Enclsoing the horse in singly-exponential time

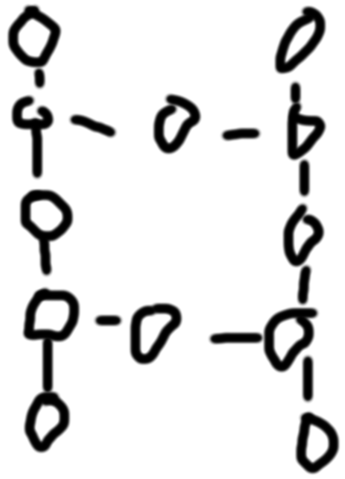
Consider the Steiner Tree problem:

Problem: **Steiner Tree**
Input: a graph G, a set of terminals $T \subseteq V(G)$, and an integer k.
Output: Does G contain a **connected** subset of size $\leq k$ that contains all of T?

Without the connectivity requirement, this problem can easily be solved by standard DP techniques (although it is also trivial: Just output `yes` if and only if $|T| \leq k$). When trying to formulate a treewidth-based DP for the full problem, however, we run into an issue: A naive formulation might look as follows:

$$A(x, f) = \text{min. size of solution } S \text{ in } V_x \text{ such that } B_x \cap S = f$$

with $n\, 2^{\text{tw}}$ many states. This does not quite work, however: consider a grid graph like the following:



Here, $A(\text{row } 3, \{(3,1),(3,3)\})$ tells us nothing about whether $(3,1)$ and $(3,3)$ are connected somewhere within $V_{\text{row } 3}$ (ie. through $(4,2)$) or outside (through $(2,2)$). Subsequently we will not be able to ensure that $(3,1)$ and $(3,3)$ are connected at all!

The "simple" way to work around this is to represent which subsets of B_x are connected in V_x directly in the DP state itself, ie.

$$A : (x : \text{tree decomposition bag}) \times (f : 2^{B_x}) \times \text{equiv. rel. on } f \to \mathbb{N}$$

$$\begin{aligned} A(x, f, c) = \text{min. size of solution } S \text{ in } V_x \text{ such that } B_x \cap S = f \\ \text{and } (u, v) \in c \text{ iff. there is a path } u - v \text{ in } V_x \cap S \end{aligned}$$

And while this works (and leads to a running time of the form $2^{O(\text{tw} \log \text{tw})} n$), the dependence on the treewidth is no longer singly exponential.

The Cut&Count technique [5] is a recent (2013) technique invented to solve problems with "connectivity constraints" such as the above in singly-exponential time.

Cut&Count

It will turn out that, instead of finding an exact answer, it will be sufficient to count the number of solutions (of a slightly modified problem) modulo 2. (proof of this comes later). As is usual whenever a DP gets too complicated, we will also take the solution size as a parameter k instead of computing the optimum directly.

Let R be the set of all "solutions" of the given size without the connectivity constraint (we will call these "potential solutions"), and S be the subset of potential solutions which are actually connected.

$$R = \{X \subseteq V(G) \mid T \subseteq X, |X| = k\}$$
$$S = \{X \in R \mid X \text{ is connected}\}$$

We would like to compute $|S| \bmod 2$, but this seems hard. Computing the parity of $|R|$ is easy (in fact computing the whole of $|R|$ is easy), but this will not be very useful as the parity of $|S|$ and $|R|$ may differ.[1]

The main idea of the Cut&Count technique is to replace the "global" connectivity constraint with a local constraint about graph cuts. More specifically, consider some potential solution X. Let a "consistent cut" of X be a cut (X_l, X_r) such that no edge in $G[X]$ crosses the cut. Equivalently, a consistent cut consists of an assignment of a side ("left" or "right") to each connected component of $G[X]$, and thus the number of consistent cuts is exactly $2^{\mathrm{cc}(G[X])}$, where $\mathrm{cc}(H)$ is the number of connected components in H.

This is almost what we want: if we somehow halved this number, we would get a number which is odd if $G[X]$ is connected and even otherwise. So let

$$C = \{(X, (X_l, X_r)) \mid X \in R, (X_l, X_r) \text{ consistent cut of } X\}$$

Then $|C| \equiv |S| \bmod 2$, since connected potential solutions[2] contribute by exactly 1 to $|C|$ all other potential solutions contribute by $0 \bmod 2$.

To halve this number, we may simply pick a vertex r we know is in the solution (ie. any vertex in T) and require that it be put on the left side of the cut.

To compute $|C|$ we may use standard DP techniques: Given a bag t, a subset $X \subset B_t$ and an assignment $s : A \to \{\mathrm{L}, \mathrm{R}\}$, let $c(t, A, s)$ be the number of pairs $(X, (X_l, X_r))$ in C except restricted to $G[V_t]$ such that $X \cap B_t = A$ and s gives the side of each vertex in A in the cut, requiring that $s(r) = \mathrm{L}$. Computing c in time $O^*\left(2^{O(k)}\right)$ should then be relatively straightforward.

The promised proof of why parity is enough

The following definition and lemma are taken from [5]:

Definition. A *function* $\omega : U \to \mathbb{Z}$ isolates a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$.

Lemma. *(Isolation Lemma, [6])* Let $\mathcal{F} \subseteq 2^U$ be a set family over a universe U with $|\mathcal{F}| > 0$. For each $u \in U$, choose a weight $\omega(u) \in \{1, 2, ..., N\}$ uniformly and independently at random. Then

$$\mathbb{P}[\omega \text{ isolates } \mathcal{F}] \geq 1 - \frac{|U|}{N}$$

The relevance of the Isolation Lemma is as follows: Let $U = V$ and $\mathcal{F}$ be the set of solutions. Then the isolation lemma implies that if we randomly weigh the vertices in the graph, whp. there will be some

[1] testing footnotes

[2] aka solutions

weight w such that exactly one solution has weight w. In particular, 1 is odd, so if we only look for solutions of weight w, if ω isolates $\mathcal{F}$, we will detect that we have an odd number of solutions and thus that a solution exists.

We get the following algorithm: let $N = 2|V|$, and uniformly and independently at random assign a weight in $\{1, \ldots, N\}$ to each vertex in the graph. Then, for each w from 1 to $N|V|$[3], compute the number of solutions of total weight $w \bmod 2$. If any of them is nonzero, return `True`, otherwise return `False`.

Obviously, if the algorithm returns `True`, there must be at least one solution. Otherwise, the probability that a solution exists is $\leq |V|/N = 1/2$, by the Isolation Lemma, as outlined above.

Bibliography

[5] M. Cygan, *Cut&Count technique for graph connectivity problems parameterized by treewidth*. 2011.

[6] 'Matching is as Easy as Matrix Inversion'.

[3] Or any upper bound on the maximum solution weight, for Steiner Tree $k|V|$ suffices[4]

[4] testing nested footnotes