COMPUTING THE TUTTE POLYNOMIAL FOR RESTRICTED "WIDTH"

Petr Hliněný

Faculty of Informatics,
Masaryk University in Brno,
Botanická 68a, 602 00 Brno, Czech Rep.

e-mail: hlineny@fi.muni.cz
http://www.cs.vsb.cz/hlineny

Parts of the talk present joint work with

Omer Gimenez and **Marc Noy**

Dept. of Applied Mathematics
UPC Barcelona

1 THE TUTTE POLYNOMIAL

As everybody here probably knows. . .

Definition. For a graph $G = (V, E)$,

$$T(G; x, y) = \sum_{F \subseteq E} (x-1)^{r(E)-r(F)} (y-1)^{|F|-r(F)},$$

where $r(F) = |V| - k(F)$ and $k(F)$ is the num. of components induc. by (V, F).

This definition of the Tutte polynomial follows its matroid aspects:

$$T(M; x, y) = \sum_{A \subseteq E} (x-1)^{r_M(E)-r_M(A)} (y-1)^{|A|-r_M(A)}$$

Fact. Knowing $T(G; x, y)$ $\sim$ knowing the number of spanning subgraphs on edges F with $|F| = i$ and $k(F) = j$.

Fact. The Tutte polynomial captures a number of interesting graph properties:

- $T(G;1,1) = \#$ spanning trees,
- $T(G;2,1) = \#$ spanning forests,
- $T(G;1-x,0)\cdot * =$ the chromatic polynomial,
- $T(G;0,1-y)\cdot * =$ the flow polynomial.
- and many more...

So, not surprisingly, its computation is very hard in general...

Theorem 1.1. [Jaeger, Vertigan, and Welsh, 1990]
Evaluating the Tutte polynomial $T(G;x,y)$ at $(x,y)=(a,b)$ is $\#P$-hard unless $(a-1)(b-1)=1$ or $(a,b)\in\{(1,1),(-1,-1),(0,-1),\ (-1,0),(i,-i),(-i,i),(j,j^2),(j^2,j)\}$, where $i^2=-1$ and $j=e^{2\pi i/3}$.

2 COMPUTING FOR RESTRICTED "WIDTH"

2.1 Tree-width / branch-width

Motivation: Many hard graph properties can be computed efficiently for graphs of bounded tree-width (for example, all MSO-definable properties).

- Independently [Andrzejak / Noble, both 1998]:

  The Tutte polynomial $T(G; x, y)$ can be computed in polynomial time on a graph G of bounded tree-width.

  – The (stronger) version of Noble gives an FPT algorithm, and

  – an evaluation scheme using linear number of arithmetic operations.

- Our matroidal extension:

  Theorem 2.1. [PH, 2003] *The Tutte polynomial $T(M; x, y)$ can be computed in polynomial FPT time on a matroid M, which is represented by a matrix over a finite field and has bounded branch-width.*

  – We generalize the approach of Noble, and provide a "cleaner view" of the computation using branch-width instead of tree-width.

2.2 Cographs (i.e. clique-width 2)

This is a simplified version of the full (and difficult) algorithm for graphs of bounded clique-width…

Theorem 2.2. [Giménez, PH, Noy, 2005]
The Tutte polynomial of a cograph can be computed in subexponential time

$$\exp\left(O(n^{2/3})\right).$$

Note: Subexponential algorithms – $2^{o(n)}$

For NP-complete problems, no better solutions than an exhaustive search are expected to exist.

Hence, for naturally defined problems like the SAT with n variables, no $2^{o(n)}$ algorithm (called often *subexponential*) is expected to exist.

2.3 Clique-width / rank-width

Theorem 2.3. [Giménez, PH, Noy, 2005]
Let G be a graph with n vertices of clique-width $\leq k$ along with a k-expression for G as an input. Then the Tutte polynomial of G can be computed in subexponential time

$$\exp\left(O(n^{1-\frac{1}{k+2}})\right) .$$

Do we need a k-expression (i.e. a given decomposition) for G?

Clique-width is difficult to compute.
However, it is efficiently approximable via *rank-width*. [Oum, Seymour, 03]

Fact. A subexp. $2^{o(n)}$ algorithm for the Tutte polynomial on an n-vertex graph

$\longrightarrow$ a $2^{o(n)}$ algorithm for 3-colouring,

$\longrightarrow$ a $2^{o(n)}$ algorithm for 3-SAT – unexpected!

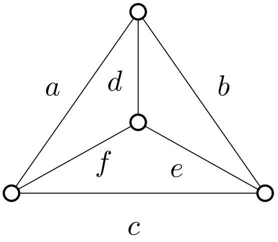
So it is very unlikely to have a subexponential algorithm for the Tutte polynomial on general graphs. . .

3 SKETCHING THE PROOFS

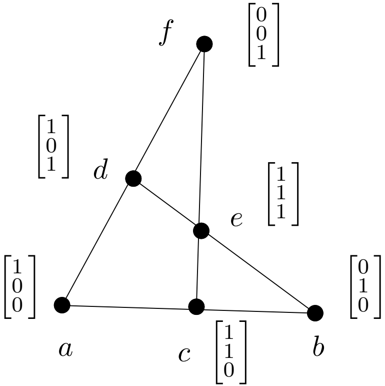
Starting with a few words about represented matroids...

- Matroids represented by matrices over a finite field $\mathbb{F}$;
- $\rightarrow$ elements give actual *points in the projective geometry* over $\mathbb{F}$.
- An illustration of the relation between graphic and represented matroids:

$\boldsymbol{K_4}$



$\longrightarrow$



3.1 The Tutte Polynomial on Matroids

Introducing the boundaried Tutte polynomial...

- *Boundaried matroid* $\bar{M}, \partial$ – a represented matroid M equipped with an arbitrary boundary subspace ∂.

  t-boundary – boundary of rank t.

- *t-boundary mark* $\mathrm{K}(\bar{M}\,|A)$ – marking the subspace $\partial(\bar{M}) \cap \langle A \rangle$ of the boundary $\partial(\bar{M})$ that is spanned by A.

  $\mathcal{K}_t^{\sim}$ – the set of all t-boundary marks.

- Let $\bar{M} = (M, \partial)$ be a t-boundaried represented matroid on E.
  The *boundaried Tutte polynomial* of $\bar{M}$ is given by

$$T_B\left(\bar{M}; x, y, Z_t\right) = \sum_{A \subseteq I} z_{\mathrm{K}\left(\bar{M}\,|A\right)} \cdot (x-1)^{r_M(I) - r_M(A)} \cdot (y-1)^{|A| - r_M(A)},$$

  where $Z_t = \left(z_{\mathrm{K}} : \mathrm{K} \in \mathcal{K}_t^{\sim}\right)$ is a vector of $|\mathcal{K}_t^{\sim}|$ free variables.

Proposition 3.1. $T(M; x, y) = T_B\left(\bar{M}; x, y, (1, \ldots, 1)\right)$.

Recursive Computation of the Boundaried Tutte Polynomial

Theorem 3.2. *Let a tree T be parsing a t-branch-decomposition of a represented boundaried matroid $\bar{M} = \bar{M}(T)$. If T is an empty tree, then*

$$T_B\left(\bar{M}(T);\, x, y, Z_0\right) = T_B\left(\bar{\Omega}_0;\, x, y, Z_0\right) = z_{\mathrm{K}\left(\bar{\Omega}_0\,|\emptyset\right)}\,.$$

If T has exactly one vertex labelled by $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$, then

$$T_B\left(\bar{\Upsilon};\, x, y, Z_1\right) = z_{\mathrm{K}\left(\bar{\Upsilon}\,|\emptyset\right)}(x-1) + z_{\mathrm{K}\left(\bar{\Upsilon}\,|I(\bar{\Upsilon})\right)}\,, \text{ or}$$

$$T_B\left(\bar{\Upsilon}_0;\, x, y, Z_0\right) = z_{\mathrm{K}\left(\bar{\Upsilon}_0\,|\emptyset\right)} + z_{\mathrm{K}\left(\bar{\Upsilon}_0\,|I(\bar{\Upsilon}_0)\right)}(y-1)\,.$$

If r is the root with composition $\odot$, and T_1, T_2 are the sons of r in T, then

$$T_B\left(\bar{M}(T);\, x, y, Z_{t_3}\right) =$$

$$= T_B\left(\bar{M}(T_1);\, x, y, Z'_{t_1}\right) \cdot T_B\left(\bar{M}(T_2);\, x, y, Z''_{t_2}\right),$$

where

$$z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2} = z_{\mathrm{K}_3(\odot;\,\mathrm{K}_1,\mathrm{K}_2)} \cdot (x-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)-\sigma(\odot)} \cdot (y-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)}$$

for each pair $\mathrm{K}_i \in \mathcal{K}^{\sim}_{t_i(\odot)}$, $i = 1, 2$.

Theorem 3.3. *Computing time summary for the Tutte polynomial on represented matroids:*

Assume that $\mathbb{F}$ is a finite field, and that t is an integer constant.

- *If M is an n-element $\mathbb{F}$-represented matroid of branch-width at most t, then the Tutte polynomial $T(M; x, y)$ can be computed in time*

$$\boldsymbol{O(n^6 \log n \log\log n)\,.}$$

- *Suppose that a, b are rational numbers $a = \frac{p_a}{q_a}$, $b = \frac{p_b}{q_b}$ of combined length l bits. Then $T(M; a, b)$ can be evaluated at a, b in time*

$$\boldsymbol{O(\,n^3 + n^2 l \cdot \log(nl) \cdot \log\log(nl)\,)\,.}$$

Remark. Noble evaluates the Tutte polynomial $T(G; a, b)$ at a, b for a graph G of bounded tree-width in time

$$O((v+p) \cdot el \cdot \log e \log\log e \cdot \log l \log\log l)\,,$$

where v is the number of vertices, e is the number of edges, and p the the size of the largest parallel class in G. Note that $n = e$ in our setting.

Our algorithm almost matches this performance, the extra $O(n^3)$ term is needed to construct the necessary branch-decomposition.

3.2 Forests in Cographs

The first (simplified) step towards the algorithm for graphs of bounded clique-width...

Definition. **Cograph** is a graph constructed from vertices using

- a *disjoint union* (no added edges), or
- a *"complete" union* (adding all edges across).

Fact. (folklore)

- All cliques are cographs.
- Precisely those graphs without induced P_4.
- Cographs are closed on complements, contractions, induced subgraphs.
- Not closed on normal subgraphs / edge deletion.
- Recognizable in P.

Theorem 3.4. *Spanning forests can be enumerated on cographs in time*

$$\exp\left(O(n^{2/3})\right).$$

Algorithm on Cographs

A **forest signature** $\boldsymbol{\alpha}$ – a multiset of component sizes (positive integers);

- represented by a *characteristic vector* $\boldsymbol{\alpha} = (a_1, a_2, \dots, a_n)$,
- *size* $s_{\boldsymbol{\alpha}} = \sum_{i=1}^{n} i \cdot a_i$ (and cardinality as usual $|\boldsymbol{\alpha}| = \sum_{i=1}^{n} a_i$).

Lemma 3.5. (folklore) *There are $2^{\Theta(\sqrt{n})}$ signatures of size n ($\sim$integer parts.).*

A **forest double-signature** $\boldsymbol{\beta}$ – a multiset of ordered pairs of integers, counting dual-labeled (nonempty) component sizes;

- a refinement of a forest signature,
- having a *characteristic vector* $\boldsymbol{\beta} = (b_{(0,1)}, b_{(0,2)}, \dots, b_{(1,0)}, b_{(1,1)}, \dots)$,
- *size* $s_{\boldsymbol{\beta}} = \sum_{(x,y)} (x+y) \cdot b_{(x,y)}$.

Lemma 3.6. *There are $\exp\left(\Theta(n^{2/3})\right)$ distinct double-signatures of size n.*

– Quite difficult to prove, but easy a slightly worse bound $\exp\left(\Theta(n^{2/3}\log n)\right)$.

We apply the following two $\exp\left(O(n^{2/3})\right)$ algorithms along the decomposition scheme of the given cograph:

Algorithm 3.7. *Combining the spanning forest signature tables of graphs F and G into the one of the disjoint union $H = F \dot\cup G$. (Simple.)*

`Input:` Graphs F, G, and their forest signature tables $\boldsymbol{T}_F, \boldsymbol{T}_G$.

`Output:` The forest signature table $\boldsymbol{T}_H$ of $H = F \dot\cup G$.

`create` *empty table* $\boldsymbol{T}_H$ *of forest signatures of size* $|V(H)|$;
`for` *all signatures* $\boldsymbol{\alpha}_F \in \Sigma_F$, $\boldsymbol{\alpha}_G \in \Sigma_G$ `do` $\exp\left(O(n^{2/3})\right) \times$
  `set` $\boldsymbol{\alpha} = \boldsymbol{\alpha}_F \uplus \boldsymbol{\alpha}_G$ *(a multiset union)*;
  `add` $\boldsymbol{T}_H[\boldsymbol{\alpha}] += \boldsymbol{T}_F[\boldsymbol{\alpha}_F] \cdot \boldsymbol{T}_G[\boldsymbol{\alpha}_G]$;
`done.`

Algorithm 3.8. *Combining the spanning forest signature tables of graphs F and G into the one of the complete union $H = F \oplus G$. (Difficult.)*

`Input:` Graphs F, G, and their forest signature tables $\boldsymbol{T}_F, \boldsymbol{T}_G$.

`Output:` The forest signature table $\boldsymbol{T}_H$ of $H = F \oplus G$.

`create` *empty table* $\boldsymbol{T}_H$ *of forest signatures of size* $|V(H)|$;

```
for all signatures α_F ∈ Σ_F, α_G ∈ Σ_G do                                   exp(O(n^{2/3}))×
   set z = |V(F)|;
   create empty table X of forest double-signatures of size z;
   set X[double-signature {(a,0) : a ∈ α_F}] = 1;
   for each c ∈ α_G (with repetition) do                                      O(n)×
      create empty table X′ of forest double-signatures of size z + c;
      for all double signatures β of size z s.t. X[β] > 0 do                  exp(O(n^{2/3}))×
(*)      for all submultisets γ ⊆ β (with repetition) do                      exp(O(n^{2/3}))×
            set d_1 = Σ_{(x,y)∈γ} x, d_2 = Σ_{(x,y)∈γ} y;
            set double-signature β′ = (β − γ) ⊎ {(d_1, d_2 + c)};
            add X′[β′] += X[β] · Π_{(x,y)∈γ} cx;                              O(n)
         done
      done
      copy X = X′, z = z + c; dispose X′;
   done
   for all double-signatures β of size |V(H)| do                             exp(O(n^{2/3}))×
      set signature α_0 = {x + y : (x,y) ∈ β};
      add T_H[α_0] += X[β] · T_F[α_F] · T_G[α_G];
   done
done.
```

3.3 The Tutte Polynomial on Cographs

Extending Algorithms 3.7,3.8 for the Tutte polynomial is not so difficult...

Extensions:

- Enumerate edge-subsets (spanning subgraphs) instead of forests.
- *Subgraph signatures* analogously record the component sizes. Moreover, we record the total number of edges.
- When joining components, we may add many (≥ 1) edges between two components, $\rightarrow$ computing "cellular selections".

Definition. *Cellular selection* from $C_1, \ldots, C_k$:
Selecting an ℓ-element subset $L \subseteq C_1 \cup \ldots C_k$, st. $L \cap C_i \neq \emptyset$ for all i.

A nice exercise: Let $d_i = |C_i|$, and $u_{i,j}$ be the number of partial selections of j elements from the first i cells. Then

$$u_{i,j} = \sum_{s=1}^{r} u_{i-1,j-s} \cdot \binom{d_i}{s}.$$

Theorem 3.9. *The Tutte polynomial of a cograph can be computed in time*

$$\exp\left(O(n^{2/3})\right).$$

3.4 Clique-Width

- Formal definition [Courcelle, Olariu, 00] (implicit [Courcelle et al, 93]).

Definition. Constructing a vertex-labeled graph G using the operations

- a new labeled vertex,
- a disjoint union of two graphs
- $\rho_{i\to j}$ relabeling of all i's to j's,
- η_{i-j} adding all edges between labels i and j.

(Called a *k-expression*.)

Clique-width $= \min$ number of labels needed to construct (unlabeled) G.

- Cographs have clique-width $= 2$, paths ≤ 3, cycles ≤ 4.
- Bounding the clique-width of a graph allows to efficiently solve all problems expressed in the MSO logic of adjacency graphs (MS$_1$) – quantifying over vertices and their sets. [Courcelle, Makowsky, Rotics, 00]

  (Bounding the tree-width allows to efficiently solve all problems in MS$_2$.)
- The chromatic number (and the chromatic polynomial) is polynomial time (not FPT) for graphs of bounded clique-width. [Kobler, Rotics, 03]

Algorithm on Bounded Clique-Width

A **subgraph k-signature** $\boldsymbol{\beta}$ – a multiset of ordered k-tuples of integers, counting k-labeled (nonempty) component sizes.

(Analogous to double-signatures. . .)

Lemma 3.10. *There are* $\exp\left(\Theta(n^{k/(k+1)})\right)$ *distinct k-signatures of size n.*

Extending the algorithm – processing the η_{i-j} operation:

- Using only one signature table for the whole graph.
- Thus need an artificial new label 0 for iterative processing of components intersecting label j (corresp. to the sign. table of the second graph).
- A new (easy) point of adding edges inside a component.

Our full result:

Theorem 3.11. *Let G be a graph with n vertices of clique-width $\leq k$ along with a k-expression for G as an input. Then the Tutte polynomial of G can be computed in time*

$$\exp\left(O(n^{1-\frac{1}{k+2}})\right) .$$

4 OPEN QUESTIONS

Just a few ones related to our talk. . .

- [Kobler, Rotics, 03] compute the chromatic number of a graph of bounded clique-width in polynomial time, however, not in FPT.

  Is the chromatic number FPT wrt. clique-width?
  (i.e. polynomial with a fixed exponent?)

- Is the Tutte polynomial on graphs of bounded clique-width in P, or #P-hard, or between?

  (#P-hardness is not yet excluded by a subexponential algorithm!)

- What structural or "width" restriction is sufficient to efficiently compute the Tutte polynomial of an abstract matroid?

  (The polynomial is #P-hard over all matroids of branch-width three!)