\right\}$, and $z_{i,j}$ and $w_{i,j}$ become $z_{N\cdot i+j}$ and $w_{N\cdot i+j}$ respectively.