1 Installation

In order to use the Ant building system, both `ant` and `cpptasks` have to be installed. For example, in Fedora, these are in the packages `ant` and `cpptasks`, respectively.

Also, the Boost library including headers need to be installed. In Fedora, these are in the package `boost-devel`.

Compiling all programs in this package is done by a single `ant` call in the root directory:

```
$ ant
```

In order to run optimization, Matlab and Cplex need to be installed. For certifying the solution, you will need Mathematica.

2 Generating drawings

There are several separate programs in this package:

- **drawingenum**. This program enumerates all drawings of the $N \times K$ complete bipartite graph. The program takes two arguments (plus perhaps some options), namely N and K. When used with the `-vector` option, its output is a list of space-separated numbers. The first line of the output lists N and K. All other lines are of the form

  $$\langle id \rangle \quad \langle crossing\_count \rangle \quad \langle crossings \rangle$$

  where *id* is just a number that counts the number of drawings, *crossing_count* is the number of crossings that follows, and *crossings* is a list of *crossing_count* quadruples (i, j, k, l), each representing a crossing of the edge $a_i b_j$ with the edge $a_k b_l$. (Here $a_1, \ldots, a_N$ and $b_1, \ldots, b_N$ are the 'white' and 'black' vertices, respectively, of the bipartite graph.)

  Without the `-vector` option, the output of the program is human-readable but less space-efficient.

- **canonical**. This program reads the input from DrawingEnum and calculates the *unlabelled* canonical form of each of the drawings returned by DrawingEnum. Each line is of the form

  $$\langle N \rangle \quad \langle K \rangle \quad \langle N_\ell \rangle \quad \langle K_\ell \rangle \quad \langle \mathrm{cr} \rangle \quad \langle crossings \rangle$$

  where *crossings* is the lexicographically smallest vector that can be obtained by permuting the indices of the white vertices and the black vertices. Notice that the program only deals with unlabelled flags, so $N_\ell = K_\ell = 0$ for each line in the output.

- **generate_drawings**. This program uses DrawingEnum and utils/canonical in combination with the GNU utilities `sort` and `uniq`, to determine all isomorphically different drawings. It takes two arguments, N and K. The results are saved in the file `dr`$\langle N \rangle \langle K \rangle$`.txt`. The output format is the same as utils/canonical.

3 Solving the SDP

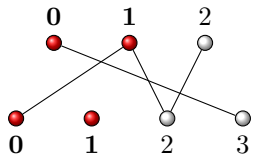
Two directories: `3x3` and `3x4`

Workflow:

- Run `ant generate` in respective directory to generate SDP. This generates: matlab files `parameters.m`, `crossings.m`, `CSineq`i`.m`, `CSmatrix`i`.m`; Mathematica file `problemdata.m`.
- Run `lp` in Matlab to solve SDP
- Run `savecertificate(wineq)` in Matlab to save certificate (even if lp.m was interrupted)
- Run `math < checkcertificate.m` to certify the solution found. Method: see Section 6.

4 Representation of flags

The flags are partially labelled $N \times K$ bipartite graphs in which we store all crossing pairs of edges. For example, consider the following (nonexistent) drawing of $K_{3,4}$ with one crossing:



The flag in this figure has $N = 3$ vertices on the top, $K = 4$ on the bottom, $N_\ell = 2$ labelled vertices on the top (marked in red), $K_\ell = 2$ labelled vertices on the bottom (marked in red), and $\mathrm{cr} = 3$ crossings. So that the labels $0, 1$ on both sides are fixed. The white vertices are unlabelled, i.e., the labels $2, 3$ are meaningless for the flag – they are only *virtual labels* used for representation in the computer. There are three crossings, namely a_2b_2 with a_0b_3, a_1b_2 with a_0b_3, and a_0b_3 with a_1b_0 (in a purposely unnatural order). This flag can be represented by the vector

$$\underbrace{3,4,2,2,1,}_{N,K,N_\ell,K_\ell,\mathrm{cr}} \quad \underbrace{2,2,0,3}_{\text{crossing 1}} \quad \underbrace{1,2,0,3}_{\text{crossing 2}} \quad \underbrace{0,3,1,0}_{\text{crossing 3}}$$

Notice that there is some freedom in how we specify this vector. For example, we may switch the first two entries of each crossing with its last two entries. Moreover, we may reorder the crossings. Finally, since the vertices b_2 and b_3 are unlabelled, we may switch them an obtain the same flag. The *canonical form* of a flag is the representation over all permutations of virtual labels of the unlabelled vertices, all permutations of the crossings, and all permutations of the edges within the crossings. The canonical form of the flag above can be constructed by switching labels 2 and 3 on the bottom of the graph, which gives

$$\underbrace{3,4,2,2,1,}_{N,K,N_\ell,K_\ell,\mathrm{cr}} \quad \underbrace{2,3,0,2}_{\text{crossing 1}} \quad \underbrace{1,3,0,2}_{\text{crossing 2}} \quad \underbrace{0,2,1,0}_{\text{crossing 3}}$$

and then reordering the crossings and the edges inside them:

$$\underbrace{3,4,2,2,1,}_{N,K,N_\ell,K_\ell,\mathrm{cr}} \quad \underbrace{0,2,1,0}_{\text{crossing 3}} \quad \underbrace{0,2,1,3}_{\text{crossing 2}} \quad \underbrace{0,2,2,3}_{\text{crossing 1}}$$

This canonical form has the property that two flags F_1 and F_2 have the same canonical form if and only if they are isomorphic.

We construct the set $\mathcal{F}(N, K; N_\ell, K_\ell)$ as follows. We first generate all drawings of the $N \times K$ complete bipartite graph in the plane, with the property that no incident edges cross and no two edges cross more than once. For each such drawing, we go through all combinations of ordered subsets of $\{0, \dots, N-1\}$ (of size N_ℓ) and ordered subsets of $\{0, \dots, K-1\}$ (of size K_ℓ), and label the vertices in these subsets accordingly. We then calculate the canonical form of each of the resulting labelled flags and insert it into the $\mathcal{F}(N, K; N_\ell, K_\ell)$ if this canonical form has not been seen before.

For brevity, we will write $\mathcal{F}(N, K) = \mathcal{F}(N, K; 0, 0)$. The flag generation procedure above gives $|\mathcal{F}(3, 3)| = 112$, $|\mathcal{F}(3, 4)| = 6393$, and $|\mathcal{F}(3, 3; 3, 2)| = 3618$.

5 The Cauchy-Schwarz matrix

Fix N', K', N'_ℓ, K'_ℓ satisfying $0 \le N'_\ell \le N' \le N$, $0 \le K'_\ell \le K' \le K$, $2N_\ell - N' \ge 0$, and $2K_\ell - K' \ge 0$. These conditions ensure that the number of top and bottom vertices in the product of two elements of $\mathcal{F}(N', K'; N'_\ell, K'_\ell)$ does not exceed the number of top and bottom of an element in $\mathcal{F}(N, K)$.

We now construct a matrix $Z(x)$ whose rows and columns are indexed by $\mathcal{F}(N', K'; N'_\ell, K'_\ell)$, and where x is a probability function on $\mathcal{F}(N, K)$. Each entry corresponding to $F_1, F_2 \in \mathcal{F}(N', K'; N'_\ell, K'_\ell)$ is given by

$$\sum_{F \in \mathcal{F}(N,K)} p(F_1, F_2; F) x(F),$$

This gives a matrix We construct the matrix $Z(x)$ by going through each flag $F \in \mathcal{F}(N, K)$, constructing

6 Certifying the correctness of the solution

We are trying to solve the following semidefinite optimization problem:

$$\begin{aligned} z^* = \min\, & c^\mathsf{T} x \\ & e^\mathsf{T} x = 1 \\ & Bx \leq b, \\ & A_m(x) \succeq 0, \qquad m = 1, \ldots, M. \\ & x \geq 0, \end{aligned}$$

where x_i is the i'th flag, c_i is the number of crossings in the i'th flag, and $A_m(x)$ is the m'th Cauchy-Schwarz matrix. Notice that $A_m(x)$ is linear in x, so that we can write

$$A_m(x) = \sum_{i=1}^{n} x_i F_i^m,$$

where F_i^m is a constant matrix. Since $A_m(x)$ is positive semidefinite if and only if $w^\mathsf{T} A_m(x) w \geq 0$ for all $w \in \mathbb{R}^{n_m}$, this semidefinite problem may be written as a linear optimization problem with infinitely many constraints as follows:

$$\begin{aligned} z^* = \min\, & c^\mathsf{T} x \\ & e^\mathsf{T} x = 1 \\ & Bx \leq b, \\ & \sum_{i=1}^{n} \left(w^\mathsf{T} F_i^m w\right) x_i \geq 0, \text{ for all } w \in \mathbb{R}^{n_m},\ i = 1, \ldots, M. \\ & x \geq 0. \end{aligned}$$

Our software generates a finite set $\{w_1, \ldots, w_k\}$ of rational vectors w, where vector w_j corresponds to matrix $A_{m(j)}$. Hence, our software solves the linear optimization problem

$$\begin{aligned} z' = \min\, & c^\mathsf{T} x \\ & e^\mathsf{T} x = 1 \\ & Bx \leq b, \\ & \sum_{i=1}^{n} \left(w_j^\mathsf{T} F_i^{m(j)} w_j\right) x_i \geq 0, \text{ for } j = 1, \ldots, k. \\ & x \geq 0. \end{aligned}$$

Clearly, since the constraints of this latter problem are a subset of infinite set of constraints of the original problem, we have that $z' \leq z^*$. The dual of this problem is

$$\begin{aligned} z_D = \max\, & \mu - \nu^\mathsf{T} b \\ & \mu - \sum_{j=1}^{p} \nu_j B_j + \sum_{j=1}^{k} \left(w_j^\mathsf{T} F_i^{m(j)} w_j\right) \lambda_j \leq c_i^\mathsf{T} \\ & \nu \geq 0, \lambda \geq 0, z \text{ unrestricted.} \end{aligned}$$

By strong duality, we have $z_D = z' \leq z^*$. Thus, we have the following theorem:

Theorem 1. *Let $\lambda \geq 0$ and $\nu \geq 0$. Let*

$$\mu = \min_{j=1,\ldots,n} \left\{ c_i^T + \sum_{j=1}^{p} \nu_j B_j - \sum_{j=1}^{k} \lambda_j w_j^T F_i^{m(j)} w_j \right\}.$$

Then, (λ, ν, μ) is a dual feasible solution with objective value $\mu - \nu^T b$, and hence $z^* \geq \mu - \nu^T b$.

If

$$\mu - \sum_{i=1}^{p} \nu_j B_j + \sum_{j=1}^{k} \lambda_j w_j^T F_i^{m(j)} w_j \leq c_j,$$

for all $i = 1, \ldots, n$, then (μ, ν, λ) is a dual solution that certifies that $z^* \geq z' = z_D \geq \alpha$, as required.

7 Results

7.1 3 by 3 case

We obtain

$$z^* \geq 1.95245.$$

This correponds to 0.86775 times the Zarankiewicz bound

7.2 3 by 4 case

We obtain

$$z^* \geq 4.07257.$$

This correponds to 0.90501 times the Zarankiewicz bound