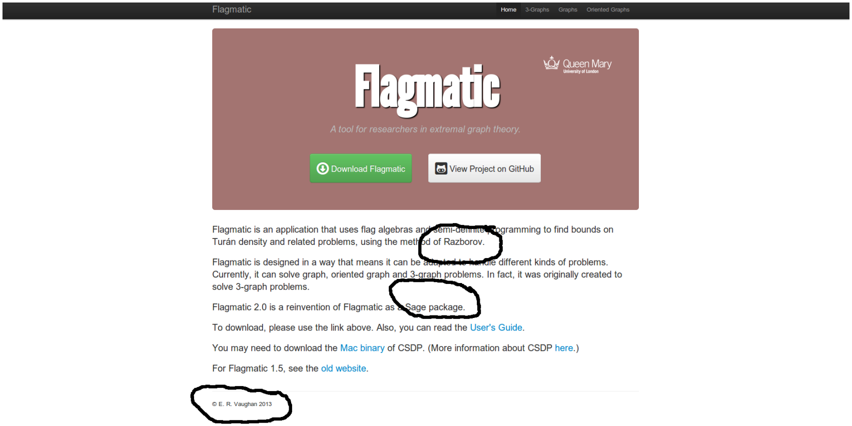
Flagmatic

Jakub Sliačan

University of Warwick

April 13, 2015



Flagmatic

A tool for researchers in extremal graph theory.

Flagmatic is an application that uses flag algebras and semi-definite programming to find bounds on Turán density and related problems, using the method of Razborov.

Flagmatic is designed in a way that means it can be adapted to handle different kinds of problems. Currently, it can solve graph, oriented graph and 3-graph problems. In fact, it was originally created to solve 3-graph problems.

Flagmatic 2.0 is a reinvention of Flagmatic as a Sage package.

To download, please use the link above. Also, you can read the User's Guide.

You may need to download the Mac binary of CSDP. (More information about CSDP here.)

For Flagmatic 1.5, see the old website.

© E. R. Vaughan 2013

`https://github.com/jsliacan/flagmatic-dev.git`
(tested with Sage 6.4)

Problem type

Maximize induced density of a small H in a big F-free G.

induced density: $\#$ induced copies of H in G, normalized by $\binom{|G|}{|H|}$

Example

Maximize the density of *in a* *-free G when $|G| \to \infty$.*

Answer

$\phi($ $) \leq 1/2$. *Complete balanced bipartite G:* $\phi($ $) \geq 1/2$.

How?

Asymptotic density only:

$$\phi(\underset{\bullet}{\overset{\bullet}{|}}) = \lim_{n\to\infty} \max_{|G|=n} d(\underset{\bullet}{\overset{\bullet}{|}}; G) \quad \text{(exists)}$$

Start small:

$$d(\underset{\bullet}{\overset{\bullet}{|}}; G) = \sum_{|F|=k} d(\underset{\bullet}{\overset{\bullet}{|}}; F) d(F; G)$$

Do not know $d(F;G)$, but $\sum_{|F|=k} d(F;G) = 1$.

Bound:

$$d(\underset{\bullet}{\overset{\bullet}{|}}; G) \leq \max_{|F|=k} d(\underset{\bullet}{\overset{\bullet}{|}}, F) \qquad \text{(poor)}$$

Need a better bound

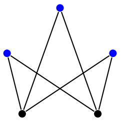
The above bound is rarely sharp.

Example

$d($ $; G) \le \max_{|F|=3} d($ $; F)$

$= d($ $;$ $) = 2/3$

Only sharp if every subgraph of G on 3 vertices is a *.*
Impossible for G with ≥ 5 vertices:



Account for subgraph overlaps

G● is G with one vertex red. Then $d($ $, G$●$)$ is the normalized degree of the red vertex.

1. $d($ $; G$●$)d($ $; G$●$)$ choosing two neighbours of ● (repetition allowed)
2. $d($ $,$ $; G$●$) = d($ $; G$●$) + d($ $; G$●$)$ choosing two neighbours of ● (repetition disallowed)

Negligible difference when G big. $\implies$ start with 1., switch to 2., uncolor (average over all choices of ● in G). Left with $\alpha d($ $; G)$.

$\llbracket d($ $; G$●$)d($ $; G$●$) \rrbracket$● $\sim \frac{1}{3} d($ $; G)$

Manipulation

Vector $v = [d($ $; G_{\bullet}), d($ $; G_{\bullet})]$.

$$\llbracket vv^T \rrbracket_{\bullet} \geq 0$$

Similarly, for every $A \succeq 0$,

$$\llbracket vAv^T \rrbracket_{\bullet} \geq 0$$

$$
\begin{aligned}
d(\text{}&\text{}
\end{aligned}
$$
$d($ $; G) = \sum_{|F|=3} d($ $; F) d(F; G)$

$\leq \sum_{|F|=3} d($ $; F) d(F; G) + \llbracket vAv^T \rrbracket \qquad \text{with } A \succeq 0$

$= \sum_{|F|=3} \left(d($ $; F) + c_F \right) d(F; G)$

$\leq \max_{|F|=3} d($ $; F) + c_F$

Delegating tasks to the PC

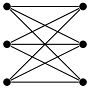
Clearly, the proces was rather systematic. Need to know: density graphs, forbidden graphs. The rest can be done by the PC.

Optimization:

$$\begin{gathered}\min \gamma : \\ d(\overset{\bullet}{\underset{\bullet}{|}}; F) + c_F \leq \gamma \quad \text{,for all } F \\ A \succeq 0\end{gathered}$$

Mantel in Flagmatic

Maximise in a graph without copies of .
Recall $\phi($ $) \leq 1/2$. Extremal graph is complete balanced bipartite:

e.g. 

In Flagmatic 2.0 [Emil's]

Listing 1: Mantel's theorem.

```
p = GraphProblem(3, forbid="3:121323")
c = GraphBlowupConstruction("2:12")
p.set_extremal_construction(c)
p.solve_sdp(solver="csdp")
p.make_exact()
```

Listing 2: Output

```
Forbidding 3:121323 as a subgraph.
Generating graphs...
Generated 3 graphs.
Generating types and flags...
Generated 1 types of order 1, with [2] flags of order 2.
Computing products.
Writing SDP input file...
Running SDP solver...
Returncode is 0. Objective value is 0.50000001.
Checking numerical bound...
Bound of 1/2 attained by:
1/2 : graph 0 (3:)
1/2 : graph 2 (3:1213)
```

Three modes of Flagmatic-dev

- Plain mode [Emil's]
- Optimization mode [Assumptions]
- Feasibility mode. [No objective function]

Plain mode

D^* quantum graph $a_1 D_1 + \dots a_k D_k$, some k
$\mathcal{T}$ set of type graphs

$$\boxed{\begin{aligned} &\min \delta : \\ &\quad D^* + \sum_{\tau \in \mathcal{T}} \left[\!\left[\mathbf{p}_\tau Q_\tau \mathbf{p}_\tau^T \right]\!\right]_\tau \le \delta \\ &\quad Q_\tau \succeq 0, \quad \forall \tau \in \mathcal{T} \\ &\quad \delta \ge 0 \end{aligned}}$$

Example: Minimizing monochromatic 4-cliques in a 2-colored clique

Sperfeld '12

$$m_{K_4+\overline{K_4}} \geq \frac{1}{34.7858} = 0.0287473624294971$$

Listing 3: In Flagmatic

```
p = GraphProblem(8, density=[("4:",1),("4:121314232434", 1)],
                    minimize=True)
p.solve_sdp(solver="csdp")
p.make_exact()
p.write_certificate("monocolor.cert")
```

Optimization mode

Assumption

$$S = \sum_{\substack{W \in \mathcal{F}' \subseteq \mathcal{F}^\sigma \\ |\mathcal{F}'| < \infty}} b_W W \geq b, \quad b_w \in \mathbb{R}$$

SDP problem

$$\begin{aligned}
\min \delta :& \\
D^* &+ \left[\!\left[(S_1 - b_1) \sum_{i=1}^{l_1} c_i^1 F_i^1 \right]\!\right]_{\sigma_1} + \ldots + \left[\!\left[(S_M - b_M) \sum_{i=1}^{l_M} c_i^M F_i^M \right]\!\right]_{\sigma_M} + \sum_{\tau \in \mathcal{T}} \left[\!\left[\mathbf{p}_\tau Q_\tau \mathbf{p}_\tau^T \right]\!\right]_\tau \leq \delta \\
Q_\tau &\succeq 0, \quad \forall \tau \in \mathcal{T} \\
c_i^1 &\geq 0, \quad \forall i = 1, \ldots, l_1 \\
&\vdots \\
c_i^M &\geq 0, \quad \forall i = 1, \ldots, l_M \\
\delta &\geq 0
\end{aligned}$$

Example: Sós problem

Let G be your big graph with edge density $1/2$. Suppose you know that if you randomly sample 4 vertices from $V(G)$, then the number of edges you see on them (as induced in G) is exactly what it would be in an Erdős-Rényi random graph $\mathbb{G}(n, 1/2)$.

Then your graph G is $\frac{1}{2}$-pseudorandom.

Example

Listing 4: Sós problem

```
N = binomial(4,2)

def dens(pp, n, k):
    return binomial(n,k)*pp^k*(1-pp)^(n-k)

sp = GraphProblem(4,
                  density=[("4:12132434(0)", -4), ("4:12233124(0)", 1)
                           ("4:1434(0)", 1), ("4:1324(0)", -4)],
                  types=["2:","2:12"],
                  mode="optimization")

sp.add_assumption("0:", [("4:(0)", 1)], dens(1/2, N, 0), equality=True
sp.add_assumption("0:", [("4:12(0)", 1)], dens(1/2, N, 1) , equality=T
sp.add_assumption("0:", [("4:1223(0)", 1), ("4:1234(0)", 1)], dens(1/2
                  equality=True)
sp.add_assumption("0:", [("4:121314(0)", 1), ("4:122334(0)", 1), ("4:1
                  dens(1/2, N, 3), equality=True)
sp.add_assumption("0:", [("4:12233441(0)", 1), ("4:12233134(0)", 1)],
                  equality=True)
sp.add_assumption("0:", [("4:1223344113", 1)], dens(1/2, N, 5), equali
sp.add_assumption("0:", [("4:122334411324", 1)], dens(1/2, N, 6), equa
sp.solve_sdp(solver="csdp")
```

Feasibility mode

[Not tested]

$$
\begin{aligned}
& \min \delta : \\
& \left[\!\left[(S_1 - b_1) \sum_{i=1}^{l_1} c_i^1 F_i^1 \right]\!\right]_{\sigma_1} + \ldots + \left[\!\left[(S_M - b_M) \sum_{i=1}^{l_M} c_i^M F_i^M \right]\!\right]_{\sigma_M} + \sum_{\tau \in \mathcal{T}} \left[\!\left[\mathbf{p}_\tau Q_\tau \mathbf{p}_\tau^T \right]\!\right]_\tau \leq \delta \\
& Q_\tau \succeq 0, \quad \forall \tau \in \mathcal{T} \\
& c_i^1 \geq 0, \quad \forall i = 1, \ldots, l_1 \\
& \vdots \\
& c_i^M \geq 0, \quad \forall i = 1, \ldots, l_M \\
& \sum_{j=1}^{M} \sum_{i=1}^{l_j} c_i^j = 1
\end{aligned}
$$