TECHNICAL UNIVERSITY OF CRETE, ECE

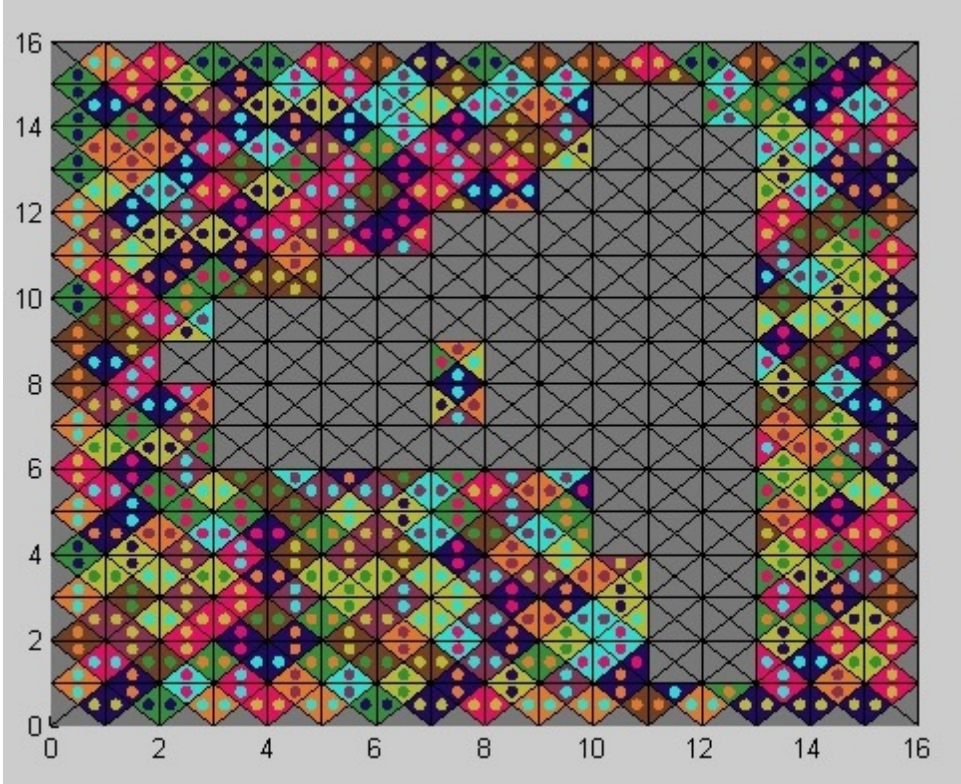
Convex Optimization THL413

Project

Eternity II

Asteri Irini (A.M. 2011030038)

Chrysogelos Periklis (A.M. 2011030064)



Contents

1 Game description

Eternity II is a tesselation puzzle game consisting of 256 square pieces. Each piece is characterized by four colors and associated symbols, one at each edge. Colored symbols may vary per piece or from piece to piece. There are 22 distinct edge-colors plus a solid gray which forms puzzle's frame. Ultimate object of game is tiling the 16×16 gridded board constrained by the requirement of matching the adjacent edjes of neighboring pieces. Game has been designed to be difficult to solve by brute-force computer search and it is considered a NP-hard problem.

Specifically game's rule are

1. All adjacent pieces must match in color (and symbol) at their touching edges.
2. Only solid gray edges must appear in puzzle's boundary.
3. The board must be completely covered.
4. One mandatory piece (numbered 139 in the full game) must have a predetermined cell on the board[1].

Current project deals with modeling and solving scaled down $M \times M$ tesselation problems with the use of convex optimization techniques.

2 Setting up the equations

Every $M \times M$ puzzle comprises M^2 pieces. Modeling the game we choose to represent each piece by a matrix

$$P_i \triangleq \begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \end{bmatrix}^T \in \mathbb{R}^{4 \times L} \quad i = 1 \dots M^2. \tag{1}$$

where each one of four components p_{ij} represents the color of respective edge. Each color corresponds to a L-dimensional standard basis vector $e_l \in \mathbb{R}^L$ or $\mathbf{0}$ if gray. Rows p_{ij} are ordered counterclockwise from east to south edge(figure 1).

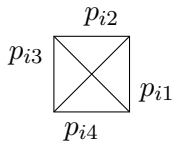


Figure 1: Edge representation by row vectors p_{ij}

[1]This rule is omitted in scaled-down problems.

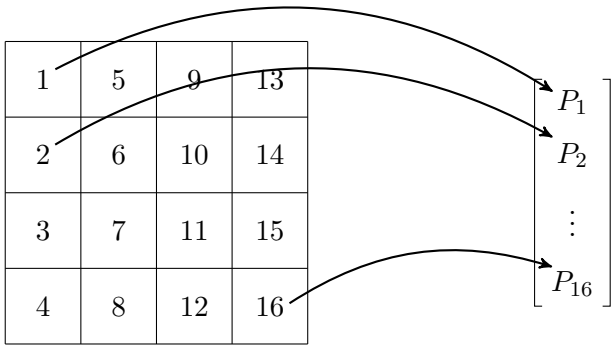


Figure 2: Numbered 4×4 gridded board, every cell corresponds to a fixed P's index

Cells of $M \times M$ gridded board are numbered with $k = 1 \dots M^2$ (figure 2) and pieces' placement in board is represented by the placement of sub-matrices P_i to a matrix $P \in \mathbb{R}^{4M^2 \times L}$. Pieces are initially placed in order of their given index and the initial state is reflected in matrix

$$P \triangleq \begin{bmatrix} P_1 \\ \vdots \\ P_{M^2} \end{bmatrix} \in \mathbb{R}^{4M^2 \times L}. \tag{2}$$

Moving pieces corresponds to permuting pieces P_i on P, while rotating a piece is equivalent to circularly shifting its row indeces. Therefore starting from a random initial state every solution may be described by a permutation Ξ of pieces and a rotation $\Pi_i \in \mathbb{R}^{4\times 4}$ of each individual piece,

$$(\Xi \otimes I_4)\Pi P = (\Xi \otimes I_4) \begin{bmatrix} \Pi_1 P_1 \\ \vdots \\ \Pi_{M^2} P_{M^2} \end{bmatrix} \in \mathbb{R}^{4M^2 \times L} \tag{3}$$

where

$$\Pi_i \in \{\pi_1, \pi_2, \pi_3, \pi_4\} \tag{4a}$$

$$\triangleq \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{270^o}, \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{180^o}, \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{90^o} \right\} \tag{4b}$$

2.1 Defining constraints

According to rule 1 touching edges of adjacent pieces must be identical. In a $M \times M$ board there are $2M(M-1)$ touching edges, therefore $2M(M-1)$ constraints appear. Identical edges are represented by identical standard basis vectors p_{ij}. So requisite constraint can be expressed as the zero difference of each edge couple.

The fact that vectorized board P is fixed dictates that locations of touching edges are specified a priori. So we can simply form the differences between adjacent edges. We express each difference by a vector from the set $\{\Delta_i \in R^{4M^2}, i = 1 \dots 2M(M-1)\}$ and define the matrix

$$\Delta \triangleq \begin{bmatrix} \Delta_1^T \\ \vdots \\ \Delta_{2M(M-1)}^T \end{bmatrix} \in \mathbb{R}^{2M(M-1)\times 4M^2} \tag{5}$$

whose entries $\in \{-1, 0, 1\}$. Vectors Δ_i comprise $4M^2$ entries corresponding to edge's population. Each vector Δ_i includes one couple $-1, 1$ in indeces that represent the i-th edge couple.

So final constraint can be expressed as

$$\Delta(\Xi \otimes I_4)\Pi P = \mathbf{0} \in \mathbb{R}^{2M(M-1)\times L}. \tag{6}$$

Matrix Δ is a *sparse constant* fat matrix.

According to rule 2 only grey edges must appear across puzzles boundary. This fact implies that there are $4M$ edjes, i.e. $4M$ locations in $\mathbb{R}^{4M^2\times L}$ which have value $\mathbf{0}$. The requisite constraints may be imprinted as one equality constraint

$$\beta^T(\Xi \otimes I_4)\Pi P\mathbf{1} = 0 \tag{7}$$

where $\beta \in \mathbb{R}^{4M^2}$ is a sparse constant vector with entries $\in \{0, 1\}$, having nonzero elements only at indexes corresponding to outer frame.

We define

$$\Phi \triangleq (\Xi \otimes I_4)\Pi \in \mathbb{R}^{4M\times 4M}. \tag{8}$$

Due to Ξ's and Π's structures, Φ is also a structured permutation matrix

$$\Phi \triangleq \begin{bmatrix} \phi_{11} & \dots & \phi_{1M} \\ \vdots & \ddots & \vdots \\ \phi_{M1} & \dots & \phi_{MM} \end{bmatrix} \tag{9}$$

where $\Phi_{ij} \in \{0, 1\}$ and $\phi_{ij} \in \{\pi_1, \pi_2, \pi_3, \pi_4, \mathbf{0}\}$.

2.2 Optimality Conditions

Our aim is to find the optimal permutation matrix Φ that solves the puzzle. According to Φ's definition any Φ that corresponds to a valid movement of pieces must have a single 1 in each column and a single 1 in each row. This implies that at optimality every row and column sums up to 1 and also matrix Φ has minimal cardinality. So at optimality Φ is an sparse and non-negative matrix. Blocks ϕ_{ij} are by default circulant, i.e. $\in \pi_1, \pi_2, \pi_3, \pi_4, \mathbf{0}$. This implies that entries along each antidiagonal are equal and corner pairs of 2×2 submatrices on antidiagonal are equal.

We are looking for a minimization problem whose solution, if exists, certifies that puzzle has been solved. Due to above constraints the problem can be described as a cardinality minimization problem,

$$\begin{aligned} \underset{\Phi \in 4M^2 \times 4M^2}{minimize} \; & \parallel vec\Phi \parallel_0 \\ s.t. \quad & \Delta\Phi P = \mathbf{0} \\ & \beta^T \Phi P \mathbf{1} = 0 \\ & \Phi 1 = \mathbf{1} \\ & \Phi^T 1 = \mathbf{1} \\ & (I \otimes R_d)\Phi(I \otimes R_d^T) = (I \otimes S_d)\Phi(I \otimes S_d^T) \\ & (I \otimes R_\phi)\Phi(I \otimes S_\phi^T) = (I \otimes S_\phi)\Phi(I \otimes R_\phi^T) \\ & \Phi \geq \mathbf{0} \end{aligned} \tag{10}$$

where matrices

$$R_d \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}, \quad S_d \triangleq \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 4} \tag{11a}$$

$$R_\phi \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \quad S_\phi \triangleq \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 4} \tag{12a}$$

enforce circulance. R,S constraints are not studied extensively as they become redundant later.

By converting Φ to a vector $vec\Phi$ we create the following equivalent minimization problem with affine and inequality constraints,

$$\begin{aligned} \underset{\Phi \in 4M^2 \times 4M^2}{minimize} \; & \parallel vec\Phi \parallel_0 \\ s.t. \quad & (P^T \otimes \Delta) vec\Phi = \mathbf{0} \\ & (P\mathbf{1} \otimes \beta)^T vec\Phi = 0 \\ & (\mathbf{1}_{4M}^T \otimes I_{4M}) vec\Phi = \mathbf{1_{4M}} \\ & (I_{4M}^T \otimes \mathbf{1}_{4M}) vec\Phi = \mathbf{1_{4M}} \\ & vec\Phi \geq \mathbf{0} \end{aligned} \tag{13}$$

The problem can be abbreviated in

$$\begin{aligned} \underset{\Phi \in 4M^2 \times 4M^2}{minimize} \quad & \| \, vec\Phi \, \|_0 \\ s.t. \quad & Evec\Phi = d \\ & vec\Phi \geq \mathbf{0} \end{aligned} \tag{14}$$

Due to large dimension this problem encounters large memory consumption and numerical errors which prevent its solution. In following sections we try to diminish problem's dimension.

Taking advantage Φ's structure we can lessen variable's dimension. Each block ϕ_{ij} is circulant at optimality which means that it has only four degrees of freedom. Each block can be represented by its first column. So we can use as variable the "sampled" Φ

$$\tilde{\Phi} \triangleq [\Phi(:,1), \Phi(:,5), \dots \Phi(:,4M^2-3)] \in \mathbb{R}^{4M^2 \times M^2} \tag{15}$$

Reconstruction of Φ is implemented through formula

$$\Phi = \sum_{i=1\dots4} (I \otimes \pi_i)(\tilde{\Phi} \otimes e_i^T) \in \mathbb{R}^{4M^2 \times 4M^2} \tag{16}$$

and the respective formula for vectorized variables

$$vec\Phi = \left(\sum_{i=1\dots4} (I \otimes e_i \otimes I \otimes \pi_i) \right) vec\tilde{\Phi} \tag{17a}$$

$$\triangleq Y vec\tilde{\Phi} \in \mathbb{R}^{16M^4}, \qquad Y \in \mathbb{R}^{16M^4 \times 4M^4} \tag{17b}$$

Constraints for circulance of ϕ_{ij} can now be eliminated as circulance is succeded through reconstruction of Φ.
Problem is reformulated to

$$\underset{\tilde{\Phi} \in 4M^2 \times M^2}{minimize} \quad \| \, vec\tilde{\Phi} \, \|_0 \tag{18a}$$

$$s.t. \quad (P^T \otimes \Delta) Y vec\tilde{\Phi} = \mathbf{0} \tag{18b}$$

$$(P\mathbf{1} \otimes \beta)^T Y vec\tilde{\Phi} = 0 \tag{18c}$$

$$(\mathbf{1}^T \otimes I \otimes \mathbf{1}_4^T) vec\tilde{\Phi} = \mathbf{1} \tag{18d}$$

$$(I \otimes \mathbf{1}_{4M}^T) vec\tilde{\Phi} = \mathbf{1} \tag{18e}$$

$$vec\tilde{\Phi} \geq \mathbf{0} \tag{18f}$$

$$\tag{18g}$$

whose optimal value (minimum cardinality) is now M^2.

3 Problem requirements

After formulating the problem, its properties and size must be evaluated. For this purpose we define matrices:

$$E_1 \triangleq (P^T \otimes \Delta)Y \qquad \in\mathbb{R}^{2LM(M-1)\times 4M^4} \tag{19a}$$

$$E_2 \triangleq (P\mathbf{1} \otimes \beta)^T Y \qquad \in\mathbb{R}^{1\times 4M^4} \tag{19b}$$

$$E_3 \triangleq (\mathbf{1}^T \otimes I \otimes \mathbf{1}_4^T) \qquad \in\mathbb{R}^{M^2\times 4M^4} \tag{19c}$$

$$E_4 \triangleq (I \otimes \mathbf{1}_{4M^2}^T) \qquad \in\mathbb{R}^{M^2\times 4M^4} \tag{19d}$$

$$E \triangleq \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} \qquad \in\mathbb{R}^{2LM(M-1)+2M^2+1\times 4M^4} \tag{19e}$$

vector:

$$\mathbf{d} \triangleq \begin{bmatrix} \mathbf{0}_{2LM(M-1)+1} \\ \mathbf{1}_{2M^2} \end{bmatrix} \tag{20}$$

And restate the problem as follows.

$$\begin{aligned} \underset{\mathbf{x}}{minimize} \quad & \| \mathbf{x} \|_0 \\ s.t. \quad & E\mathbf{x} = \mathbf{d} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{21}$$

The size and properties of the constraints' set is examined.

3.1 Size of problem

Matrix E is of size $O(M^6L)$ which seems too big for even representing it in the memory. Minimization's variable is of size $4M^4$, which is also not small.

M	L	Equalities	Size of variable $\mathbf{x}$	Entries of E	Size of E (bytes)
4	4	129	1 024	132 096	1 056 768
6	4	313	5 184	1 622 592	12 980 736
6	22	1 393	5 184	7 221 312	57 770 496
8	22	2 593	16 384	42 483 712	339 869 696
10	22	4 161	40 000	166 440 000	1 331 520 000
16	22	11 073	262 144	2 902 720 512	23 221 764 096

Table 1: Problem's dimensions for various board's sizes and different number of colors assuming E is represented as a normal matrix of doubles (64bit)

As shown in table 1 problem is too big even for small dimensions.

Based on the above sizes, representing the problem in memory is infeasible in its full form, starting from games much smaller than original 16x16.

So some structure of E has to be exposed to be able to proceed.

3.2 Sparsity

Observing entries of E, table 2 is filled.

Matrix	Values	Non Zero Entries (%)		
		(M,L)=(4,4)	(M,L)=(12,22)	(M,L)=(16,22)
E_1	$\{0, \pm 1\}$	2.34	0.05	0.03
E_2	$\{0, 1, 2\}$	60.9	28.2	22.0
E_3	$\{0, 1\}$	6.25	0.69	0.39
E_4	$\{0, 1\}$	6.25	0.69	0.39
E	$\{0, \pm 1, 2\}$	3.77	0.09	0.05

Table 2: Entries and sparsity of every matrix, for some game sizes

All entries of E belong in $\{0, \pm 1\}$ expect from some entries of E_2. Also E_2 has the lowest sparsity. Fortunately, it is small compared to rest of E. Furthermore E_2 is positive, as $\mathbf{x}$ and under the constraint $E_2\mathbf{x} = 0$, it can be shown that for every non zero element of E_2, corresponding entry in $\mathbf{x}$ is zero.

Defining the set $\mathcal{T}$ containing all indexes of E_2 which contain zero entries and redefining:

$$E \triangleq \begin{bmatrix} E_1 \\ E_3 \\ E_4 \end{bmatrix} \tag{22a}$$

$$\mathbf{d} \triangleq \begin{bmatrix} \mathbf{0}_{2LM(M-1)} \\ \mathbf{1}_{2M^2} \end{bmatrix} \tag{22b}$$

problem can be reformulated as:

$$\begin{aligned} \underset{\mathbf{x}}{minimize} \quad & \| \mathbf{x}_{\mathcal{T}} \|_0 \\ s.t. \quad & E\mathbf{x}_{\mathcal{T}} = \mathbf{d} \\ & \mathbf{x} \geq \mathbf{0} \qquad\qquad \mathbf{x}_{\overline{\mathcal{T}}} = \mathbf{0} \end{aligned} \tag{23}$$

Remembering the set $\mathcal{T}$ to be used after solving the problem, permits restating it in the previous form:

$$\begin{aligned} \underset{\mathbf{x}}{minimize} \quad & \| \mathbf{x} \|_0 \\ s.t. \quad & E\mathbf{x} = \mathbf{d} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{24}$$

using the redefined matrices and vertexes $E, \mathbf{x}, \mathbf{d}$. Solution should transformed appropriately after solving the problem.

Updated sparsity table is given in table 3.

Matrix	Values	Non Zero Entries (%)		
		(M,L)=(4,4)	(M,L)=(12,22)	(M,L)=(16,22)
E_1	$\{0, \pm 1\}$	2.34	0.05	0.03
E_3	$\{0, 1\}$	6.25	0.69	0.39
E_4	$\{0, 1\}$	6.25	0.69	0.39
E	$\{0, \pm 1\}$	3.33	0.08	0.05

Table 3: Entries and sparsity of every matrix, for some game sizes

Now, all entries of E belong in $\{0, \pm 1\}$.

Big sparsity shows that sparse matrices can be used to reduce the required memory and $\{0, \pm 1\}$ shows that two matrices can be used instead of one, containing logical values, one representing 1's, and the other to represent -1.

The latter has not been tested enough, as it was not friendly to the method used to solve minimization problems. Nevertheless it was a good step to minimize the required space, as expect from sparsity, group of entries could be encoded in single words. Every elementwise multiplication would be a bitwise and, while every sum (in a group) would be a population count. All this could have provided a good speed up, expect from memory, but it neither matlab friendly nor tools friendly.

M	L	Eq.s	Size $\mathbf{x}$	Entries of E	E (bytes)	Size (sparse) (%)
4	4	128	1 024	131 072	77 832	7,42
6	4	312	5 184	1 617 408	437 768	3,38
6	22	1 392	5 184	7 216 128	437 768	0,76
8	22	2 592	16 384	42 467 328	1 458 184	0,43
10	22	4 160	40 000	166 400 000	3 673 608	0,28
16	22	11 072	262 144	2 902 458 368	25 231 368	0,11

Table 4: Problem's dimensions for various board's sizes and different number of colors assuming E is represented as a sparse matrix of doubles (64bit). Last column represents the final percentage of size used using sparse vs full matrix

Memory consumption for using sparse matrices is presented in table 4. Observable is the fact that as the game grows in dimensions, more memory (proportional) is saved. Also number of colors does not seem to influence the resulting size of the sparse matrix, making this representation even more powerful as colors are incremented.

3.3 Numerical errors

Size of data does not only causes problems regarding memory and required time. It also makes numerical errors bigger, as more operations are executed. For example, non zero values of small magnitude may produce a somewhat significant results. This creates failures in many solvers, as it is shown later.

Using sparse matrices does not make a change. Using structures of integers is not applicable, as optimization requires, in general, floating point computations and this is the point where most numerical errors are produced.

4 Cardinality Minimization

The next obstacle faced while theoretically analyzing the problem is the type of the cost function, which is not convex.

Vector's cardinality is a quasiconcave function in $\mathbb{R}^n_+$, as:

$$\| \theta\mathbf{x} + (1-\theta)\mathbf{y} \|_0 \geq \min\{\| \mathbf{x} \|_0, \| \mathbf{y} \|_0\}, \quad \forall x, y \in \mathbb{R}^n_+, \theta \in [0,1] \tag{25}$$

In addition, cost function is not continuous.

So, minimizing it, is not possible without a transformation.

4.1 Constructing a Continuous Problem

This leads to the definition of a new problem, let it be P1, described as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{minimize} \quad & \mathbf{y}^T\mathbf{x} \\ s.t. \quad & E\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \geq \mathbf{0} \\ & \mathbf{y} \leq \mathbf{1} \\ & \mathbf{1}^T\mathbf{y} = n - k \end{aligned} \tag{26}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and k the known minimum cardinality of $\mathbf{x}$.

Constraints $\mathbf{x}, \mathbf{y} \geq \mathbf{0}$ imply an underestimator of P1 and more precisely, if p^*_{P1} the solution of P1, then it is guaranteed that $0 \leq p^*_{P1}$.

Proposition 1. *For every minimizer $\mathbf{x_1}$ of the original problem, there a exist a $\mathbf{y_1}$ such that $(\mathbf{x_1}, \mathbf{y_1})$ is a minimizer of P1.*

Proof. Assume $\mathbf{x_1}$ minimizer of the original problem and $\mathbf{y_1} \triangleq \mathbf{1}_n - \mathbf{x_1}$.

Every solution of the original problem is a boolean vector, so:

$$\mathbf{y_1}_i, \mathbf{x_1}_i \in \{0,1\} \qquad \forall i \tag{27}$$

Implying $\mathbf{1}_n \geq \mathbf{y_1} \geq \mathbf{0}$. Also $\| \mathbf{x_1} \|_0 = \mathbf{1}_n^T \mathbf{x_1} = k$. Giving:

$$\mathbf{1}_n^T \mathbf{y_1} = \mathbf{1}_n^T (\mathbf{1}_n - \mathbf{x_1}) = \mathbf{1}_n^T \mathbf{1}_n - \mathbf{1}_n^T \mathbf{x_1} = n - k \tag{28}$$

Satisfying the last constraint.

In addition,

$$\mathbf{y_1}^T \mathbf{x_1} = (\mathbf{1}_n - \mathbf{x_1})^T \mathbf{x_1} = \mathbf{1}_n^T \mathbf{x_1} - \mathbf{x_1}^T \mathbf{x_1} = n - n = 0 \leq p^*_{P1} \tag{29}$$

So minimum of P1 is 0 and $(\mathbf{x_1}, \mathbf{y_1} = \mathbf{1}_n - \mathbf{x_1})$ is a minimizer of P1. □

Proposition 2. *For every minimizer* $(\mathbf{x_1}, \mathbf{y_1})$ *of P1,* $\mathbf{x_1}$ *is also a minimizer of the original problem.*

Proof. Let $(\mathbf{x_1}, \mathbf{y_1})$ be a minimizer of P1, which implies $\mathbf{y_1}^T \mathbf{x_1} = 0$. Constraints of the original problem are a superset of P1's, so they are satisfied.

If $\| \mathbf{x_1} \|_0 = k$, then $\mathbf{x_1}$ is a solution of the original problem.

$$\mathbf{y_1}^T \mathbf{x_1} = 0 \qquad \Longrightarrow \qquad \mathbf{y_{1i}} \mathbf{x_{1i}} = 0 \qquad \forall i \tag{30}$$

as $\mathbf{y_{1i}} \mathbf{x_{1i}} \geq 0$ by the constraints. Consequence of the above is that:

$$\| \mathbf{y_1} \|_0 + \| \mathbf{x_1} \|_0 \leq n \qquad \Longleftrightarrow \qquad \| \mathbf{x_1} \|_0 \leq n - \| \mathbf{y_1} \|_0 \tag{31}$$

Also by $\mathbf{1}_n^T \mathbf{y_1} = n - k$, $\mathbf{y_1} \leq \mathbf{1}_n^T$ it is derived that $\| \mathbf{y_1} \|_0 \geq n - k$. So $\| \mathbf{x_1} \|_0 \leq n - (n - k) = k$. But $\| \mathbf{x_1} \|_0 \geq k$ by constraint $E\mathbf{x_1} = \mathbf{b}$. Leaving only the option $\| \mathbf{x_1} \|_0 = k$ which sets $\mathbf{x_1}$ as a minimizer of the original problem. □

4.2 Equivalence of Problems

By propositions 2 and 1 it is guaranteed that by locating a minimizer of P1, a minimizer of the original problem can be tracked and P1 is feasible because the original problem is.

In other words, solving P1 will give a solution to the original problem.

This solves the problem of having a non continuous cost function, but the problem is still not convex.

4.3 Approximation by Convex Problems

But P1 in many cases is solvable by iterating over two convex problems P2, P3. When P2 is defined by:

$$\begin{aligned} \underset{\mathbf{x}}{minimize} \quad & \mathbf{y}^T \mathbf{x} \\ s.t. \quad & E\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{32}$$

And P3 defined as:

$$\begin{aligned} \underset{\mathbf{y}}{minimize} \quad & \mathbf{y}^T\mathbf{x} \\ s.t. \quad & \mathbf{0} \leq \mathbf{y} \leq \mathbf{1} \\ & \mathbf{1}^T\mathbf{y} = n - k \end{aligned} \tag{33}$$

It is easily observed that each one of the problems is a minimization focusing to one of the two variables, keeping the other constant. More over, both problems are convex.

P2 tries to find the minimum by peeking a "good" point given a $\mathbf{y}$ in each iteration. P3 will still try to minimize the cost by giving greater weights to the elements of $\mathbf{y}$ corresponding to the greater elements of $\mathbf{x}$. This forces the next P2 to consider minimizing small elements of $\mathbf{x}$ and maximizing big ones, getting it closer to a boolean vector, by always trying to keep the constraints valid.

The sequence, while trying to force $\mathbf{x}$ to a boolean vector of the wanted cardinality, makes it possible to consider passing from non boolean vector or even changing goals as to which elements of $\mathbf{x}$ are going to be active. Adding some noise in $\mathbf{y}$ whenever there is no progress, is required for better convergence probability.

In general the method does not guarantee a convergence, but in Eternity II, as the problem has been formulated, it seems that it is common to converge.

Iteration normally ends when $\mathbf{y}^T\mathbf{x}$ becomes sufficiently small.

4.3.1 Starting value of y

Starting value of $\mathbf{y}$ seems to play a great role on the speed and convergence of the method. If $\mathbf{y}$ is close to its optimal value, iteration will be reduced, while being close to it's compliment gives a very slow convergence, if it happens.

An extreme example is when optimal $\mathbf{y}$ is known prior to starting the cardinality minimization. In this case, the minimization is started with it and it is expected to be no more than a single iteration.

But this is an ideal case, not applicable to the real problem. So three approaches have been tested.

Completely random starting y First having a random $\mathbf{y}$ ($\mathbf{1} \geq \mathbf{y} \geq \mathbf{0}$) was tested. This seems to give a convergence with a pretty high probability, but it may be after a big number of iterations.

Also each iteration may be pretty big, especially in P2.

Starting with $\mathbf{y} = \mathbf{0.5}$ So a second approach was to test a starting $\mathbf{y} = \mathbf{0.5}$, the middle of $\mathbf{y}$'s space. This of course does not satisfy its constraints, but there is no need for it as P3 will enforce them after being calculated.

First iterations in this case seem to be wasted. Results produced in first iteration are almost meaningless, but also first iteration is executed pretty fast.

Almost the same results have been produced by setting a starting $\mathbf{y} = \frac{n-k}{2n}\mathbf{1}$, but with a small performance improvement.

Random starting point near $\frac{n-k}{2n}\mathbf{1}$ A hybrid approach followed in the testing process. Starting elements of $\mathbf{y}$, in this experiment, have been picked by a uniform random variable in an interval centered at 0.5 and a width of about 0.2.

This method seemed to produce the best results in general. First iteration was a little slower than with the random $\mathbf{y}$, but they produced much better results.

4.3.2 Meaning of intermediate results

Of big interest are the intermediate values of $\mathbf{x}$, picked after each execution of P2.

At first glance someone would expect them to be meaningless or at least being partially logical. A non boolean vector is returned by P2, without even satisfying problem's constraints fully[2].

Testing has shown that the above assumption is wrong. Thresholding the results can produce a partial solution, representing method's belief. A number of pieces will be placed on the board, with non of them repeated, and all of their existing connections matched. All pieces placed do exist. Of course, all this stands only if a right threshold has been used, but in most cases, 0.5 gave a good result.

4.4 Results

Let's first define the measured sizes.

Intermediate measured cardinality is counted as the cardinality of the thresholded intermediate solution, while maximum constraint error is counted as the number of non satisfied constraints using the thresholded result.

Iteration where ended when at least one of the following was true:

1. Thresholded results has reached the minimum cardinality while satisfying the constraints

2. An iteration limit has been reached

[2]CVX's limitation

Also a complete or partial randomization of $\mathbf{y}$ was happening every time an arbitrary condition of non-progress was satisfied. Usually the condition was not satisfied.

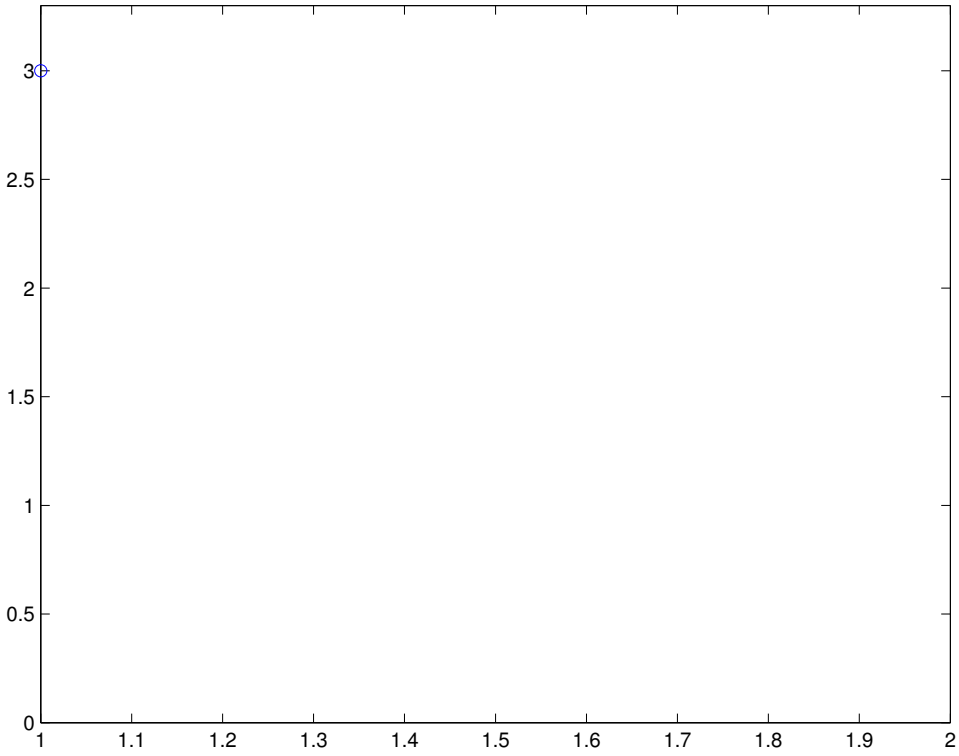


Figure 3: Intermediate cardinality M=2,L=22

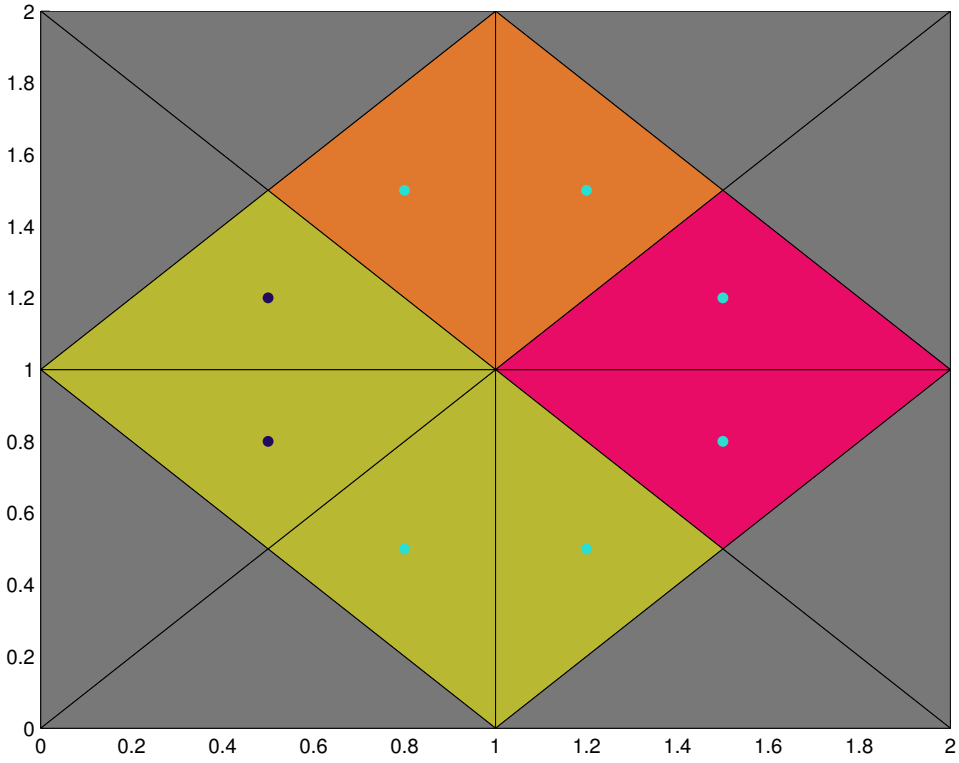


Figure 4: Solved puzzle M=2,L=22

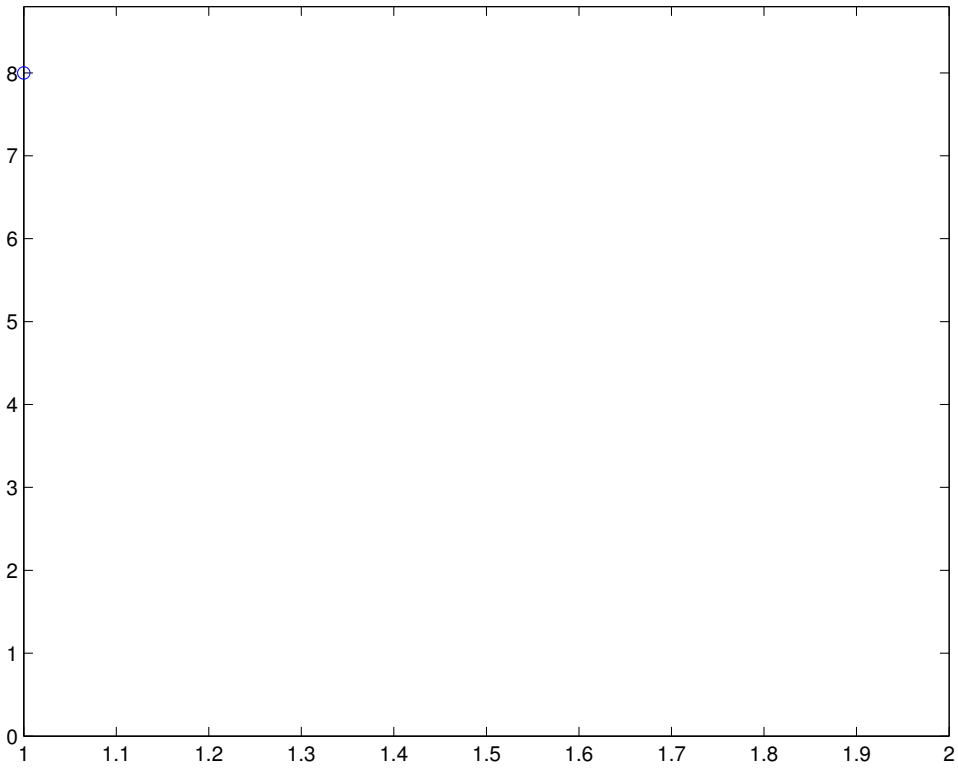


Figure 5: Intermediate cardinality M=3,L=22

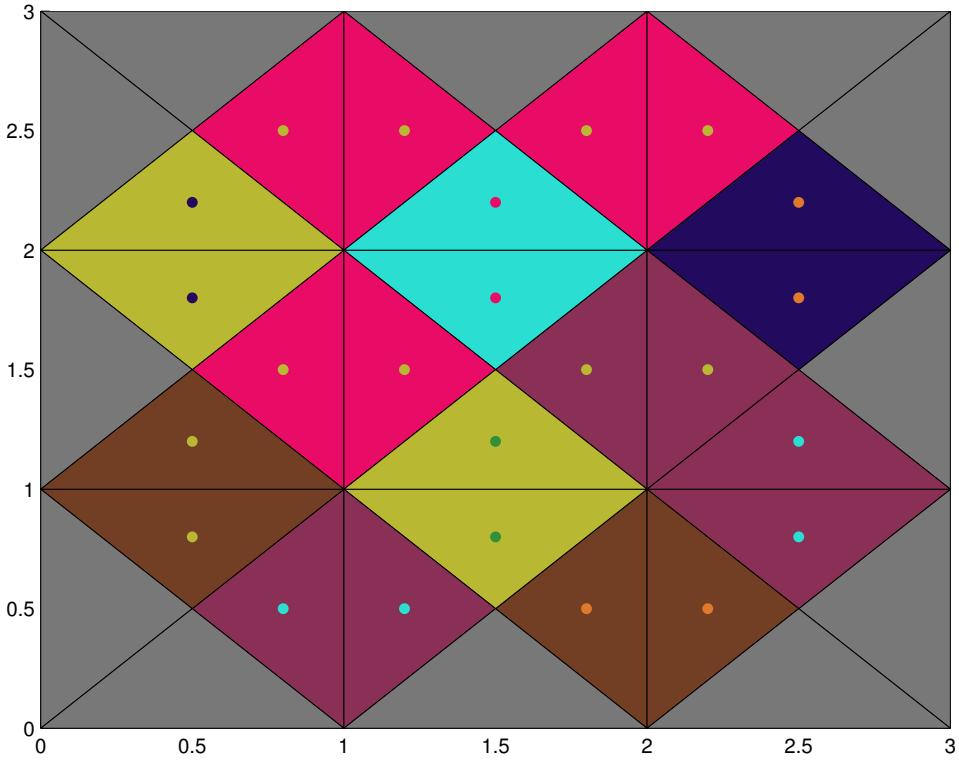


Figure 6: Solved puzzle M=3,L=22

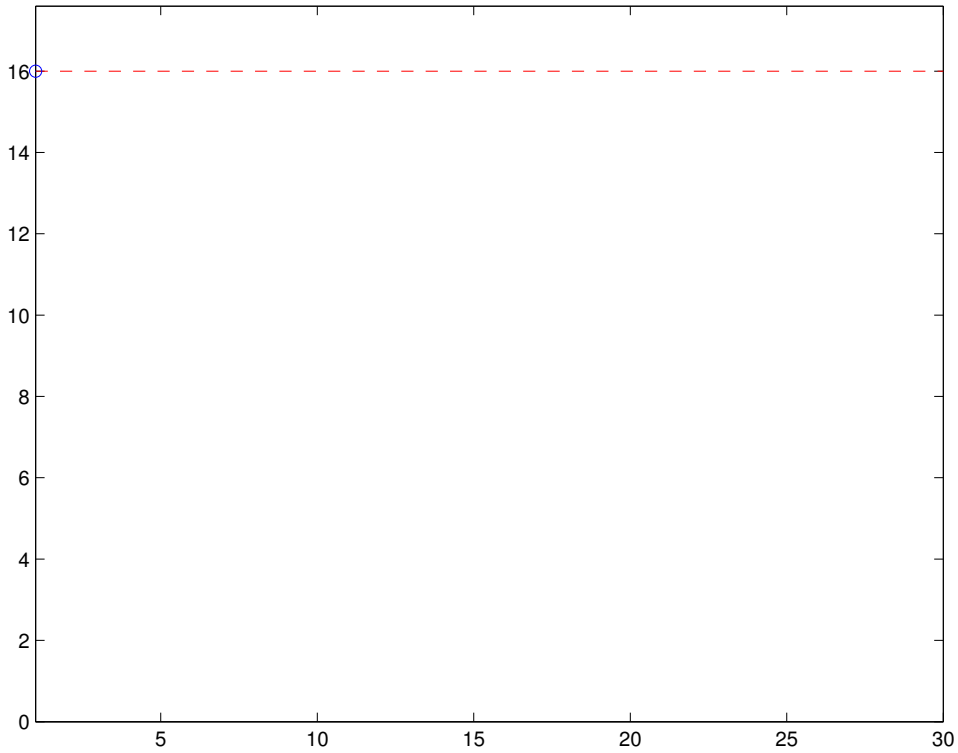


Figure 7: Intermediate cardinality M=4,L=4

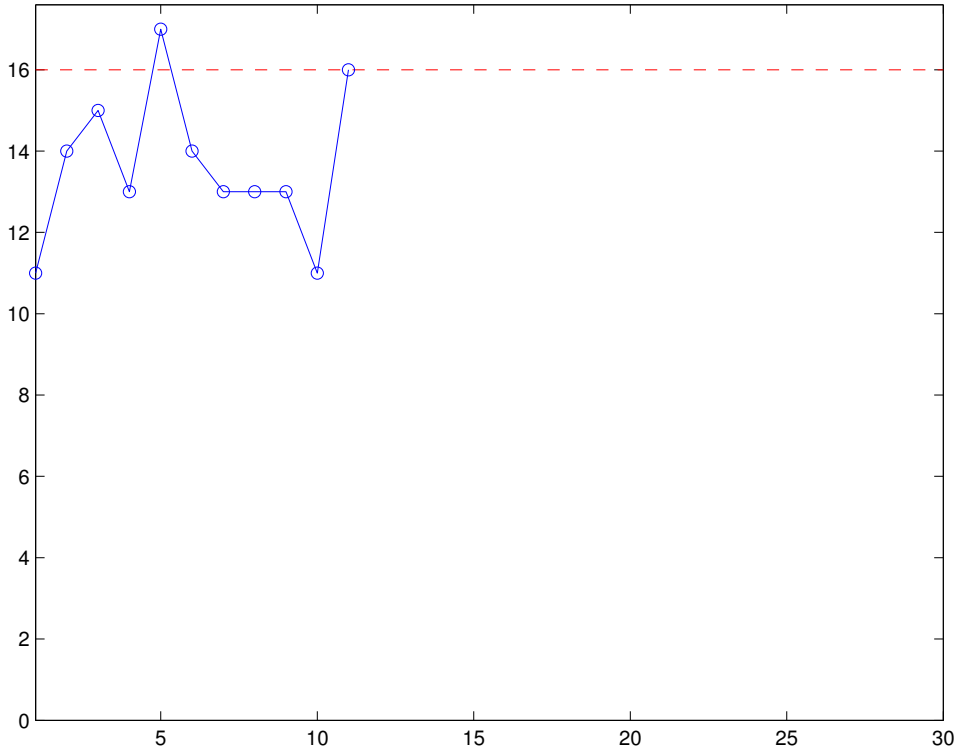


Figure 8: Intermediate cardinality M=4,L=22

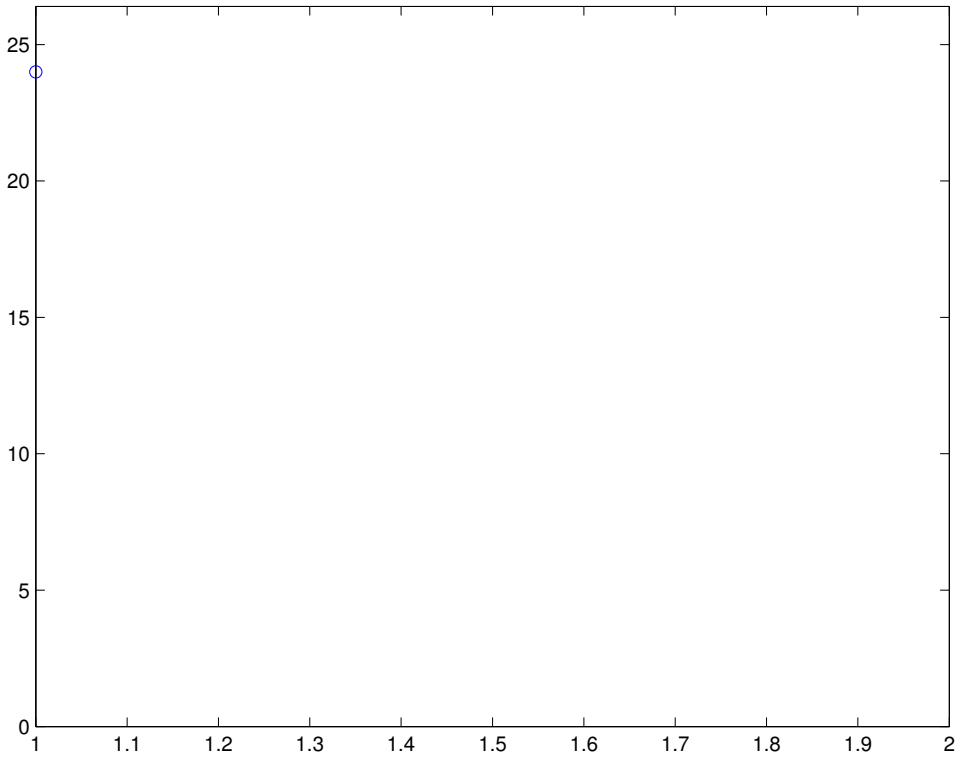


Figure 9: Intermediate cardinality M=5,L=22

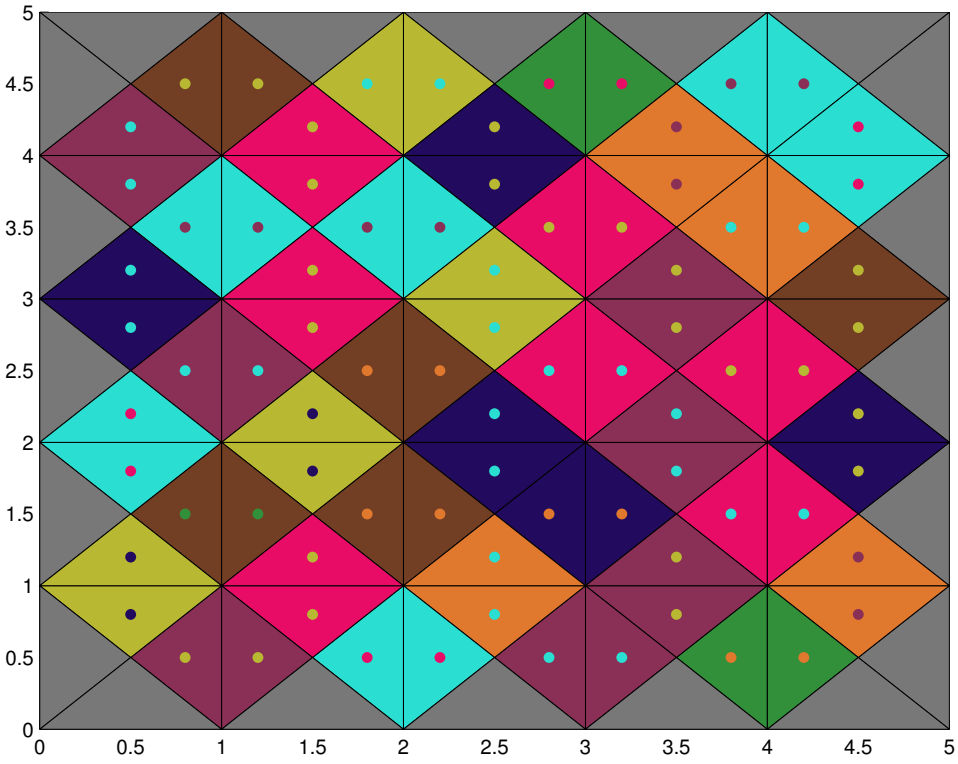


Figure 10: Solved puzzle M=5,L=22

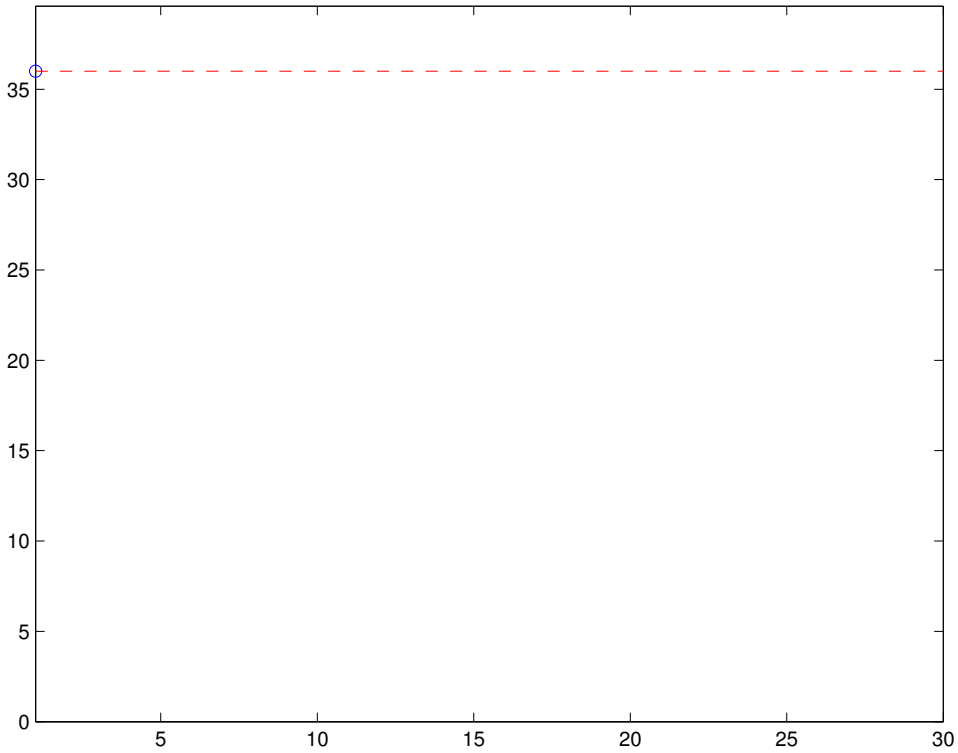


Figure 11: Intermediate cardinality M=6,L=22

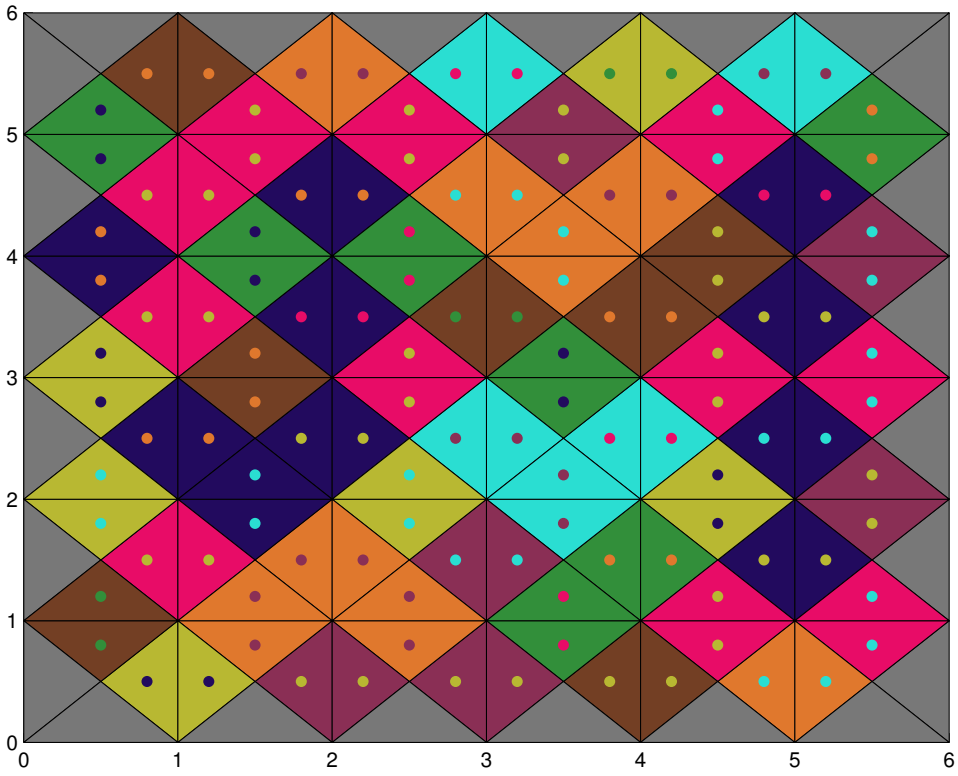


Figure 12: Solved puzzle M=6,L=22

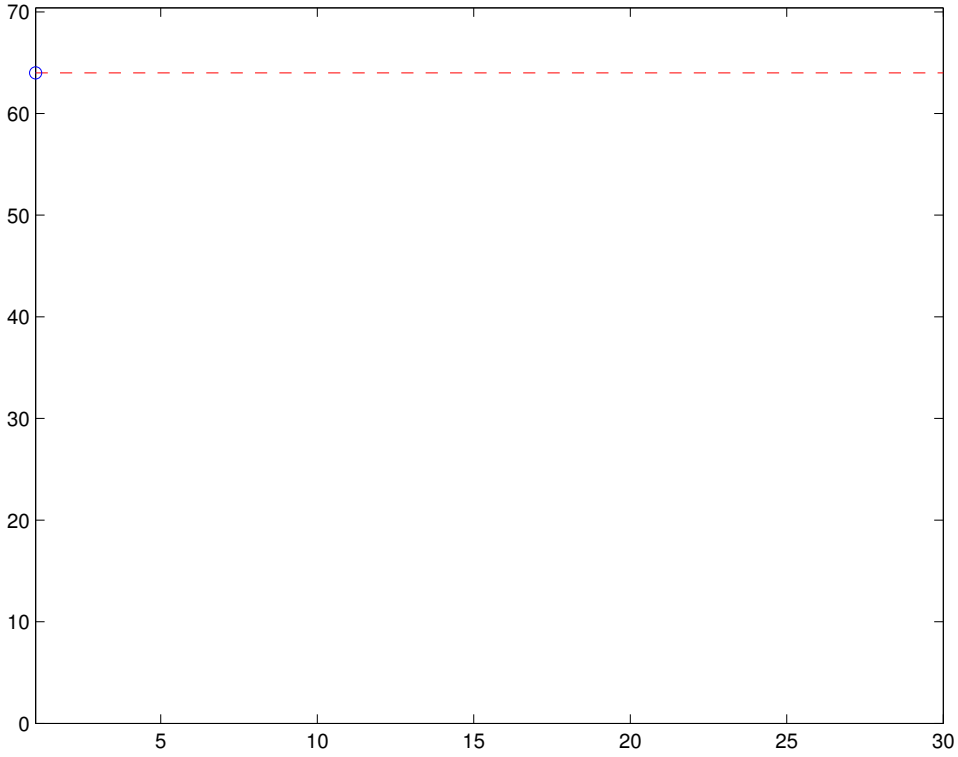


Figure 13: Intermediate cardinality M=8,L=22

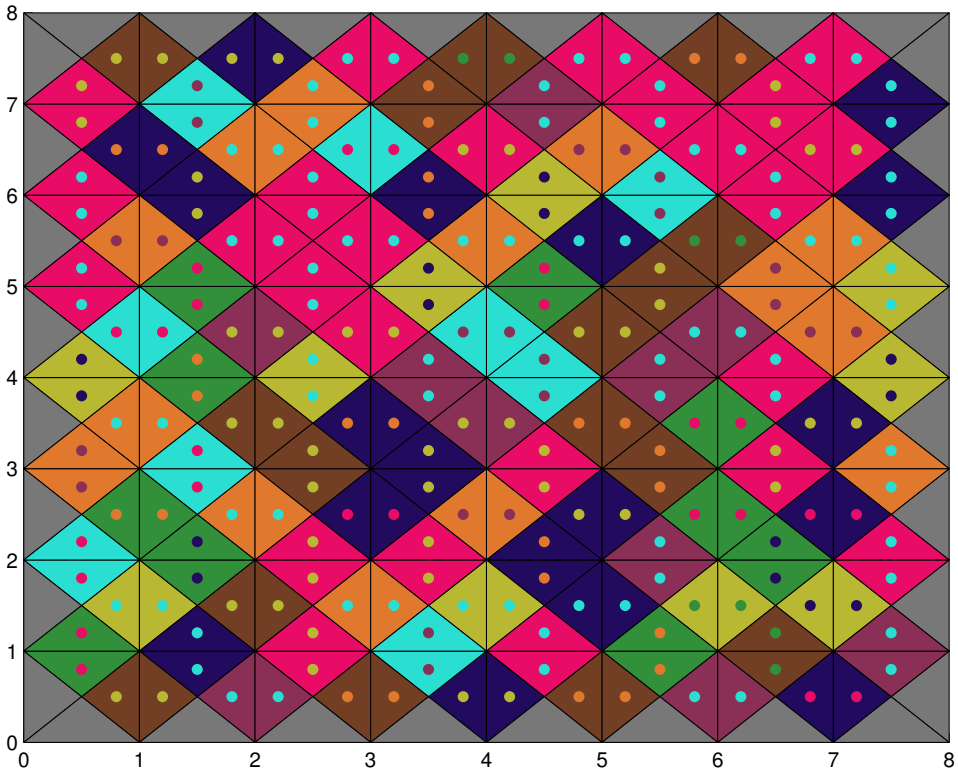


Figure 14: Solved puzzle M=8,L=22

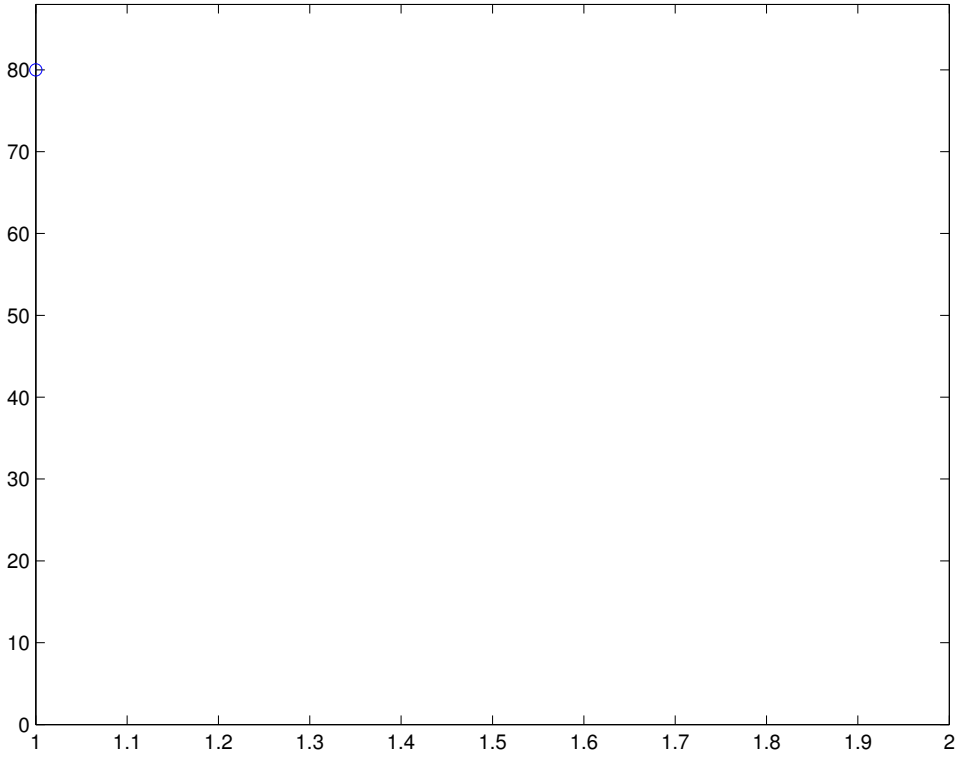


Figure 15: Intermediate cardinality M=9,L=22

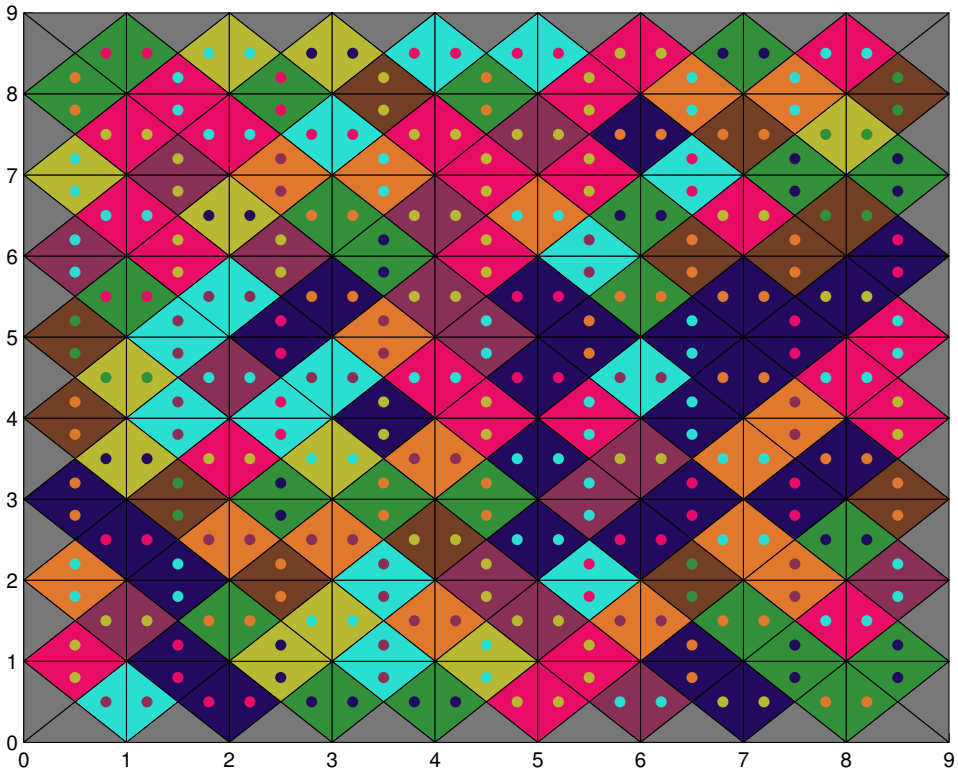


Figure 16: Solved puzzle M=9,L=22

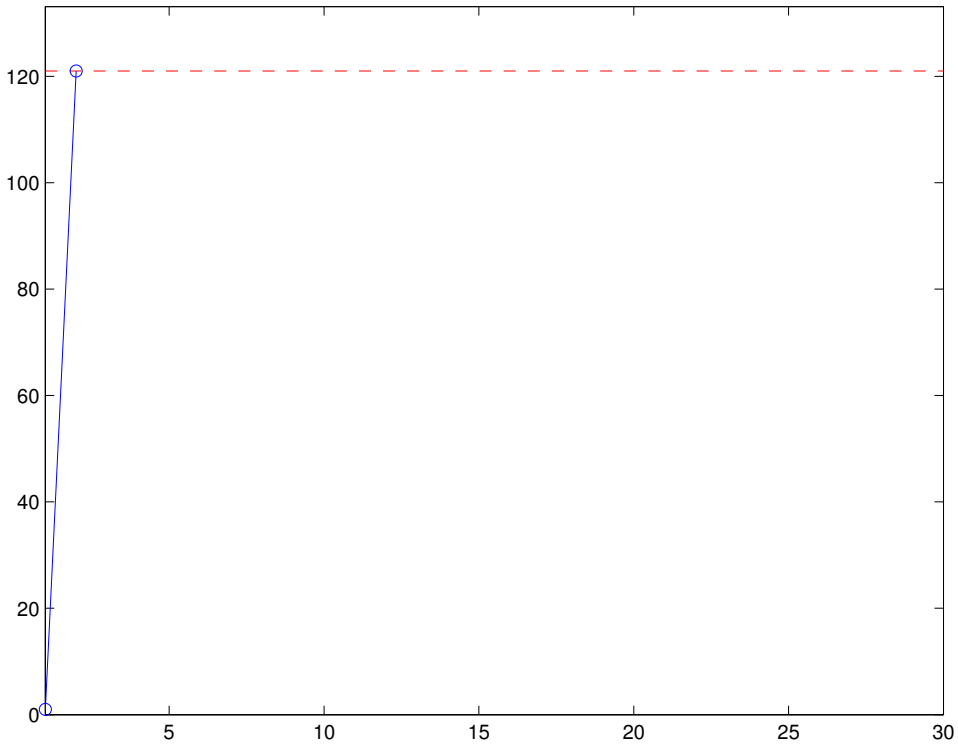


Figure 17: Intermediate cardinality M=11,L=22

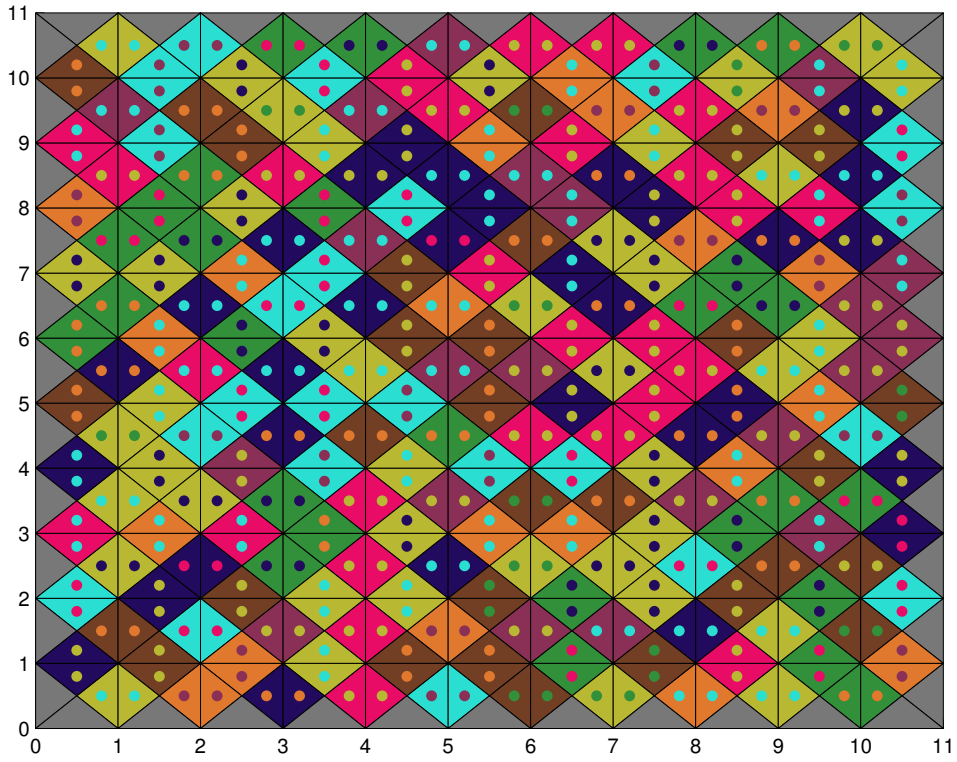


Figure 18: Solved puzzle M=11,L=22

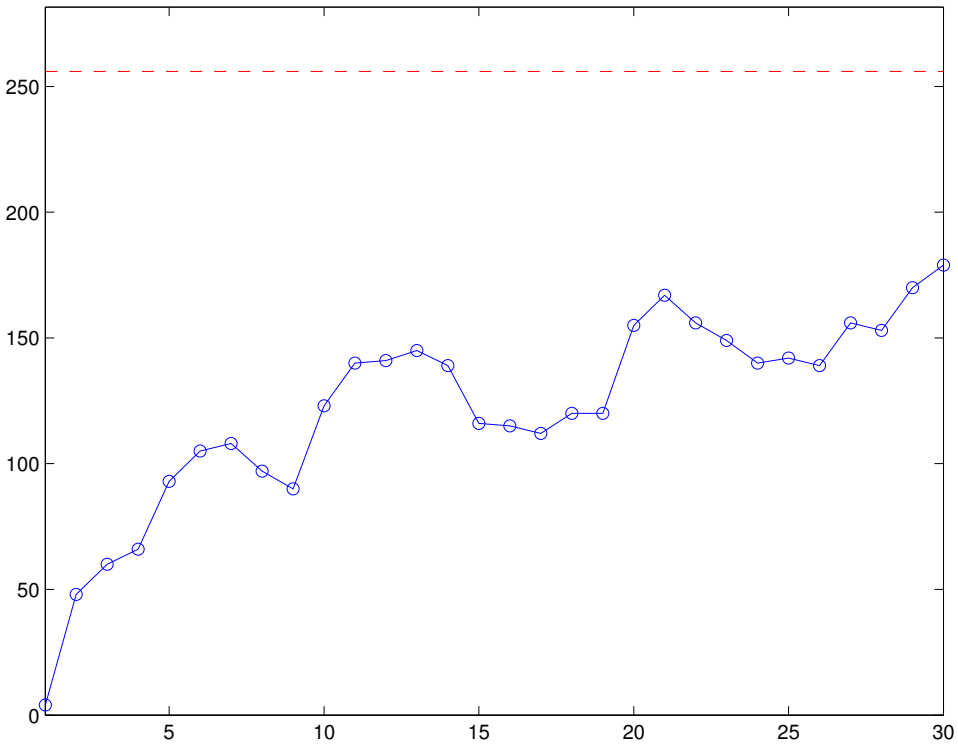
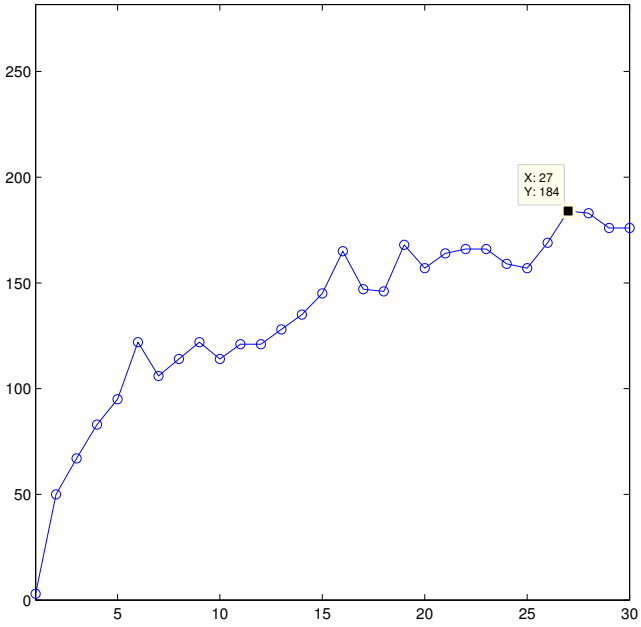


Figure 19: Intermediate cardinality M=16,L=22



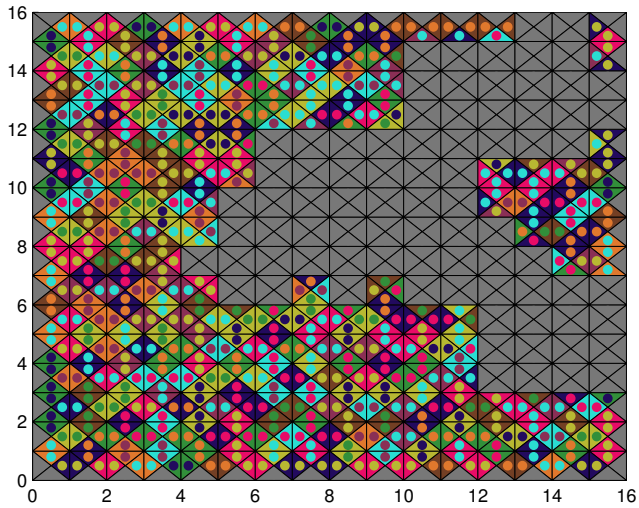


Figure 20: Intermediate solution M=16,L=22, unsolved (placed 176 pieces)

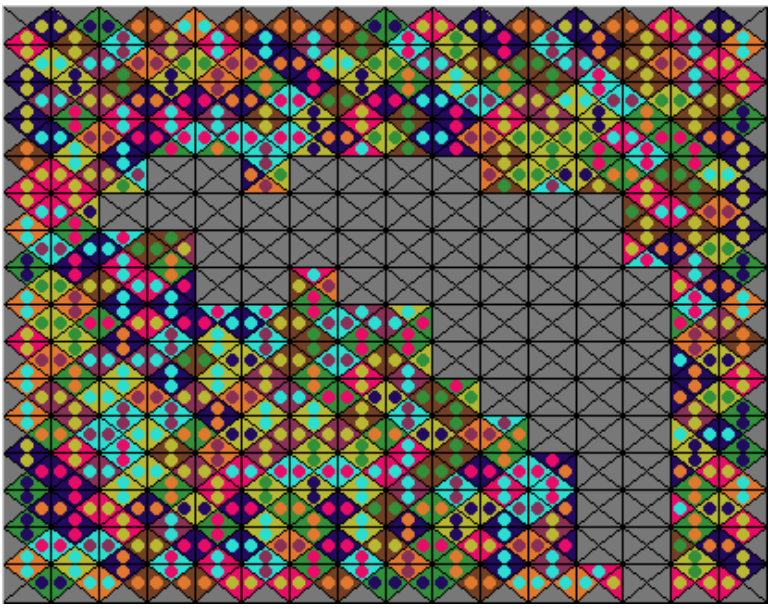


Figure 21: Intermediate solution M=16,L=22, unsolved (placed 197 pieces)

5 Constraint elimination

5.1 Elimination of boundary constraints

The first stage of constraint's elimination results from matrices' properties in constraint

$$(P\mathbf{1} \otimes \beta)^T Y vec\tilde{\Phi} = 0 \tag{34}$$

Studying the above equation we observe that all matrices and vectors $P, \mathbf{1}, \beta, Y$ have by definition only non negative entries. Since right hand equals zero we can deduce that all $(vec\Phi)_i$ entries that correspond to non zero entries of row vector $(P\mathbf{1} \otimes \beta)^T Y$ must be zero. Indicatively, for full game's dimension this stage eliminates 57.840 from 262.144 variables.

After this elimination process zero rows may derive, so zero row removal is carried out.

5.2 Elimination based on polyhedral cone theory

Relying on polyhedral cone theory we extract the following observations,

- Columns of E constitute a set of generators for a pointed polyhedral cone $K = \{Evec\tilde{\Phi} | \tilde{\Phi} \succeq 0\}$.
- Vector b resides on that polyhedral cone's boundary.

Proof. The dual cone of any set (convex or not) K is defined as

$$K^* \triangleq \{y \in \mathbb{R}^n | \langle y, x \rangle \geq 0 \text{ for all } x \in K\} \tag{35}$$

a unique cone that is always closed and convex as the intersection of halfspaces.

Via boundary-membership relations for a proper cone we know that

$$x \in boundary(K) \iff \exists y \neq \mathbf{0} : \langle y, x \rangle = 0, y \in K^*, x \in K \tag{36}$$

b is by default a non zero vector with form $b = \begin{bmatrix} 0 & 0 & \dots & 1 & 1 \end{bmatrix}^T$. Therefore there is always a vector $y \neq \mathbf{0}$ with form

$$y = \begin{bmatrix} y_1 & \dots & y_m & 0 & \dots & 0 \end{bmatrix}^T \tag{37}$$

for $y_1 \dots y_m \neq 0$ for which $\langle y, b \rangle = 0$. So b resides on K's boundary.

□

Our aim is to prune all generators that do not belong in the smallest cone's face, to which b belongs, so as to eliminate a number of constraints. Generators of the smallest face must hold a minimal cardinality solution.

Relying on Dattoro's example (Finding smallest face) we can decide whether or not eliminate a generator by solving the following feasibility problem. E_i column belongs to the smallest face that b resides if and only if problem

$$\begin{aligned} \underset{\tilde{\Phi},\mu\in\mathbb{R}}{find} \quad & \tilde{\Phi},\mu \\ s.t. \quad & \mu b - E_i = Evec\tilde{\Phi} \\ & vec\Phi \succeq 0 \end{aligned} \tag{38}$$

is feasible. Via Saunders transformation problem is reformulated to

$$\begin{aligned} \underset{\tilde{\Phi},\mu\in\mathbb{R}}{find} \quad & \tilde{\Phi},\mu \\ s.t. \quad & \mu b = Evec\tilde{\Phi} \\ & vec\Phi \succeq 0 \\ & (vec\Phi)_i \geq 1 \end{aligned} \tag{39}$$

In our problem any minimal cardinality is by constraints binary. So the above feasibility problem may be reformulated to

$$\begin{aligned} \underset{\tilde{\Phi}}{find} \quad & \tilde{\Phi} \\ s.t. \quad & b = Evec\tilde{\Phi} \\ & vec\Phi \succeq 0 \qquad (vec\Phi)_i = 1 \end{aligned} \tag{40}$$

If the above problem is infeasible then the only choice remaining is $(vec\Phi)_i$ equals 0, which implies that column E_i does not belong to the smallest face generators and may be discarded. After this stage of elimination dimension of prolem is significantly reduced. Column elimination depends on problems geometry.

5.3 Elimination of conically dependent columns

After the latter form of elimination we check matrix E for conically dependent columns. This test is also carried out through a sequence of linear feasibility problems

$$\begin{aligned} \text{find} \quad & x \\ \text{s.t} \quad & Ax = A_i \\ & x \succeq 0 \\ & x_i = 0 \end{aligned} \tag{41}$$

where A_i is the i-th column of A. In case of feasibility column i is conically dependent and can be discarded from the generator set. Though it is observed that smallest face generators in Eternity II found in previous stage are always conically independent. So no further elimination is carried out and this stage is rejected.

6 Feasibility Problem

In any above methods, as stated, many feasibility problems must be solved. All of them are linear programs in normal form, described as:

$$\begin{aligned} \underset{\mathbf{x}}{minimize} \quad & 0 \\ s.t. \quad & E\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{42}$$

The first approach to this was to use CVX to solve them. But it was not successful.

6.1 Inaccuracy of CVX

Problem has a big set of constraints and variables, this created enough numerical errors to make CVX decide erroneously about (in)feasibility of the problem in many cases, even for the smallest 4x4 problem.

In case of deciding to reject the elimination, no harm is made by a misleading (in)feasibility, there will just be more constraints than if no error occurred. But this is not the only outcome.

The worst case is when an elimination is made because of a wrong feasibility test. After such an event the solution space has changed. A group of valid solutions has been eliminated and marked as infeasible. In the absolutely worst scenario, after ending the elimination there may be no more solutions and cardinality minimization can only prove that the rest of the problem is infeasible, if it ever discovers it.

This makes CVX on (42) not a good solution.

Also even by detecting failures of CVX, whenever it reports them, to many possible eliminations seem to be skipped.

6.1.1 Feasibility via the dual using CVX

Next approach was to try to run a feasibility test via its dual. By Farka's lemma feasibility test can be expressed by testing for infeasibility of:

$$\begin{aligned} \underset{\mathbf{y}}{minimize} \quad & 0 \\ s.t. \quad & E^T\mathbf{y} < \mathbf{0} \\ & -b^T\mathbf{y} < \mathbf{0} \end{aligned} \tag{43}$$

This approach yield the same results but false-positives and misses where not in the same tests as of (42).

So next idea was to run this test only when (42) had failed. This decreased speed but gave a small improvement in detection.

6.2 Simplex Method

Trying to eliminate as many constraints as possible, the next step was to use Simplex for the feasibility problem.

Simplex is only working on feasible problems, so there is a need to transform the problem to a feasible. This is done by introducing an extra variable $\mathbf{y}$ and problem's constants A, $\mathbf{c}$:

$$c_i \triangleq |b_i| \qquad \forall i \tag{44a}$$

$$\mathbf{A_{i,*}} \triangleq \begin{cases} \begin{bmatrix} \mathbf{E_{i,*}} & \mathbf{e_i}^T \end{bmatrix}, & b_i \geq 0 \\ \begin{bmatrix} -\mathbf{E_{i,*}} & \mathbf{e_i}^T \end{bmatrix}, & b_i < 0 \end{cases}, \qquad \forall i \tag{44b}$$

Simplex's problem is now:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{minimize} \quad & \mathbf{1}^T\mathbf{y} \\ s.t. \quad & A\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \mathbf{c} \\ & \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \geq \mathbf{0} \end{aligned} \tag{45}$$

which is feasible because $\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \mathbf{c}\end{bmatrix}$ is feasible. If (45)'s solution is zero, then (42) is feasible.

After implementing an using it for feasibility tests, no more wrong pruning where happening. Also for small problems (4x4) there was a speed improvement (x8) vs CVX. But this was still not fast enough. For bigger problems the method was converging too slowly. Also getting into cycling was possible because of degeneracy.

6.3 Revised simplex

Next step was to upgrade the method again. We have fat matrices, so using Simplex is inefficient, comparing to using Revised Simplex. While Simplex updates all the tableau, Revised Simplex only keeps a matrix B corresponding to current step's basic variables. To be more specific, in its improved version, it keeps and updates array B^{-1}. Using this array (and some other smaller structures), it can reproduce all the steps executed and compute all the needed data.

B is a square matrix of size $m \times m$, where m is the population of equality constraints. So, updating B^{-1} is an operation on an array $m \times m$, which is less expensive than a matrix multiplication. Also less operations are needed than in normal Simplex, where a fat matrix $m \times n$ ($n \gg m$) has to be updated.

Changing from Simplex to Revised Simplex almost doubled the speed of elimination.

6.4 Pivoting

Simplex method (Revised and not) are not explicitly stating a pivoting method. In the first approach, selected variable to get in base was picked, based on which variable was having the lowest coefficient (most negative) and thereafter it was swapped with the ith variable of base where $i = argmin\{\frac{\beta_i}{\hat{a_i}} | \hat{a_i} > 0\}$, which is the simplest approach.

Later, a speed improvement has been found based in the observation that as when testing for feasibility, the only goal is to get variable $\mathbf{y}$ out of the base. Picking a variable to get in base was not altered, but it replaced a base variable with another criterion. First it searched in base for variables of vector $\mathbf{y}$ (with indexes b_{av}). And then index of variable to leave base was given by the formula:

$$i = \begin{cases} argmin\{\frac{\beta_i}{\hat{a_i}} | \hat{a_i} > 0, i \in b_{av}\}, & b_{av} \not\equiv \emptyset \\ argmin\{\frac{\beta_i}{\hat{a_i}} | \hat{a_i} > 0, i \notin b_{av}\}, & b_{av} \equiv \emptyset \end{cases} \tag{46}$$

This change made the method not only faster by some orders of magnitude in greater problems, but also applicable.[3].

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, volume 25. Cambridge University Press, 2004.

[2] J. Dattorro. *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2013.03.30 edition, 2013.

[3] David Eigen. Pivot Rules for the Simplex Method, 2011.

[4] J Frausto-Solís and A Nieto-Yáñez. An Improved Simplex-Genetic Method to Solve Hard Linear Programming Problems. *Computational ScienceICCS 2007*, pages 981–988, 2007.

[5] J A J Hall. The practical revised simplex method (Part 2), 2007.

[3] Speed up has not been calculated because for problems not 4x4 without this the time was too big

[6] J A J Hall and K I M Mckinnon. Hyper-sparsity in the revised simplex method and how to exploit it Hyper-sparsity in the revised simplex method and how to exploit it, 2000.

[7] Javier Larrosa. The Revised Simplex Method.

[8] DG Luenberger and Y Ye. *Linear and nonlinear programming*. Springer, third edition, 2008.