Embedding graphs of small treewidth in $2^{\Theta(n/\log n)}$ time

Hans L. Bodlaender (Utrecht, Eindhoven)
Joint work with Johan M. M. van Rooij and Jesper Nederlof

August 27, 2015
This talk

1. Statement of the results
2. Upper bound example
3. Lower bound example
Many NP-hard problems have time complexity $2^\Theta(n)$, or, when restricted to classes like planar graphs $2^\Theta(\sqrt{n})$. Here we have problems that

- have time complexity $2^\Theta(n / \log n)$
- are NP-complete
- upper bound technique exploits isomorphism
- lower bound conditional on ETH with same intermediate problem
- are some kind of graph embedding problem for graphs of fixed treewidth
Problems with complexity $2^{\Theta(n / \log n)}$

Note: $k$ is a constant (really!! so, maybe 2, 3, 4, 39)

- Intervalizing Colored Graphs (ToCS 2015, with Johan van Rooij)
- Path (Tree) decompositions with minimum number of bags of width $\leq k$ (with Jesper Nederlof, ESA 2015)
- Subgraph Isomorphism (Induced subgraph isomorphism, minor testing, induced minor, immersion, . . . ) for graphs of bounded treewidth (work in progress)
For each of these problems, we have some fixed value where the problem is NP-complete for graphs of pathwidth $k$ (often not our work)

For each, there is an algorithm that solves it in $2^{O(c_k n/ \log n)}$ time

For each, there is a $k$, such that each algorithm for pathwidth $k$ graphs uses $2^{\Omega(n/ \log n)}$ time, assuming the ETH ($k = 2$ or $k = 3$ for the subgraph isomorphism variants; 6 colors for Intervalizing Colored Graphs; 39 for path or tree decompositions with few bags)
Path decompositions

Definition

A path decomposition of a graph $G = (V, E)$ is a sequence of subsets of $V$, $(X_1, \ldots, X_r)$ such that

- For all $v \in V$, $\{i \mid v \in X_i\}$ is a nonempty set of consecutive numbers
- For all $\{v, w\} \in E$: there is an $i$ with $v \in X_i$ and $w \in X_i$

The width of path decomposition $(X_1, \ldots, X_r)$ is $\max_{1 \leq i \leq r} |X_i| - 1$. The pathwidth of a graph $G$ is the minimum width of a path decomposition of $G$. 
Example problem for upper bound technique

**Minimum Length Path Decomposition** \( (k) \)

**Given:** Graph \( G \)

**Question:** What is the minimum number of bags of a path decomposition of width at most \( k \) of \( G \), if existing?

Note: NP-complete even for pathwidth 1
Recursive formulation

For vertex sets $W$, $X$, $X \subseteq W$, define

$MPW_k(W, X) =$ the minimum number of bags of a path decomposition of $G[W]$ of width at most $k$ such that the last bag equals $X$.

$MPW_k(W, X) = 1$ if $|W| \leq k + 1$ and $W = X$

$MPW_k(W, X) = \min MPW_k(W - (X - Y), Y) \mid X - Y \neq \emptyset \land N(X - Y) \subseteq X \land |Y| \leq k + 1\} + 1$ if $W \neq X$

- guess the one-but-last bag $Y$ and take the best over all cases
- $X - Y \neq \emptyset$ otherwise never optimal
- vertices in $X - Y$ only appear in the last bag

Leads here to $O(n^{O(k)}2^n) = O*(2^n)$ algorithm of a similar form as Held-Karp algorithm for TSP
Memorization

- DP with memorization: use the recursive formulation as a recursive program with an auxiliary datastructure
  - first check if subproblem is in datastructure
  - if so, return that answer
  - otherwise, compute answer with recursive formula
  - store the new computed answer with its subproblem in the datastructure
  - return the new computed answer

- Costs here $O^*(2^n)$ time

Next: improve with isomorphism
Memorization

- DP with memorization: use the recursive formulation as a recursive program with an auxiliary datastructure
- first check if a subproblem that is isomorphic is in datastructure
- if so, return that answer
- otherwise, compute answer with recursive formula
- store the new computed answer with its subproblem in the datastructure
- return the new computed answer

To do: time analysis

Isomorphic: \((W, X)\) is isomorphic to \((W', X')\) iff there is an isomorphism from \(G[W]\) to \(G[X']\) that maps \(X\) to \(X'\)

Note: isomorphic instances have the same answer
A recursive algorithm with memorization

- Initialize a datastructure $D$, initially empty
- For all sets $X$ with $k$ vertices each with different color, run $\text{Compute}(V.X)$
- If at least one of these returns true, answer is YES, otherwise NO

$\text{Compute}(W, X)$
- if $D(W, X)$ is stored then return $D(W, X)$.
- else
  - answer = false;
  - if $X = W$ then answer = true;
  - else for all $Y \subseteq W$ with ... 
    - if (For all $v \in X \setminus Y$: $N(v) \subseteq X$) 
      then answer = answer or $\text{Compute}(W \setminus (X \setminus Y), Y)$
  - store $D(W, X) = $ answer
- return answer
Isomorphisms and treewidth

- B, 1988: Graph Isomorphism is XP for bounded treewidth graphs
- Lokshtanov et al, 2013, Graph Isomorphism is FPT for bounded treewidth graphs. This works also for labeled versions, and also gives canonical representations

Corollary (Lokshtanov et al., 2013)

For each fixed $k$, there is an $O(n^6)$ algorithm, that gives a pair $(W, X)$ with $G[W]$ pathwidth at most $k$, computes a canonical representation $r(W, X)$ such that $(W, X)$ is isomorphic to $(W', X)$ if and only if $r(W, X) = r(W', X)$. 
Algorithm scheme

Compute\((W, X)\)
  Compute \(r(W, X)\)
  if \(D(r(W, X))\) is stored, then return \(D(r(W, X))\)
  else compute answer as before
    store \(D(r(W, X)) = \text{answer}\)
    return answer

Time is \(n^{O(k)}\) time the number of nonisomorphic pairs \((W, X)\)
Analyzing isomorphisms

- Consider some fixed $X$ with $\leq k + 1$ vertices. Consider the connected components of $G[V \setminus X]$. We can partition these into isomorphic classes, with $W_1$ in the same class as $W_2$, iff $(W_1, X)$ is isomorphic to $(W_2, X)$.

- For each such component with vertex set $Q$, if we consider a pair $(W, X)$ in our algorithm then $Q \subseteq W$ or $Q \cap W = \emptyset$.

- $(W, X)$ is isomorphic to $(W', X)$, if and only if for each isomorphism class of components, the number of components in this class that are a subset of $W$ equals the number of components that are a subset of $W'$.

- So, we can characterize (up to isomorphism) isomorphic classes of $(\ldots, X)$ by telling for each isomorphism class of components how many are ‘inside’
Counting

**Lemma**

For each $k$, there is a positive integer $c_k$ such that there are at most $2^{c_k n}$ nonisomorphic graphs of pathwidth at most $k$ with $n$ vertices.

Proof sketch: nice path decomposition with $O(n)$ bags gives $2^k + k + 1$ possibilities per bag up to isomorphism

**Corollary**

For each $k$, there is a positive integer $c'_k$, such that for each set $X$ with $\leq k + 1$ vertices, there are at most $2^{c'_k q}$ non-isomorphic components of $G[V \setminus X]$.

Proof sketch: previous bound multiplied by choices for adjacencies to $X$
Definition

A component is *large* if it has more than \( \frac{1}{2c'_k} \log n \) vertices

- Each large component is ‘inside’ or not: at most \( 2^{\frac{n}{2c'_k} \log n} \) possibilities for large components
- At most \( 2^{c'_k \frac{1}{2c'_k} \log n} = 2^{\log n/2} = \sqrt{n} \) nonisomorphic small components
- Each of these \( \leq \sqrt{n} \) classes has between 0 and \( n \) components ‘inside \( W \)’: \( n^{\sqrt{n}} \) possibilities for small components
- In total \( n^k \cdot 2^{O(\log n)} \cdot n^{\sqrt{n}} = 2^{O(n/\log n)} \) nonisomorphic pairs \((X, W)\) → gives bound on running time of our algorithm
Subgraph Isomorphism for graphs of pathwidth 2

Given: Graphs $G$, $H$ both with pathwidth at most 2.

Question: Is $G$ isomorphic to a subgraph of $H$?
Theorem

Suppose the Exponential Time Hypothesis holds. Then each algorithm for Subgraph Isomorphism for Graphs of Pathwidth 2 uses at least $2^\Omega(n/\log n)$ time.

Route:

- Exact 3-Satisfiability (sparse: $O(n)$ clauses)
- Partition into triangles of 3-colorable 4-regular graphs
- 3-Dimensional Matching with each value in $O(1)$ triples
- Bitstring grouping problem with $O(\log n)$ length strings
- Our problems . . .
• Hypothesis by Impagliazzo and Paturi (1999)
• Nowadays used as hypothesis in many complexity results to show lower bounds
• States: $3$-SATISFIABILITY cannot be solved in subexponential time — actually the statement is slightly stronger:

**Conjecture (The Exponential Time Hypothesis)**

Let $s_3$ be the infimum of the real numbers $\delta$ such that there is an algorithm that solves $3$-SATISFIABILITY in $2^{\delta n}$ time. Then $s_3 > 0$. 
Given are 3 sets of $n$ strings with $O(\log n)$ bits each. Can we group these such that

- Each group has one string from $A$, one from $B$, and one from $C$
- Each string is in one group
- Each group has no two 1s in the same coordinate (so 1001 can go with 0100 and 0010; we cannot have 1000 with 1100 as both have a 1 on the first coordinate)

This problem cannot be solved in $2^{o(n)}$ time unless the ETH fails (proof in steps via sparse 3-DM)
On the board: construction of graphs $G$ and $H$ with $O(n \log n)$ vertices and pathwidth 2
Wrapping up lower bound

- Under ETH, each algorithm for String Grouping with $3n$ strings of $O(\log n)$ bits uses $c^n$ time for some $c > 1$.
- We build graphs with $O(n \log n)$ vertices and pathwidth 2
- So we have:

**Theorem**

*Assume the Exponential Time Hypothesis holds. Then each algorithm for Subgraph Isomorphism for Graphs of Pathwidth 2 uses $2^{\Omega(n/\log n)}$ time.*
Conclusions

- Work in progress for different notions of ‘embedding’
- Small changes in algorithmic technique for treewidth/tree decompositions
- Open: *Graph Covering* (XP? FPT? Lower bound?)
- Other applications?