Tractable hypergraph properties for constraint satisfaction and conjunctive queries

Dániel Marx

Tel Aviv University, Israel

May 16, 2010

Foundations of Computer Science Seminar

Weizmann Institute, Israel

Overview

- Main question: How does the graph/hypergraph describing the structure of a constraint satisfaction instance influence the complexity?
- Already well-understood if every constraint is **binary** (i.e., involves 2 variables): complexity is closely connected with treewidth.
- New result: understanding it in the case of **arbitrary arities.** Large arity constraints can make the problem very different.

Constraint Satisfaction Problems (CSP)

A CSP instance is given by describing the

- variables,
- domain of the variables,
- constraints on the variables.

Task: Find an assignment that satisfies every constraint.

$$I = C_1(x_1, x_2, x_3) \wedge C_2(x_2, x_4) \wedge C_3(x_1, x_3, x_4)$$

Constraint Satisfaction Problems (CSP)

A CSP instance is given by describing the

- variables,
- domain of the variables,
- constraints on the variables.

Task: Find an assignment that satisfies every constraint.

$$I = C_1(x_1, x_2, x_3) \wedge C_2(x_2, x_4) \wedge C_3(x_1, x_3, x_4)$$

Examples:

- 3SAT: 2-element domain, every constraint is ternary
- Vertex Coloring: domain is the set of colors, binary constraints
- k-Clique (in graph G): k variables, domain is the vertices of G, $\binom{k}{2}$ binary constraints

Conjunctive Queries

A **relational database** contains some number of **relational tables:**

$$\text{enrolled}(\text{Person}, \text{Course}, \text{Date})$$
$$\text{teaches}(\text{Person}, \text{Course}, \text{Year})$$
$$\text{parent}(\text{Person1}, \text{Person2})$$

A **conjunctive query** creates a new table by joining and projecting relations:

$$Q : \text{ans}(P) \leftarrow \text{enrolled}(P, C, D) \wedge \text{teaches}(P2, C, Y) \wedge \text{parent}(P2, P).$$

Conjunctive Queries

A **relational database** contains some number of **relational tables:**

$$\text{enrolled}(\text{Person}, \text{Course}, \text{Date})$$
$$\text{teaches}(\text{Person}, \text{Course}, \text{Year})$$
$$\text{parent}(\text{Person1}, \text{Person2})$$

A **conjunctive query** creates a new table by joining and projecting relations:

$$Q : \text{ans}(P) \leftarrow \text{enrolled}(P, C, D) \wedge \text{teaches}(P2, C, Y) \wedge \text{parent}(P2, P).$$

The **Boolean Conjunctive Query Evaluation** problem asks if the answer relation is empty or not.

$\Rightarrow$ This is a CSP problem!

Note: For such CSPs, the number of variables is typically small, while the domain size is large.

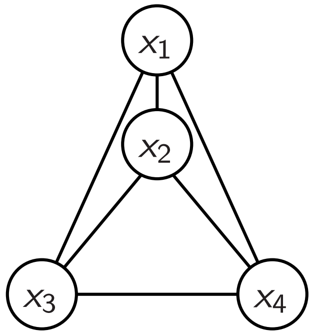
Graphs and hypergraphs related to CSP

Gaifman/primal graph: vertices are the variables, two variables are adjacent if they appear in a common constraint.

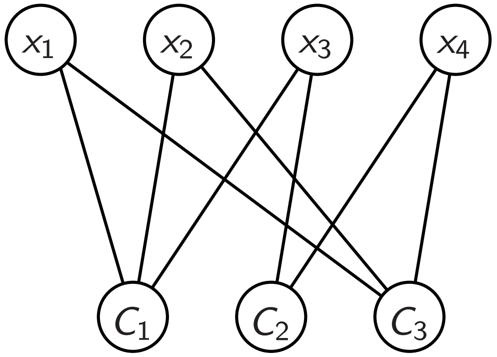
Incidence graph: bipartite graph, vertices are the variables and constraints.

Hypergraph: vertices are the variables, constraints are the hyperedges.

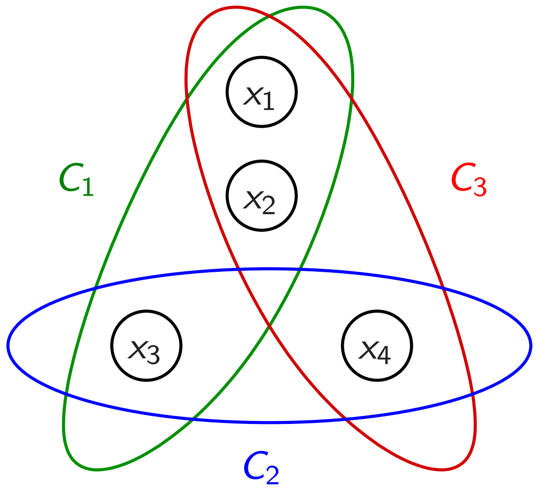
$$I = C_1(x_2, x_1, x_3) \wedge C_2(x_4, x_3) \wedge C_3(x_1, x_4, x_2)$$



Primal graph



Incidence graph



Hypergraph

CSP and hypergraphs

Big question:

What are the easy hypergraphs?

CSP and hypergraphs

Big question:

What are the easy hypergraphs?

Trivial answer: For every fixed hypergraph, the problem can be solved in polynomial time (every hypergraph has a constant number of vertices).

CSP and hypergraphs

Big question:

What are the easy hypergraphs?

Trivial answer: For every fixed hypergraph, the problem can be solved in polynomial time (every hypergraph has a constant number of vertices).

More precisely:

What are the easy **classes** of hypergraphs?

CSP and hypergraphs

Big question:

What are the easy hypergraphs?

Trivial answer: For every fixed hypergraph, the problem can be solved in polynomial time (every hypergraph has a constant number of vertices).

More precisely:

What are the easy **classes** of hypergraphs?

Even more precisely:

CSP($\mathcal{H}$): restriction of the problem such that the hypergraph is from the class $\mathcal{H}$.

What are the classes $\mathcal{H}$ that make CSP($\mathcal{H}$) polynomial-time solvable?

Representation of constraints

How are the constraints represented in the input?

- full truth table
- listing the satisfying tuples
- formula/circuit
- oracle

Representation of constraints

How are the constraints represented in the input?

- full truth table
- listing the satisfying tuples
- formula/circuit
- oracle

If the arity of every constraint is bounded by a constant, then the representations are polynomially equivalent, but if there is no bound there can be exponential difference between different representations.

The choice of representation changes the length of the input, thus can change the complexity of the problem.

Representation of constraints

How are the constraints represented in the input?

- full truth table
- listing the satisfying tuples
- formula/circuit
- oracle

In this talk: Each constraint is given by listing all the tuples that satisfy it.

Motivation: Applications in database theory (Conjunctive Query Evaluation, Conjunctive Query Containment)

Constraints are known databases, “satisfying” means “appears in the database.”

Tractability

Definition: CSP($\mathcal{H}$) is **polynomial-time solvable** if there is an algorithm solving every instance I of CSP($\mathcal{H}$) in time $\|I\|^c$ for some constant c.

Definition: CSP($\mathcal{H}$) is **fixed-parameter tractable (FPT)** if there is an algorithm solving every instance I of CSP($\mathcal{H}$) in time $f(H)\|I\|^c$ for some function f depending only on the hypergraph H and a constant c.

Note: The definition does not change if we replace $f(H)$ with a function $f'(k)$ depending on the number of variables.

Goal: Characterize those classes $\mathcal{H}$ for which CSP($\mathcal{H}$) is polynomial-time solvable/fixed-parameter tractable.

Some positive results

CSP($\mathcal{H}$) is known to be polynomial-time solvable...

- ... if $\mathcal{H}$ has bounded treewidth (a result in the AI literature).
- ... if $\mathcal{H}$ contains "acyclic" hypergraphs (a result from the database literature, late 70s).
- ... if $\mathcal{H}$ has bounded query width/hypertree width/fractional edge cover number/fractional hypertree width (1997–2009)

Can we prove more general positive results?

Can we prove negative results?

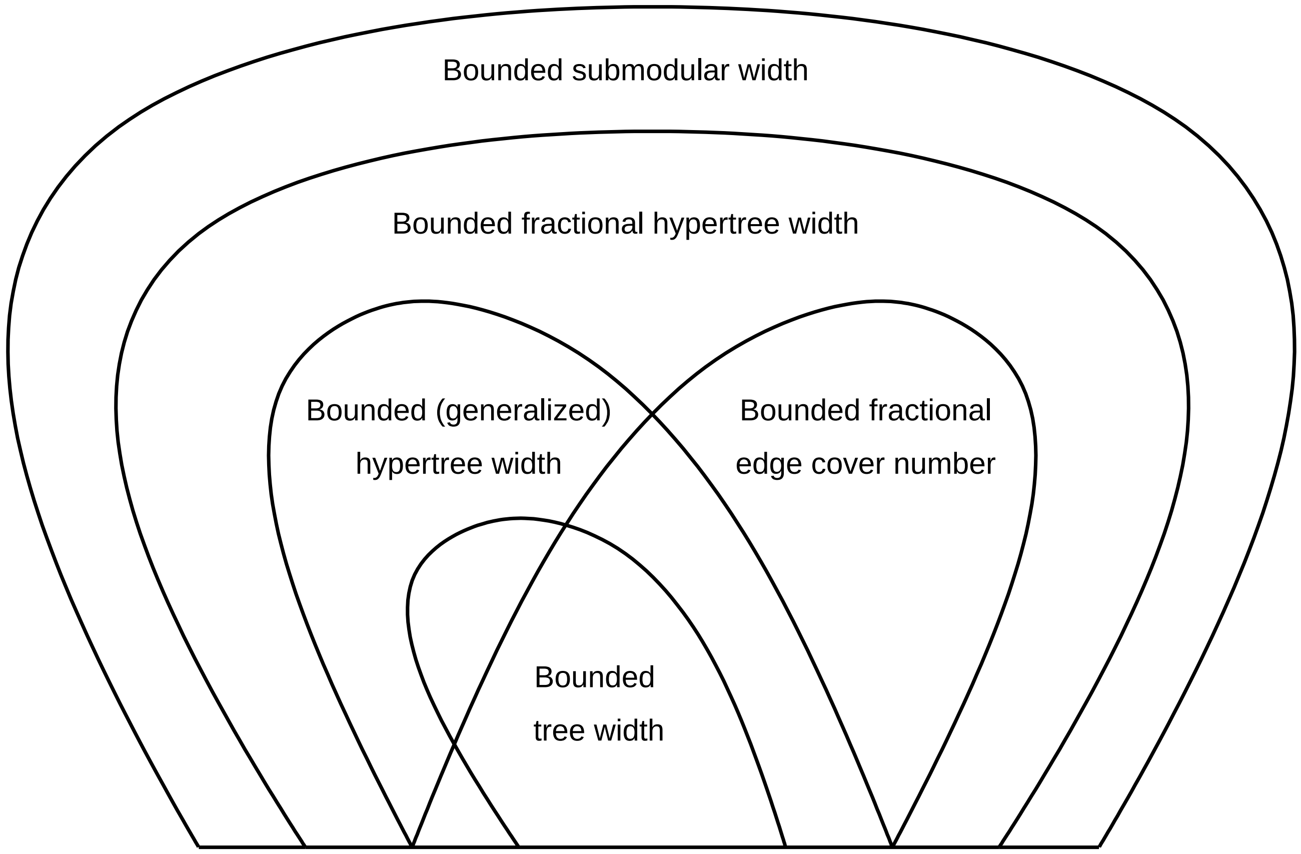
Main result

Main result: [M. 2010] Let $\mathcal{H}$ be a recursively enumerable set of hypergraphs. Assuming ETH,

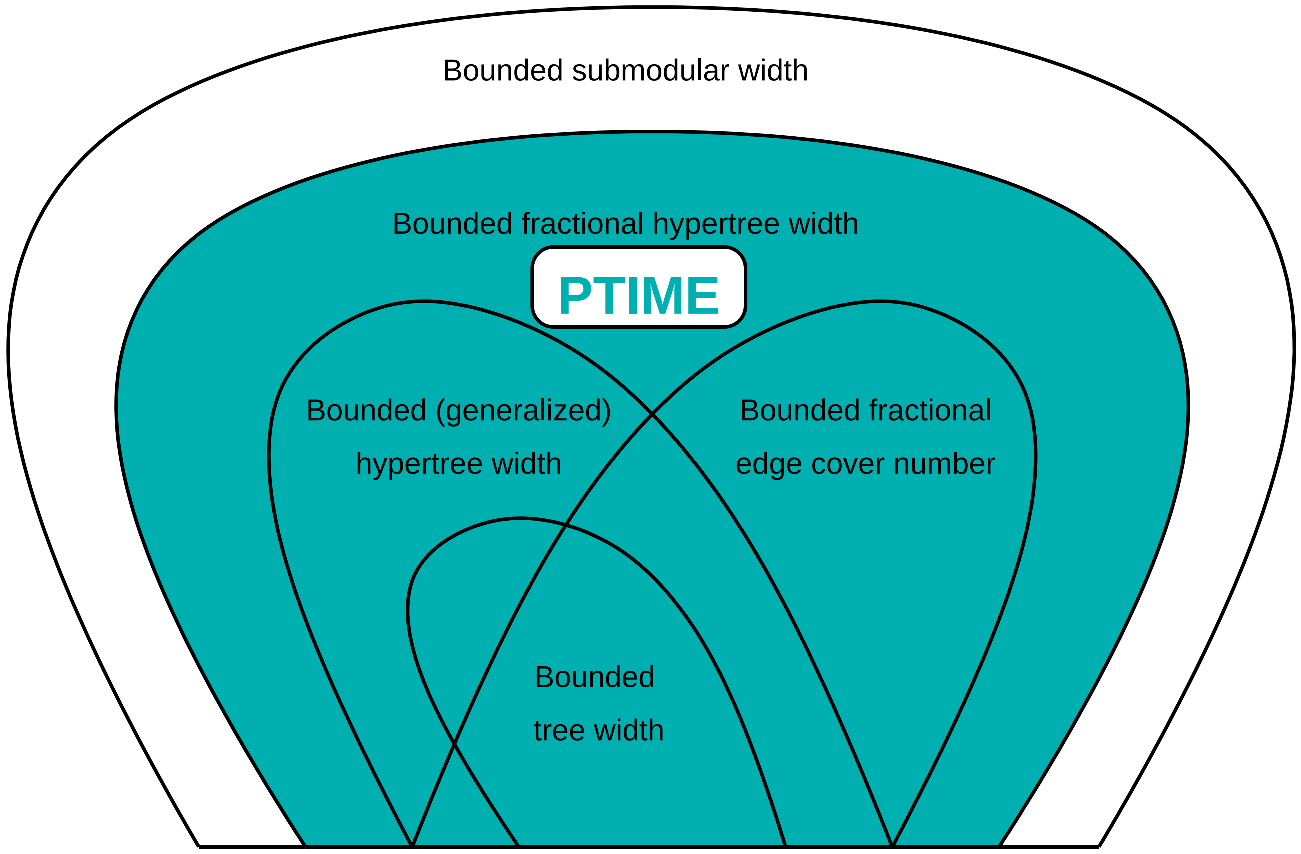
$$\text{CSP}(\mathcal{H}) \text{ is FPT} \iff \mathcal{H} \text{ has bounded submodular width.}$$

Exponential Time Hypothesis (ETH): There is no $2^{o(n)}$ time algorithm for n-variable 3SAT (known to be equivalent with “There is no $2^{o(m)}$ time algorithm for m-clause 3SAT”).

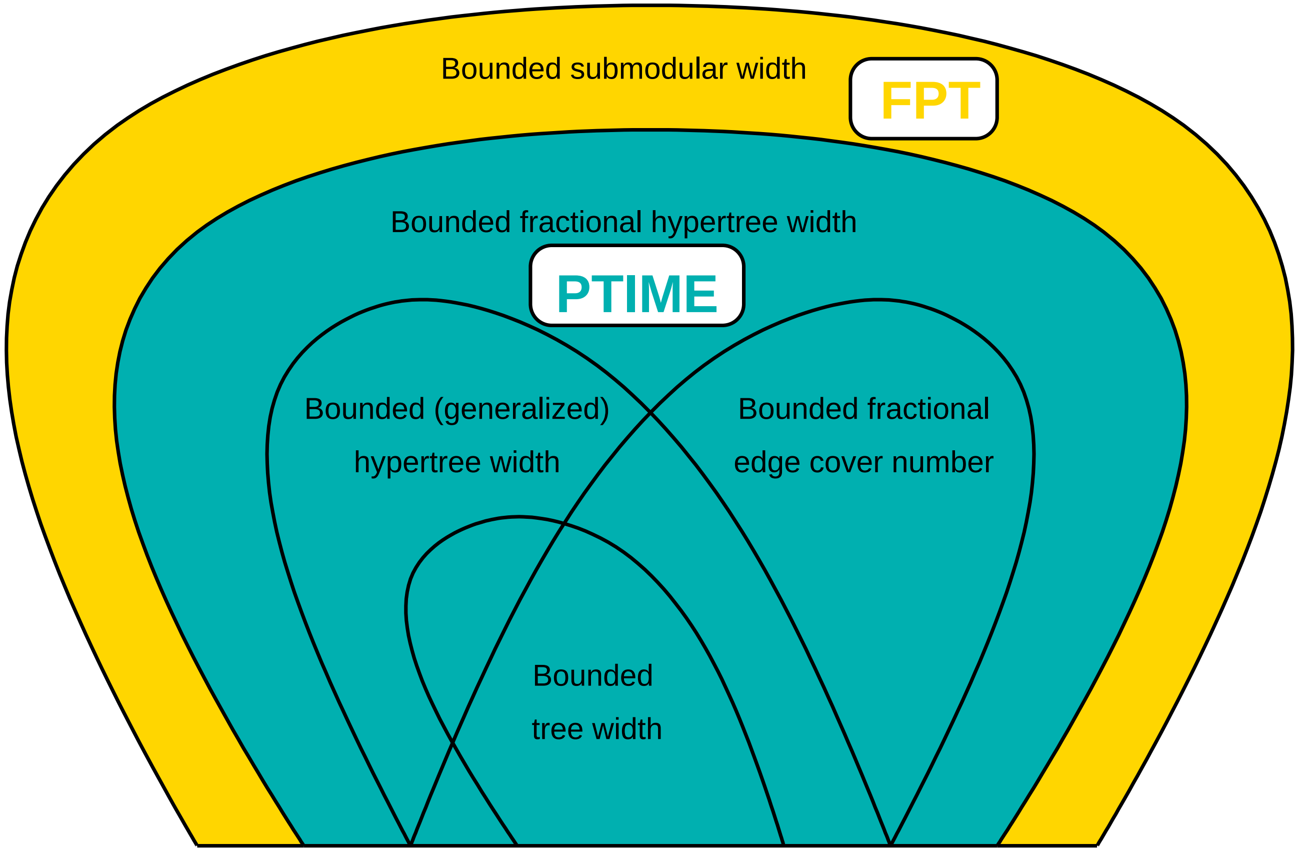
Tractable classes


Bounded submodular width
Bounded fractional hypertree width
Bounded (generalized) hypertree width
Bounded fractional edge cover number
Bounded tree width

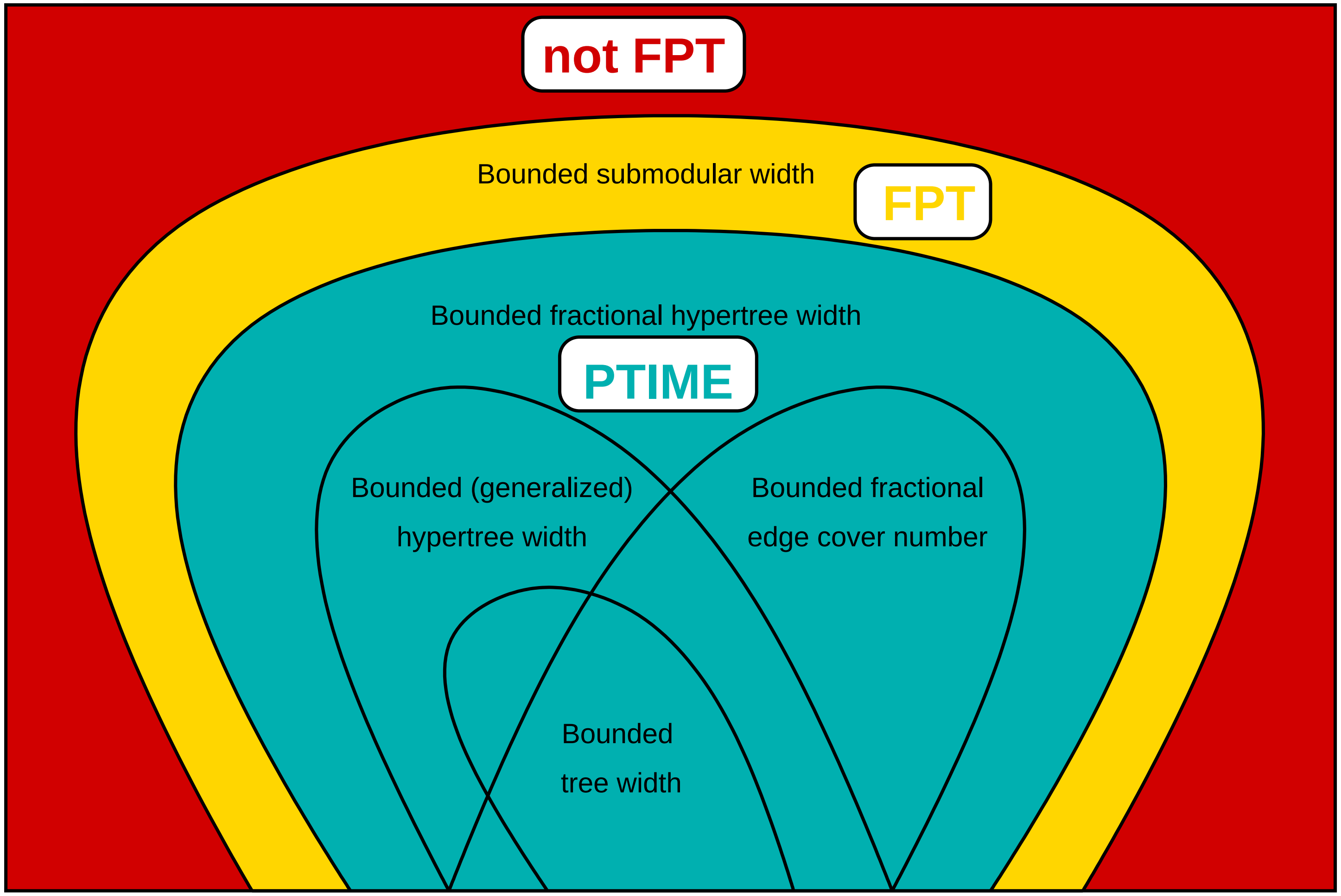
Tractable classes


Bounded submodular width
Bounded fractional hypertree width
PTIME
Bounded (generalized) hypertree width
Bounded fractional edge cover number
Bounded tree width

Tractable classes


Bounded submodular width
FPT
Bounded fractional hypertree width
PTIME
Bounded (generalized) hypertree width
Bounded fractional edge cover number
Bounded tree width

Tractable classes


not FPT
Bounded submodular width
FPT
Bounded fractional hypertree width
PTIME
Bounded (generalized) hypertree width
Bounded fractional edge cover number
Bounded tree width

Overview

Next:

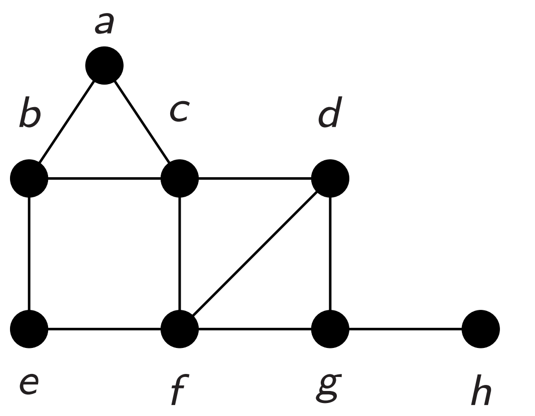
- Brief review of treewidth.
- Previous results for the special case of binary CSP: each constraint involves two variables, i.e., $\mathcal{H}$ is a class of **graphs.**
- General case: $\mathcal{H}$ is a class of **hypergraphs.**

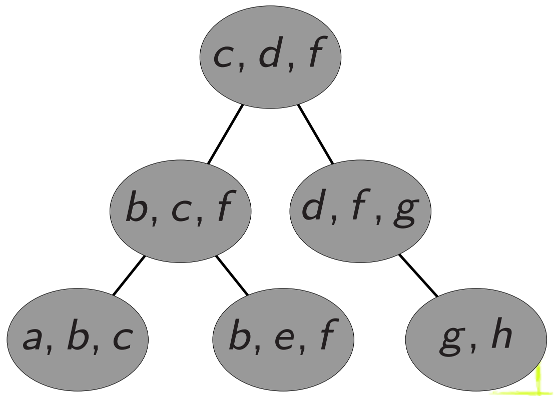
Treewidth

Treewidth: A measure of how "tree-like" the graph is. (Introduced by Robertson and Seymour.)

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.



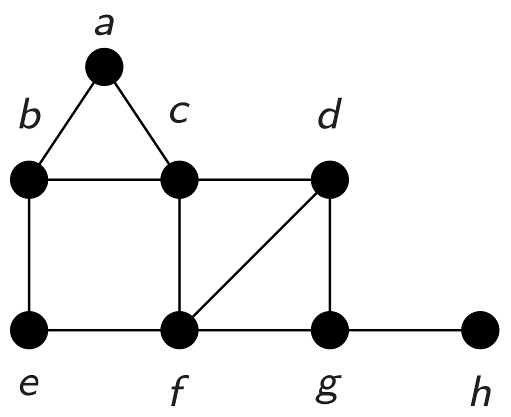


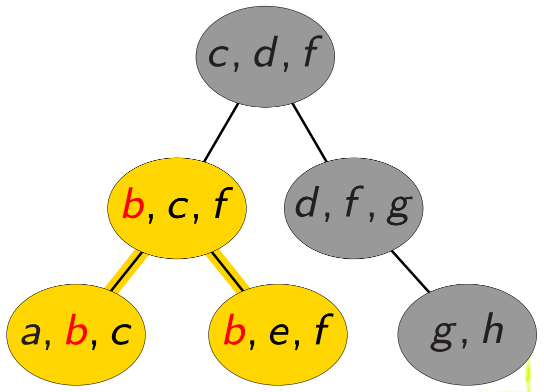
Treewidth

Treewidth: A measure of how “tree-like” the graph is.
(Introduced by Robertson and Seymour.)

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.



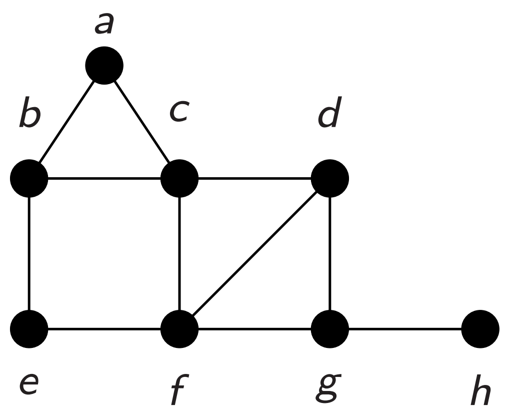


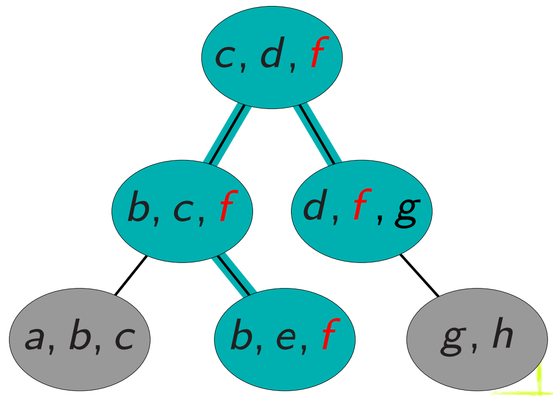
Treewidth

Treewidth: A measure of how “tree-like” the graph is. (Introduced by Robertson and Seymour.)

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.



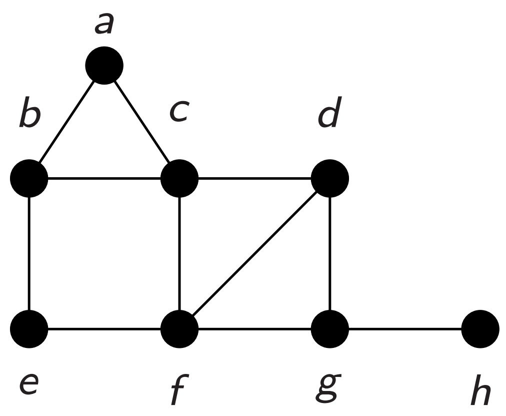


Treewidth

Treewidth: A measure of how "tree-like" the graph is. (Introduced by Robertson and Seymour.)

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

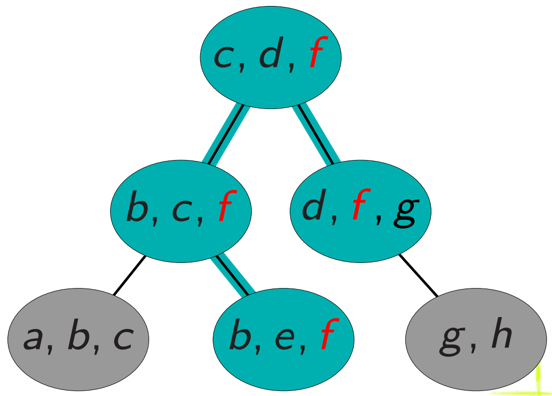
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.



Width of the decomposition: size of the largest bag minus 1.

treewidth: width of the best decomposition.

Fact: treewidth $= 1 \iff$ graph is a forest



Treewidth

Treewidth: A measure of how “tree-like” the graph is. (Introduced by Robertson and Seymour.)

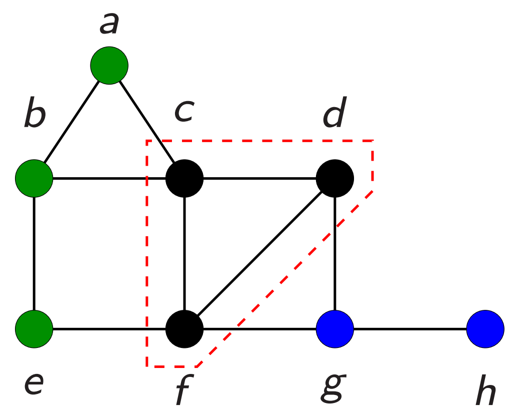
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

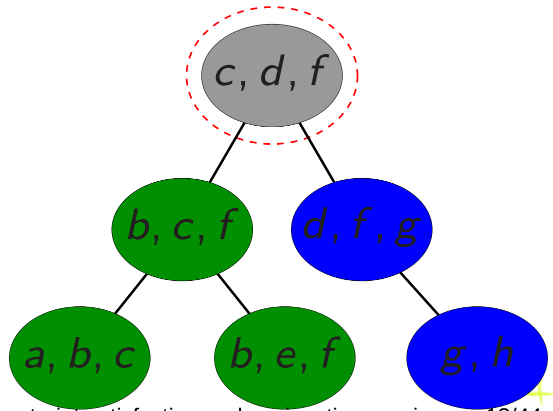
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.

Width of the decomposition: size of the largest bag minus 1.

treewidth: width of the best decomposition.

Fact: treewidth $= 1 \iff$ graph is a forest



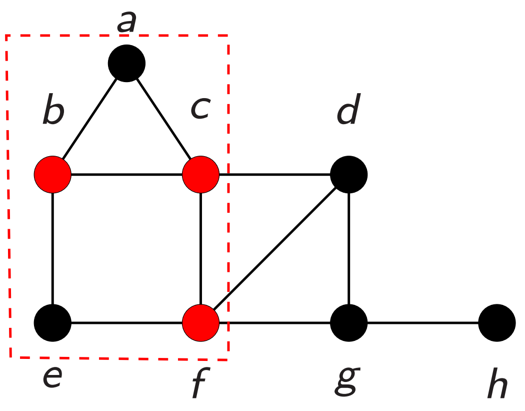


Treewidth

Treewidth: A measure of how “tree-like” the graph is. (Introduced by Robertson and Seymour.)

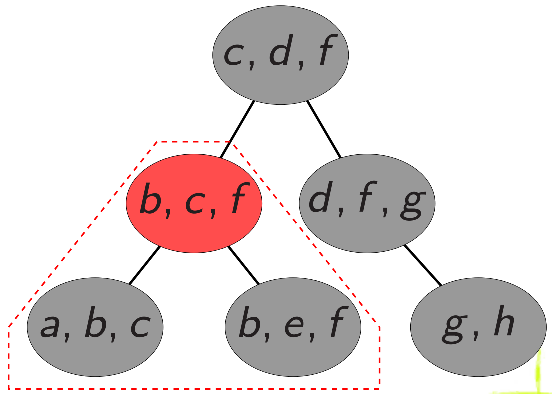
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.



Width of the decomposition: size of the largest bag minus 1.

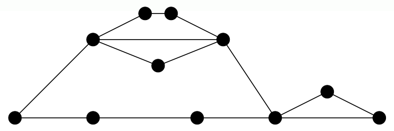
treewidth: width of the best decomposition.



Fact: treewidth $= 1 \iff$ graph is a forest

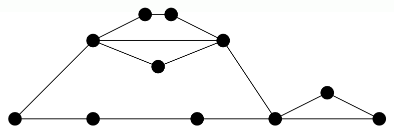
Properties of treewidth

Fact: treewidth ≤ 2 $\iff$ graph is subgraph of a series-parallel graph

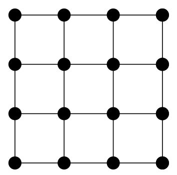


Properties of treewidth

Fact: treewidth $\leq 2 \iff$ graph is subgraph of a series-parallel graph

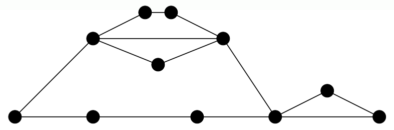


Fact: For every $k \geq 2$, the treewidth of the $k \times k$ grid is exactly k.

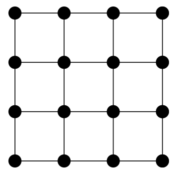


Properties of treewidth

Fact: treewidth $\leq 2 \iff$ graph is subgraph of a series-parallel graph



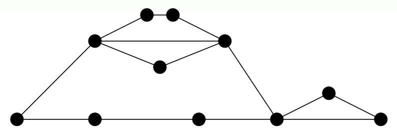
Fact: For every $k \geq 2$, the treewidth of the $k \times k$ grid is exactly k.



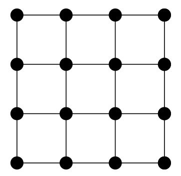
Fact: Treewidth does not increase if we delete edges/vertices, or contract edges.
$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Properties of treewidth

Fact: treewidth $\leq 2 \iff$ graph is subgraph of a series-parallel graph



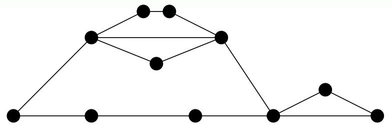
Fact: For every $k \geq 2$, the treewidth of the $k \times k$ grid is exactly k.



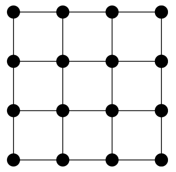
Fact: Treewidth does not increase if we delete edges/vertices, or contract edges.
$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Fact: [Excluded Grid Theorem] If the treewidth of G is at least $k^{4k^2(k+2)}$, then G has a $k \times k$ grid minor.

Properties of treewidth



Fact: treewidth $\leq 2 \iff$ graph is subgraph of a series-parallel graph



Fact: For every $k \geq 2$, the treewidth of the $k \times k$ grid is exactly k.

Fact: Treewidth does not increase if we delete edges/vertices, or contract edges.
$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Fact: [Excluded Grid Theorem] If the treewidth of G is at least $k^{4k^2(k+2)}$, then G has a $k \times k$ grid minor.

Fact: For every clique K, there is a bag B with $K \subseteq B$
$\Rightarrow$ In the primal graph of a CSP instance, the scope of each constraint is a clique, hence it is fully contained in a bag.

Treewidth and CSP

Fact: For every fixed k, CSP can be solved in polynomial time if the primal graph of the instance has treewidth at most k.

Proof sketch:

- A tree decomposition of width k can be found in linear time for every fixed k.
- For each bag, enumerate every assignment of the bag that satisfies every constraint fully contained in the bag. Each bag has at most $k+1$ variables, thus there are at most $|D|^{k+1}$ such assignments for each bag.
- Use bottom-up dynamic programming to find a satisfying assignment.
- Each constraint induces a clique in the primal graph, thus each constraint is fully contained in one of the bags.
- Running time of DP is polynomial in $|D|^{k+1}$ and the number of variables.

Dichotomy for binary CSP

We know that CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Dichotomy for binary CSP

We know that CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Theorem: [Grohe-Schwentick-Segoufin 2001] Let $\mathcal{G}$ be a recursively enumerable class of **graphs.** Assuming FPT $\neq$ W[1], the following are equivalent:

- Binary CSP($\mathcal{G}$) is polynomial-time solvable.
- Binary CSP($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

Dichotomy for binary CSP

We know that CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Theorem: [Grohe-Schwentick-Segoufin 2001] Let $\mathcal{G}$ be a recursively enumerable class of **graphs.** Assuming FPT $\neq$ W[1], the following are equivalent:

- Binary CSP($\mathcal{G}$) is polynomial-time solvable.
- Binary CSP($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

Note: FPT $\neq$ W[1] is the standard assumption of parameterized complexity.

Note: Fixed-parameter tractability does not give us more power here than polynomial-time solvability.

Note: We cannot hope a P vs. NP-complete dichotomy: there are classes $\mathcal{G}$ for which the problem is equivalent to LOGCLIQUE.

Proof outline

Suppose that $\mathcal{G}$ has unbounded treewidth, but CSP($\mathcal{G}$) is FPT.

- Assuming FPT $\neq$ W[1], there is no $f(k)n^c$ time algorithm for k-CLIQUE. But we can solve k-CLIQUE the following way:
- Formulate k-CLIQUE as a binary CSP instance on the $k \times k$ grid.
- Find a $G_k \in \mathcal{G}$ containing a $k \times k$ minor (there is such a G_k by the Excluded Grid Theorem).
- Reduce CSP on the $k \times k$ grid to CSP with graph G_k, which is an instance of CSP($\mathcal{G}$).
- Use the assumed algorithm for CSP($\mathcal{G}$).
- The running time is $f(k)n^c$: the nonpolynomial factors in the running time depend only on k (finding G_k, size of G_k, solving CSP($\mathcal{G}$))
  $\Rightarrow$ k-CLIQUE is FPT, contradicting the hypothesis FPT $\neq$ W[1].

Can you beat treewidth?

If $\mathcal{G}$ has unbounded treewidth, then there is no polynomial algorithm for binary CSP($\mathcal{G}$), but it can be solved in time $\|I\|^{O(k)}$, where k is the treewidth of the primal graph.

Is there a class $\mathcal{G}$ where we can do much better, for example, there is a $\|I\|^{O(\sqrt{k})}$ or even $\|I\|^{O(\log\log\log k)}$ algorithm for CSP($\mathcal{G}$)?

Can you beat treewidth?

If $\mathcal{G}$ has unbounded treewidth, then there is no polynomial algorithm for binary CSP($\mathcal{G}$), but it can be solved in time $\|I\|^{O(k)}$, where k is the treewidth of the primal graph.

Is there a class $\mathcal{G}$ where we can do much better, for example, there is a $\|I\|^{O(\sqrt{k})}$ or even $\|I\|^{O(\log\log\log k)}$ algorithm for CSP($\mathcal{G}$)?

Theorem: [M. 2007] If $\mathcal{G}$ is a recursively enumerable class of graphs such that binary CSP($\mathcal{G}$) can be solved in time $f(G) \cdot \|I\|^{o(k/\log k)}$ (where G is the primal graph and $k = \text{tw}(G)$), then the Exponential Time Hypothesis fails.

Embeddings

The previous proof is based on embedding the k-CLIQUE problem into a CSP instance using the grid whose existence is guaranteed by the Excluded Grid Theorem. However, this theorem is very weak: a $k \times k$ grid minor exists, if treewidth is exponentially large in k.

Embeddings

The previous proof is based on embedding the k-CLIQUE problem into a CSP instance using the grid whose existence is guaranteed by the Excluded Grid Theorem. However, this theorem is very weak: a $k \times k$ grid minor exists, if treewidth is exponentially large in k.

Definition: A **q-embedding** ϕ of graph F in graph G maps a subset of $V(G)$ to each vertex of F such that

- For every $v \in V(F)$, $\phi(v)$ is connected.
- If $u, v \in V(F)$ are adjacent in F, then $\phi(u)$ and $\phi(v)$ touch: either they intersect or there is an edge connecting them.
- Every $w \in V(G)$ appears in the images of at most q vertices of F.

Note: F is a minor of $G \iff$ there is a 1-embedding from F to G.

An embedding result

Lemma: [M. 2007] If $m = |E(F)|$ is sufficiently large and $k = \mathrm{tw}(G)$, then there is a q-embedding of F in G for $q = O(m \log k/k)$.

Note: A q-embedding for $q = O(m)$ is trivial, thus treewidth k means that we can gain a factor of $\Omega(k/\log k)$ compared to the trivial embedding.

An embedding result

Lemma: [M. 2007] If $m = |E(F)|$ is sufficiently large and $k = \text{tw}(G)$, then there is a q-embedding of F in G for $q = O(m \log k/k)$.

Note: A q-embedding for $q = O(m)$ is trivial, thus treewidth k means that we can gain a factor of $\Omega(k/\log k)$ compared to the trivial embedding.

Main ingredients of the proof:

- if treewidth is large, there is a set having no balanced separator (well-known),
- if there is a set having no balanced separator, then there is set having no sparse cut (well-known),
- $O(\log k)$ integrality gap between sparsest cut and concurrent multicommodity flows (well-known),
- the q-embedding is constructed using the paths appearing in the flows (new).

Proof

Theorem: [M. 2007] If $\mathcal{G}$ is a recursively enumerable class of graphs such that binary CSP($\mathcal{G}$) can be solved in time $f(G) \cdot \|I\|^{o(k/\log k)}$ (where G is the primal graph and $k = \text{tw}(G)$), then the Exponential Time Hypothesis fails.

Proof outline:

- Given a 3SAT instance with m clauses and n variables, we turn it into a CSP instance I_1 with $3m$ binary constraints.
- We use the embedding result to find a q-embedding of the primal graph of I_1 into some $G_k \in \mathcal{G}$ (chosen appropriately).
- We simulate I_1 by an instance I_2 whose primal graph is G_k: each variable of I_2 simulates at most q variables of I_1.
- Now the 3SAT problem can be solved by solving I_2. Calculation of the running time shows that that a “too fast” algorithm for CSP($\mathcal{G}$) would give a $2^{o(m)}$ algorithm for m-clause 3SAT, violating ETH.

Back to hypergraphs…

Tree decomposition: Bags of vertices are arranged in a tree structure satisfying the following properties:

1. For every hyperedge e, there is a bag containing the vertices of e.
2. For every vertex v, the bags containing v form a connected subtree.

Standard definitions:

Width of the decomposition: size of the largest bag minus 1.

Tree width: width of the best decomposition.

Equivalent: Treewidth of the hypergraph H is the treewidth of the graph underlying H (where every edge is a clique).

CSP and treewidth of hypergraphs

Unbounded treewidth does not imply hardness for CSP($\mathcal{H}$).

Example: Let $\mathcal{H}$ contain every hypergraph having only one edge.

- $\mathcal{H}$ has unbounded treewidth.
- CSP($\mathcal{H}$) is trivial to solve.

CSP and treewidth of hypergraphs

Unbounded treewidth does not imply hardness for CSP($\mathcal{H}$).

Example: Let $\mathcal{H}$ contain every hypergraph having only one edge.

- $\mathcal{H}$ has unbounded treewidth.
- CSP($\mathcal{H}$) is trivial to solve.

Example: Same for the class $\mathcal{H}$ containing those hypergraphs that can be covered by one edge.

Unlike in the binary case, the complexity is not monotone: adding hyperedges can make the problem easier!

Task: generalize treewidth in a way that takes this “non-monotonicity” into account.

Width measures for decompositions

Definition: Let $f : 2^{V(H)} \longrightarrow \mathbb{R}^+$ be a function assigning values to the vertex subsets of H.

- The f-width($\mathcal{T}$) of a tree decomposition $\mathcal{T}$ is the maximum of $f(B)$ over all bags B.
- The f-width(H) of hypergraph H is the minimum of f-width($\mathcal{T}$) over all tree decompositions $\mathcal{T}$ of H.

Width measures for decompositions

Definition: Let $f : 2^{V(H)} \to \mathbb{R}^+$ be a function assigning values to the vertex subsets of H.

- The f-width($\mathcal{T}$) of a tree decomposition $\mathcal{T}$ is the maximum of $f(B)$ over all bags B.
- The f-width(H) of hypergraph H is the minimum of f-width($\mathcal{T}$) over all tree decompositions $\mathcal{T}$ of H.

Previous results: We can define treewidth, query width, hypertree width, fractional hypertree width this way (details omitted).

But we take a different approach here...

Width measures for decompositions

Definition: Let $f : 2^{V(H)} \rightarrow \mathbb{R}^+$ be a function assigning values to the vertex subsets of H.

- The f-width($\mathcal{T}$) of a tree decomposition $\mathcal{T}$ is the maximum of $f(B)$ over all bags B.
- The f-width(H) of hypergraph H is the minimum of f-width($\mathcal{T}$) over all tree decompositions $\mathcal{T}$ of H.

Definition: Let $\mathcal{F}$ be a set of functions from $2^{V(H)}$ to $\mathbb{R}^+$. The $\mathcal{F}$-width of H is the maximum of f-width(H) over every $f \in \mathcal{F}$.

$\mathcal{F}$-width(H) $\leq w$ $\iff$ **for every** $f \in \mathcal{F}$ **exists** a tree decomposition $\mathcal{T}$ of $\mathcal{F}$ such that **for every** bag B of $\mathcal{T}$, $f(B) \leq w$.

Note: the tree decomposition $\mathcal{T}$ can be different for different functions $f \in \mathcal{F}$.

Submodular width

Definition: The **submodular width** of H is $\mathcal{F}$-width(H), where $\mathcal{F}$ is the set of all monotone, edge-dominated, submodular functions on the vertices of H.

Monotone: $b(X) \leq b(Y)$ for every $X \subseteq Y$.

Edge-dominated: $b(e) \leq 1$ for every hyperedge e of H.

Submodular: For arbitrary sets X, Y

$$b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y).$$

Main result

Main result: [M. 2010] Let $\mathcal{H}$ be a recursively enumerable set of hypergraphs. Assuming ETH,

$$\text{CSP}(\mathcal{H}) \text{ is FPT} \iff \mathcal{H} \text{ has bounded submodular width.}$$

- **Algorithmic side:** If $\mathcal{H}$ has bounded submodular width, then CSP($\mathcal{H}$) is FPT.

  How does it help if we know that every submodular function has a good tree decomposition?

- **Hardness:** If $\mathcal{H}$ has bounded submodular width, then CSP($\mathcal{H}$) is not FPT.

  To simulate 3SAT by CSP($\mathcal{H}$), we need an efficient embedding of a graph into a hypergraph. We know that certain submodular functions do not have good tree decompositions. How does that help in finding good embeddings?

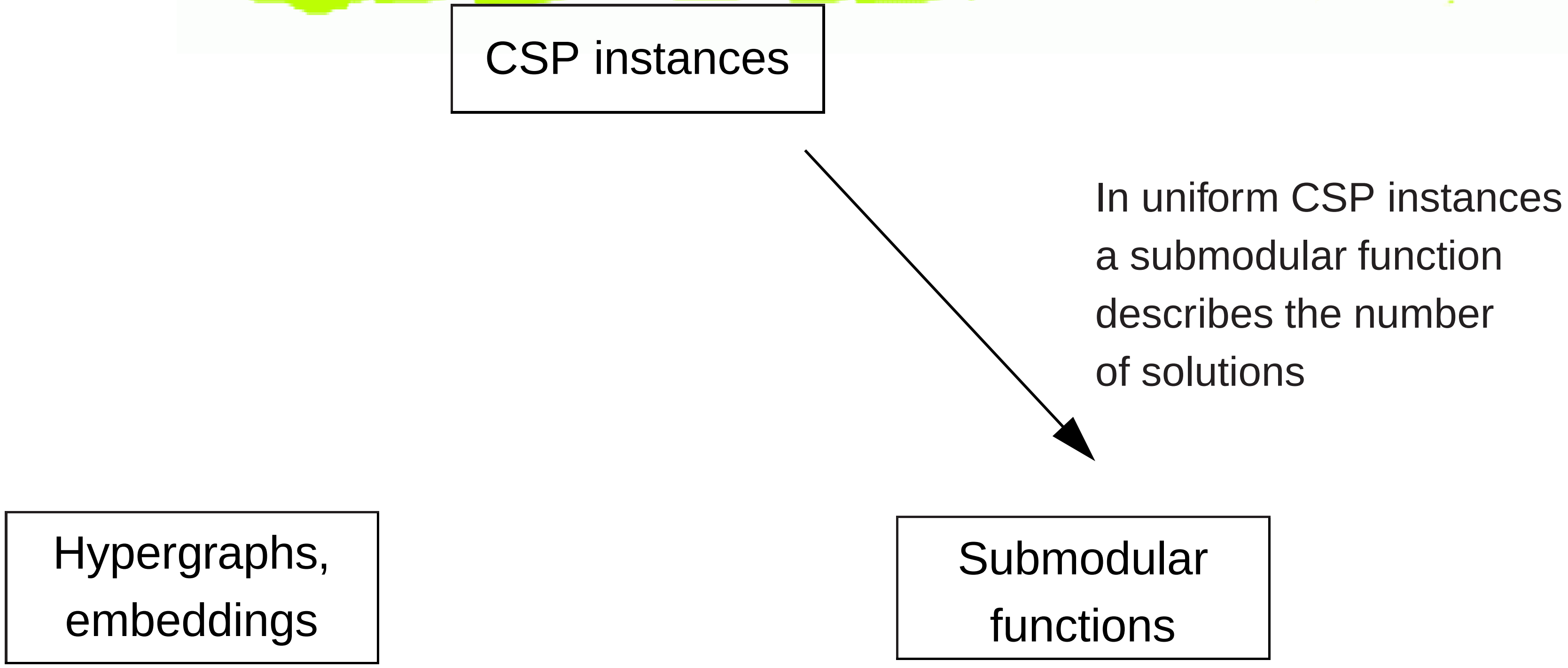
Three battlefields

CSP instances

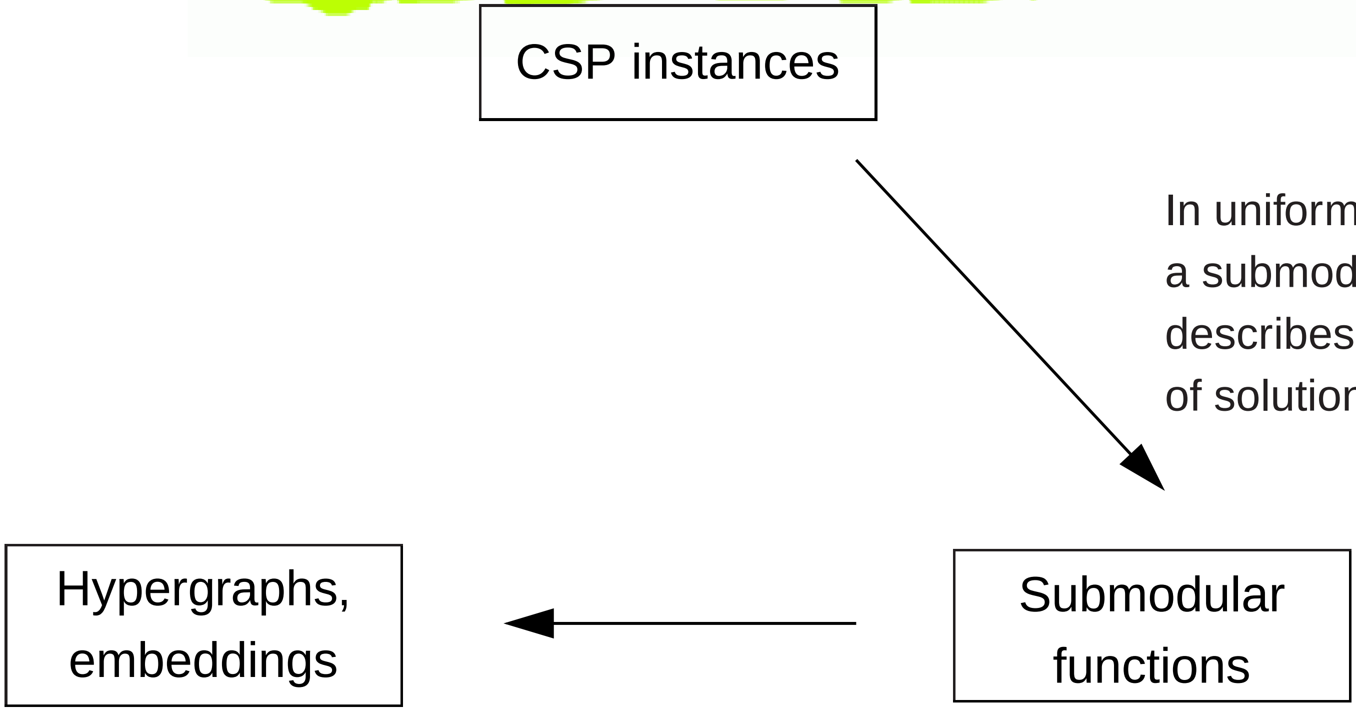
Hypergraphs, embeddings

Submodular functions

Three battlefields


CSP instances
In uniform CSP instances a submodular function describes the number of solutions
Hypergraphs, embeddings
Submodular functions

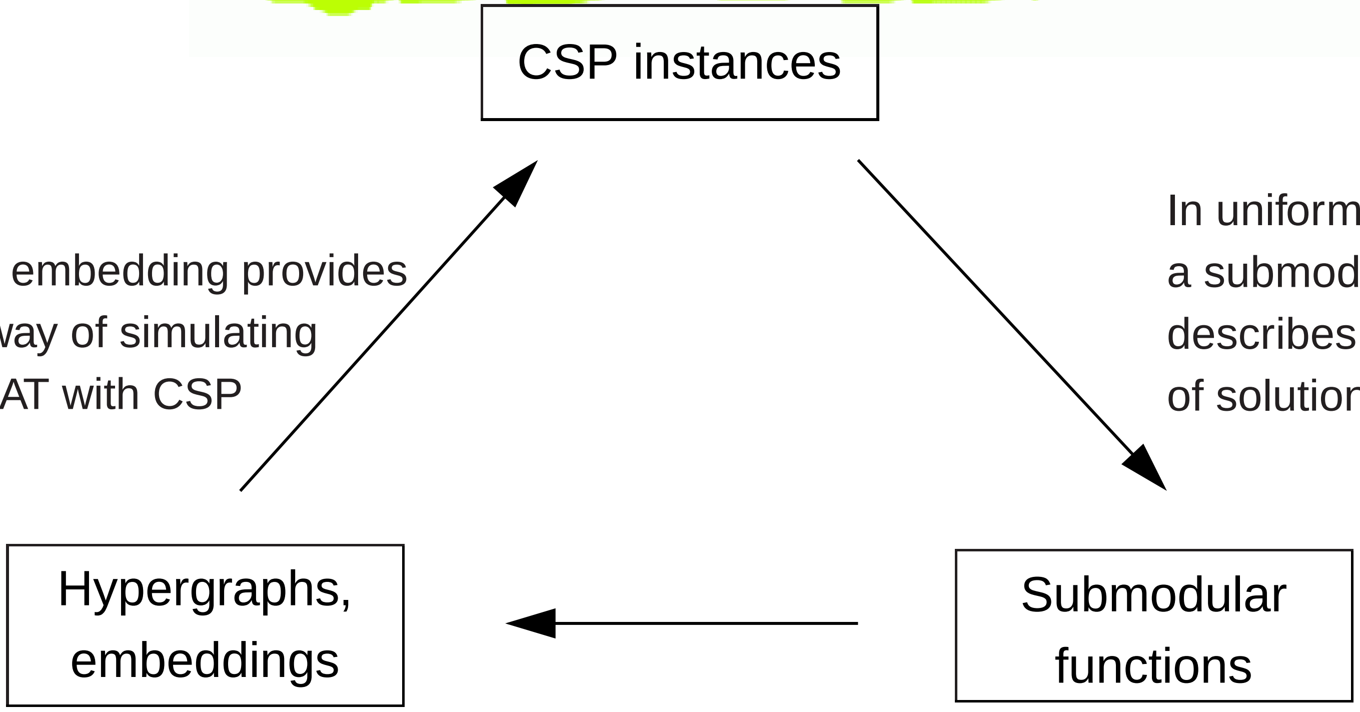
Three battlefields



In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Three battlefields

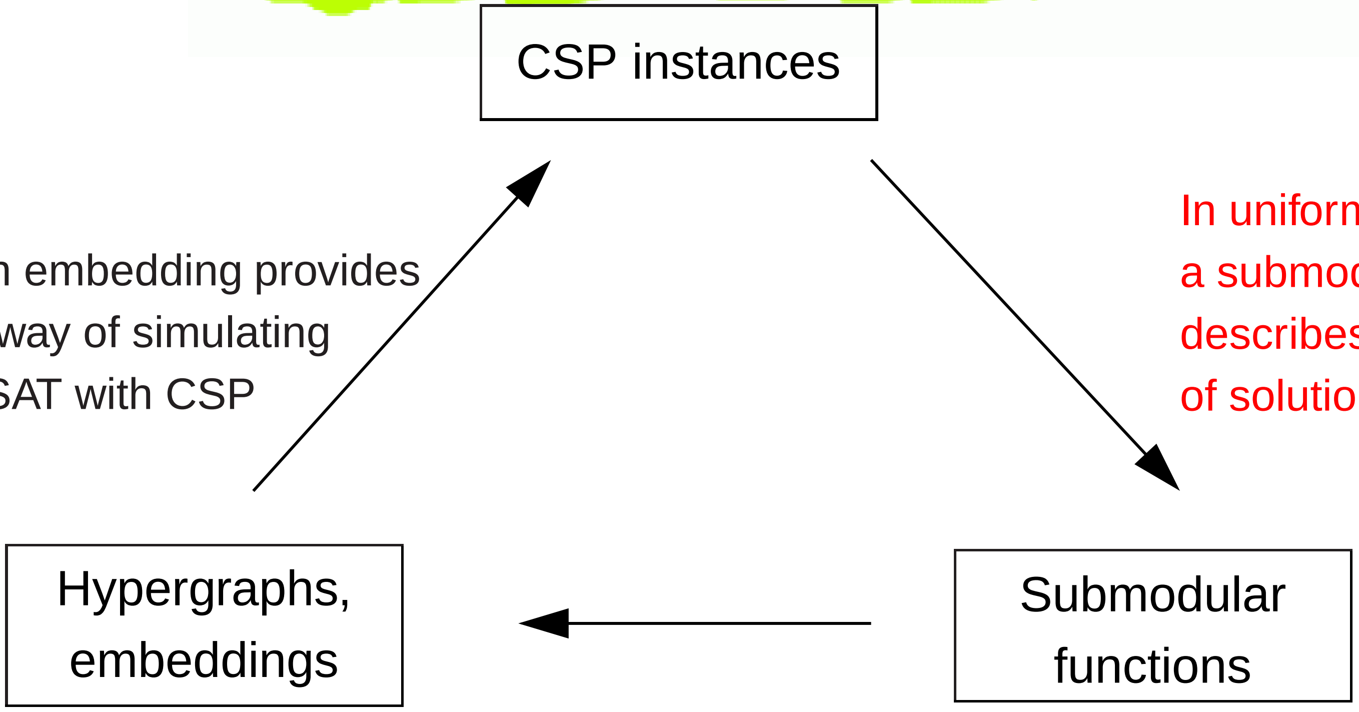


An embedding provides a way of simulating 3SAT with CSP

In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Three battlefields



An embedding provides a way of simulating 3SAT with CSP

In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Number of solutions

sol(B) : number of solutions of the instance projected to subset B of variables.

The **projection** of instance I to $B \subseteq V$ is an instance on B such that for every constraint c of I with scope S such that $S \cap B \neq \emptyset$, there is a constraint on $S \cap B$ that is satisfied if it can be extended to a satisfying tuple of c.

Example:
Projection to $\{v_1, v_2, v_3\}$

$$s = (v_2, v_3, v_4, v_5),\ R = \begin{matrix}(1,4,1,1)\\(2,3,2,4)\\(2,3,5,1)\\(5,5,1,1)\\(5,5,2,5)\end{matrix} \Rightarrow s = (v_2, v_3),\ R = \begin{matrix}(1,4)\\(2,3)\\(5,5)\end{matrix}$$

Number of solutions

sol(B) : number of solutions of the instance projected to subset B of variables.

The **projection** of instance I to $B \subseteq V$ is an instance on B such that for every constraint c of I with scope S such that $S \cap B \neq \emptyset$, there is a constraint on $S \cap B$ that is satisfied if it can be extended to a satisfying tuple of c.

Example:
Projection to $\{v_1, v_2, v_3\}$

$$s = (v_2, v_3, v_4, v_5),\ R = \begin{matrix}(1,4,1,1)\\(2,3,2,4)\\(2,3,5,1)\\(5,5,1,1)\\(5,5,2,5)\end{matrix} \Rightarrow \quad s = (v_2, v_3),\ R = \begin{matrix}(1,4)\\(2,3)\\(5,5)\end{matrix}$$

Fact: If we are given a tree decomposition of the hypergraph of instance I together with a list (having length $\leq C$) of all solutions of the **projection** to B for each bag B, then I can be solved in time polynomial in $\|I\|$ and C.

A crazy idea

Let I be a CSP instance with hypergraph H. Let $N := \|I\|$ and suppose that the submodular width of H is at most w.

Let $b(B) := \log_N \text{sol}(B)$, which is edge-dominated.

A crazy idea

Let I be a CSP instance with hypergraph H. Let $N := \|I\|$ and suppose that the submodular width of H is at most w.

Let $b(B) := \log_N \text{sol}(B)$, which is edge-dominated.

Crazy assumption: b is monotone and submodular.

Then by the definition of submodular width, there is a tree decomposition where $\text{sol}(B) \leq N^w$ for every bag
$\Rightarrow$ FPT algorithm!

A crazy idea

Let I be a CSP instance with hypergraph H. Let $N := \|I\|$ and suppose that the submodular width of H is at most w.

Let $b(B) := \log_N \text{sol}(B)$, which is edge-dominated.

Crazy assumption: b is monotone and submodular.

Then by the definition of submodular width, there is a tree decomposition where $\text{sol}(B) \leq N^w$ for every bag
$\Rightarrow$ FPT algorithm!

Problems:

- b is not necessarily monotone.
- b is not necessarily submodular.
- we don't even know the function b.

Small sets

Definition: Let X be M**-small** if $\mathrm{sol}(Y) \le M$ for every $Y \subseteq X$.

Fact: In time $f(H) \cdot (\|I\| \cdot M)^{O(1)}$, we can identify all M-small sets and compute $\mathrm{sol}(X)$ for every such set X.

We will care about the value of b only on the N^w-small sets, every other set will be “too large.”

By introducing further constraints, we can ensure that b is monotone on N^w-small sets: if an assignment $Y \subset X$ is not extendible to X, then we forbid it.

Now we know b and it is monotone on the sets we care about. But what about submodularity?

Uniformity

Definition: $\mathsf{max}(A|B)$ is the maximum number of extensions of a solution on B to a solution on A. Instance I is c**-uniform,** if $\mathsf{max}(A|B) \leq c \cdot \mathrm{sol}(A)/\mathrm{sol}(B)$ for every $B \subseteq A$.

Uniformity

Definition: $\max(A|B)$ is the maximum number of extensions of a solution on B to a solution on A. Instance I is c**-uniform,** if $\max(A|B) \le c \cdot \mathrm{sol}(A)/\mathrm{sol}(B)$ for every $B \subseteq A$.

Fact: If I is 1-uniform, then b is submodular.

$$\begin{aligned}
b(X) + b(Y) &= \log_N |\mathrm{sol}(X)| + \log_N \left(|\mathrm{sol}(X \cap Y)| \cdot \frac{|\mathrm{sol}(Y)|}{|\mathrm{sol}(X \cap Y)|} \right) \\
&= \log_N |\mathrm{sol}(X)| + \log_N \left(|\mathrm{sol}(X \cap Y)| \cdot \max(Y | X \cap Y) \right) \\
&\ge \log_N |\mathrm{sol}(X)| + \log_N \left(|\mathrm{sol}(X \cap Y)| \cdot \max(X \cup Y | X) \right) \\
&= \log_N \left(|\mathrm{sol}(X)| \cdot \frac{|\mathrm{sol}(X \cup Y)|}{|\mathrm{sol}(X)|} \right) + \log_N |\mathrm{sol}(X \cap Y)| \\
&= b(X \cap Y) + b(X \cup Y)
\end{aligned}$$

Uniformity

Definition: $\max(A|B)$ is the maximum number of extensions of a solution on B to a solution on A. Instance I is c**-uniform,** if $\max(A|B) \le c \cdot \text{sol}(A)/\text{sol}(B)$ for every $B \subseteq A$.

Fact: If I is 1-uniform, then b is submodular.

$$\begin{aligned}
b(X) + b(Y) &= \log_N |\text{sol}(X)| + \log_N \left(|\text{sol}(X \cap Y)| \cdot \frac{|\text{sol}(Y)|}{|\text{sol}(X \cap Y)|} \right) \\
&= \log_N |\text{sol}(X)| + \log_N \left(|\text{sol}(X \cap Y)| \cdot \max(Y | X \cap Y) \right) \\
&\ge \log_N |\text{sol}(X)| + \log_N \left(|\text{sol}(X \cap Y)| \cdot \max(X \cup Y | X) \right) \\
&= \log_N \left(|\text{sol}(X)| \cdot \frac{|\text{sol}(X \cup Y)|}{|\text{sol}(X)|} \right) + \log_N |\text{sol}(X \cap Y)| \\
&= b(X \cap Y) + b(X \cup Y)
\end{aligned}$$

If I is N^ϵ-uniform on N^w-small sets, then with some tweaking (adding low order terms) we can make b submodular.

Decomposition into uniform instances

Suppose that two N^w-small sets $B \subseteq A$ violate N^ϵ-uniformity: there are assignments on B having more than $N^\epsilon \cdot \mathrm{sol}(A)/\mathrm{sol}(B)$ extensions.

By adding a new constraint, we split the instance in two cases:

- in I_{small} every assignment on B has at most $\sqrt{N^\epsilon} \cdot \mathrm{sol}(A)/\mathrm{sol}(B)$ extensions,
- in I_{large} every assignment has more than that many extensions.

Repeat if necessary (idea from [Alon et al. 2006]).

We can show that the number of instances created by the procedure can be bounded by a function of the number of variables (independent of the size of the domain and the relations!).

Decomposition into uniform instances

Suppose that two N^w-small sets $B \subseteq A$ violate N^ϵ-uniformity: there are assignments on B having more than $N^\epsilon \cdot \text{sol}(A)/\text{sol}(B)$ extensions.

By adding a new constraint, we split the instance in two cases:

- in I_{small} every assignment on B has at most $\sqrt{N^\epsilon} \cdot \text{sol}(A)/\text{sol}(B)$ extensions,
- in I_{large} every assignment has more than that many extensions.

Repeat if necessary (idea from [Alon et al. 2006]).

The following measure decreases by at least $\epsilon/2$ in I_{small} and I_{large}:

$$W = \sum_{\substack{\emptyset \subseteq B \subseteq A \subseteq V \\ A,\, B \text{ are } N^w\text{-small}}} \log_N \max(A|B).$$

I_{small}: $\max(A|B)$ decreases by a factor of $\sqrt{N^\epsilon}$.
I_{large}: $\text{sol}(B) = \max(B|\emptyset)$ decreases by a factor of $\sqrt{N^\epsilon}$.

The algorithm

Algorithm for hypergraphs with submodular with at most w:

- Locate the N^w-small sets.
- Decompose the instance into a bounded number of N^ϵ-uniform instances $\Rightarrow b = \log_N \mathrm{sol}(B)$ is submodular (after some tweaking).
- For each new instance, try every tree decomposition — there has to be one where $b(B) \leq w$ and hence $\mathrm{sol}(B) \leq N^w$ for every bag b.
- Solve the new instance using this tree decomposition.

This completes the algorithmic part of the main result.

The algorithm

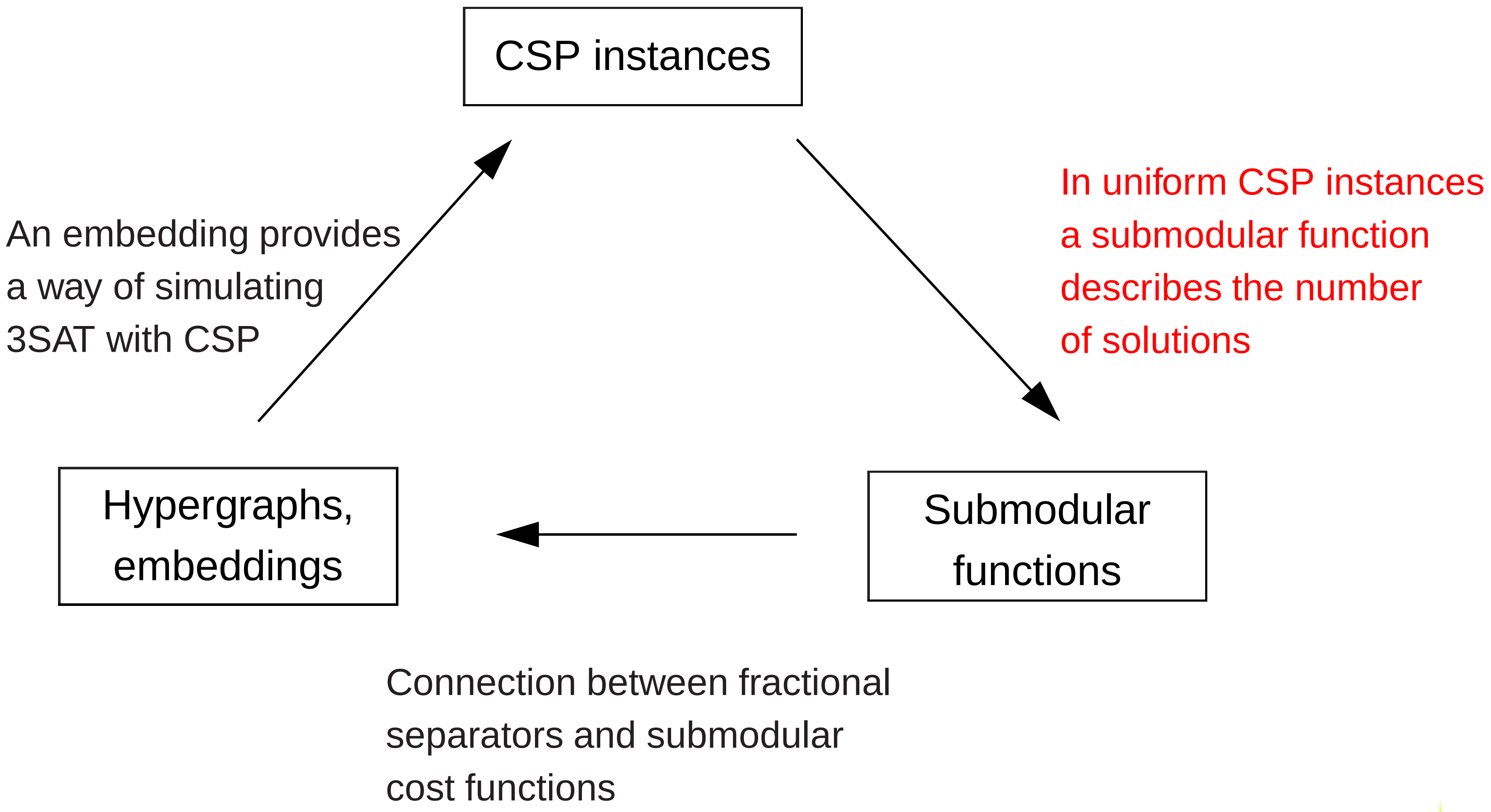
Algorithm for hypergraphs with submodular with at most w:

- Locate the N^w-small sets.
- Decompose the instance into a bounded number of N^ϵ-uniform instances $\Rightarrow b = \log_N \text{sol}(B)$ is submodular (after some tweaking).
- For each new instance, try every tree decomposition — there has to be one where $b(B) \leq w$ and hence $\text{sol}(B) \leq N^w$ for every bag b.
- Solve the new instance using this tree decomposition.

This completes the algorithmic part of the main result.

- **Idea #1:** The decomposition depends not only on the hypergraph of the instance, but on the actual constraint relations.
- **Idea #2:** We branch on adding further restrictions, and apply different tree decompositions to each resulting instance.

Three battlefields

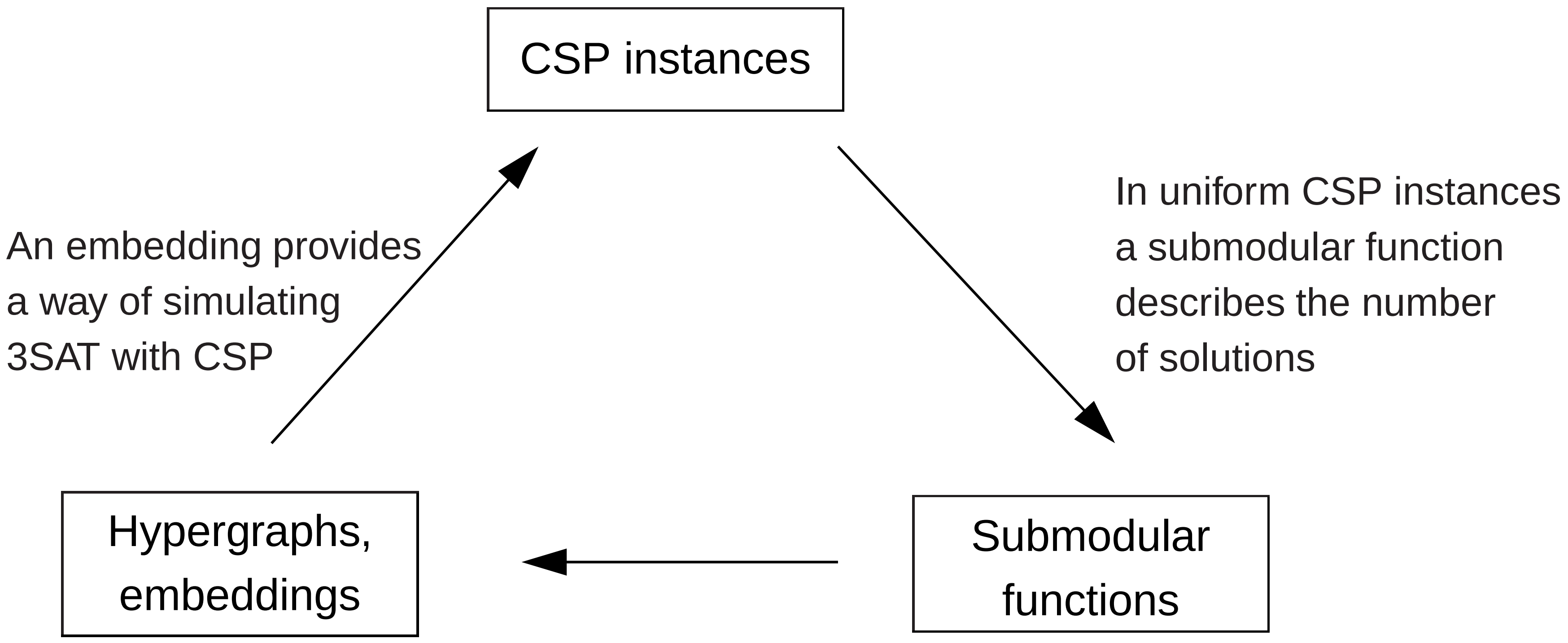


An embedding provides a way of simulating 3SAT with CSP

In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Three battlefields


CSP instances
Hypergraphs, embeddings
Submodular functions
An embedding provides a way of simulating 3SAT with CSP
In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Highly connected sets

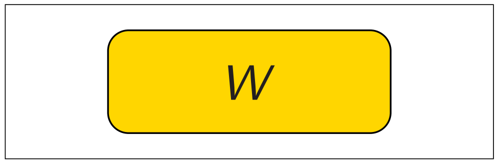
For the hardness part, we want to characterize submodular width analogously to other width measures: if submodular width is large, then there is a “large highly connected set” in the hypergraph.

Fact: If $\mathrm{tw}(G) = \Omega(k)$, then there is a set W of $O(k)$ vertices such that for every disjoint $A, B \subseteq W$, there is no (A, B)-separator of less than $\min\{|A|, |B|\}$ vertices.

Highly connected sets

Fact: If $\text{tw}(G) = \Omega(k)$, then there is a set W of $O(k)$ vertices such that for every disjoint $A, B \subseteq W$, there is no (A, B)-separator of less than $\min\{|A|, |B|\}$ vertices.

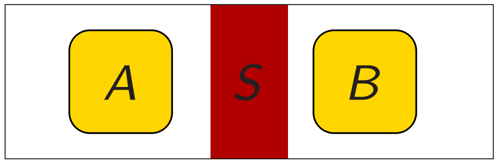
Separator-based approach:



Highly connected sets

Fact: If $\mathrm{tw}(G) = \Omega(k)$, then there is a set W of $O(k)$ vertices such that for every disjoint $A, B \subseteq W$, there is no (A, B)-separator of less than $\min\{|A|, |B|\}$ vertices.

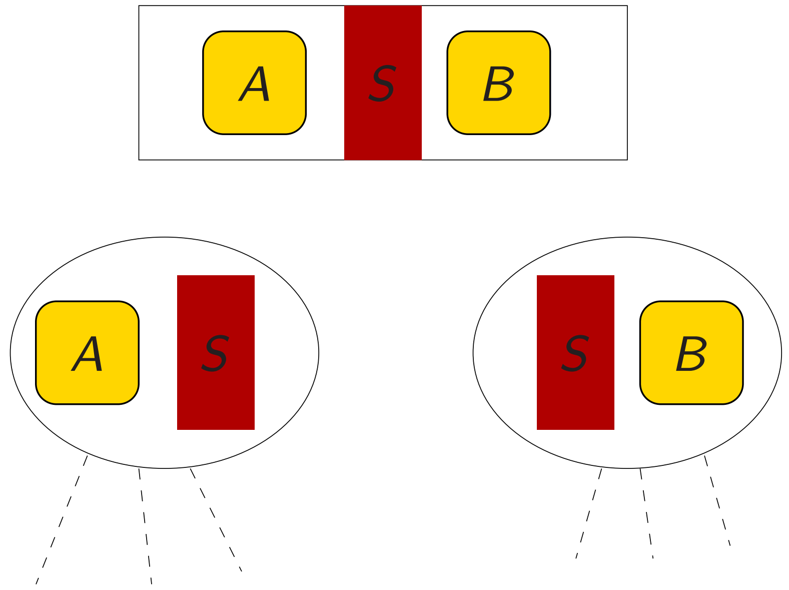
Separator-based approach:



Highly connected sets

Fact: If $\mathrm{tw}(G) = \Omega(k)$, then there is a set W of $O(k)$ vertices such that for every disjoint $A, B \subseteq W$, there is no (A, B)-separator of less than $\min\{|A|, |B|\}$ vertices.

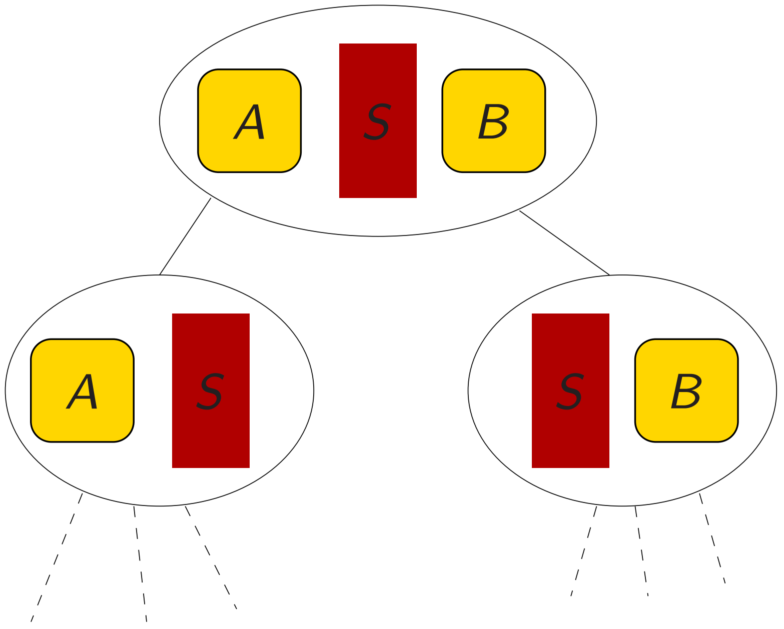
Separator-based approach:



Highly connected sets

Fact: If $\mathrm{tw}(G) = \Omega(k)$, then there is a set W of $O(k)$ vertices such that for every disjoint $A, B \subseteq W$, there is no (A, B)-separator of less than $\min\{|A|, |B|\}$ vertices.

Separator-based approach:



Highly connected set

Definition: A set W is b-connected, if for every disjoint $X, Y \subseteq W$, there is no (X, Y)-separator S with $b(S) < \min\{b(X), b(Y)\}$.

If b-width is at least w for some submodular function b, the separator-based approach of finding tree decompositions **almost** gives us (there is one major technical difficulty) a set b-connected set W with $b(W) = \Omega(w)$.

But we want a notion of highly connected set that is determined only by the hyper-graph H and is not related to any submodular function.

Highly connected set

Definition: A **fractional independent** set of H is an assignment $\mu : V(H) \to \{0,1\}$ such that $\mu(e) \leq 1$ for every hyperedge e (we define $\mu(X) = \sum_{v \in X} \mu(v)$).

Definition: A **fractional** (X, Y)**-separator** is an assignment $E(H) \to \{0,1\}$ such that every $X - Y$ path is covered by total weight at least 1.

Definition: Let $\lambda > 0$ be a constant (say, 0.01) and let μ be a fractional independent set. A set W is (μ, λ)**-connected** if for every disjoint $X, Y \subseteq W$, there is no fractional (X, Y)-separator of weight less than $\lambda \cdot \min\{\mu(X), \mu(Y)\}$.

We need to connect somehow the notions of "fractional (X, Y)-separator having small weight" and "(X, Y)-separator S with $b(S)$ small."

Result on separation

What does it mean that there is a fractional (X, Y)-separator of small weight?

Lemma: If there is a fractional (X, Y)-separator of weight w, then for every edge-dominated monotone submodular function b, there is an (X, Y)-separator S with $b(S) = O(w)$.

Result on separation

What does it mean that there is a fractional (X, Y)-separator of small weight?

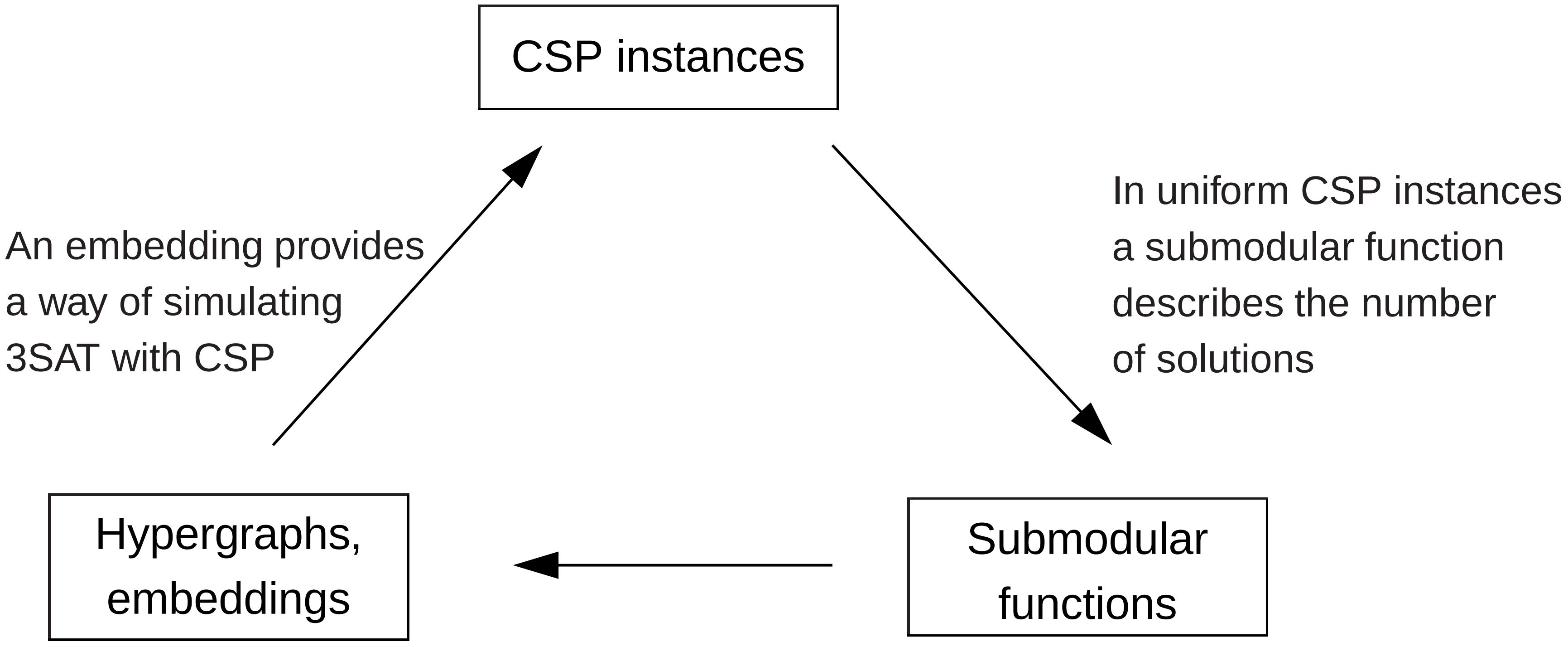
Lemma: If there is a fractional (X, Y)-separator of weight w, then for every edge-dominated monotone submodular function b, there is an (X, Y)-separator S with $b(S) = O(w)$.

Definition: (repeated) A set W is b-connected, if for every disjoint $X, Y \subseteq W$, there is no (X, Y)-separator S with $b(S) < \min\{b(X), b(Y)\}$.

If there is no (X, Y)-separator S with $b(S) < \min\{b(X), b(Y)\}$, then there is no fractional separator of weight $\lambda \cdot \min\{b(X), b(Y)\}$ for some $\lambda > 0$.

So we have obtained a set W that is “highly connected” in the sense that certain fractional separators do not exist, and this takes us into the domain of pure hypergraph properties, separators, flows, etc.

Three battlefields

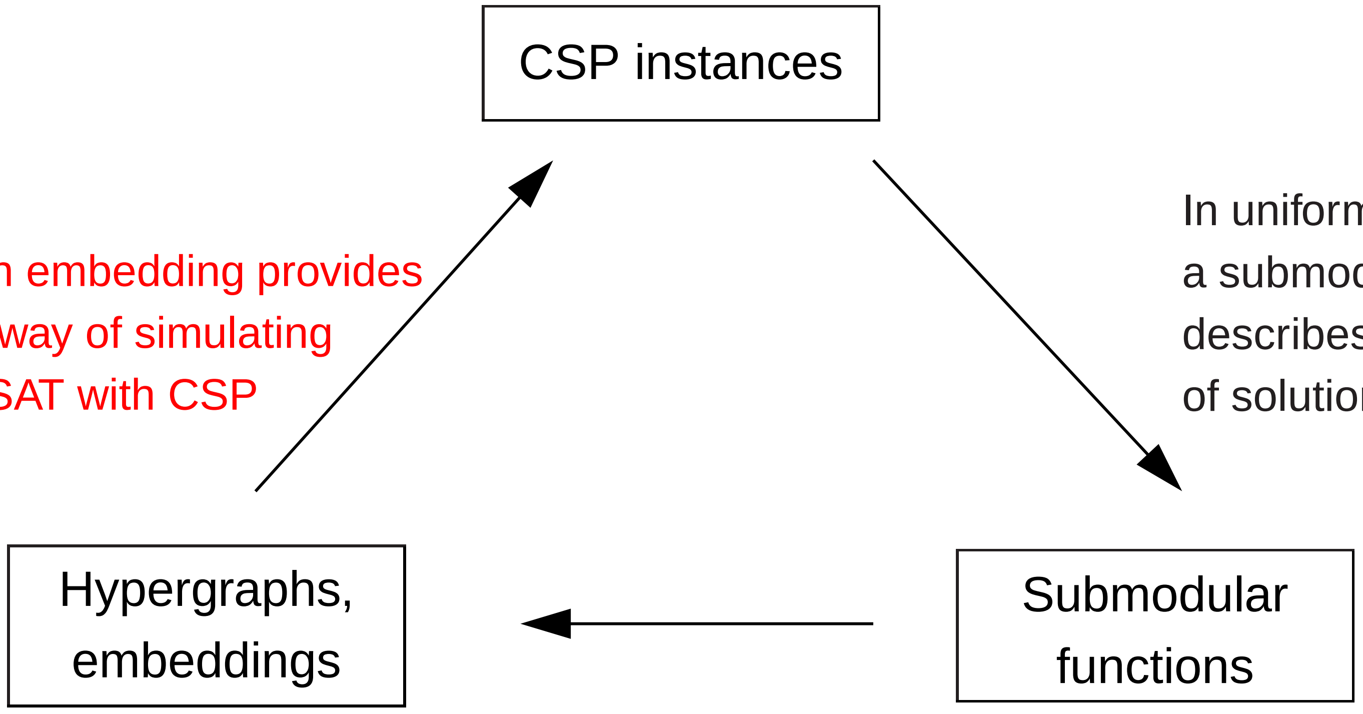


An embedding provides a way of simulating 3SAT with CSP

In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Three battlefields



An embedding provides a way of simulating 3SAT with CSP

In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Embeddings

Definition: A q**-embedding** of graph F in hypergraph H maps a subset of $V(H)$ to each vertex of H such that

- For every $v \in V(F)$, $\phi(v)$ is connected.
- If $u, v \in V(F)$ are adjacent in F, then $\phi(u)$ and $\phi(v)$ touch: there is a hyperedge intersecting both of them
- Every hyperedge e of H intersects the images of at most q vertices of F.

Lemma: [M. 2007] For **graphs** F and G, if $m = |E(F)|$ is sufficiently large and $k = \text{tw}(G)$, then there is a q-embedding of F in G for $q = O(m \log k/k)$.

Embeddings

Definition: A q-**embedding** of graph F in hypergraph H maps a subset of $V(H)$ to each vertex of H such that

- For every $v \in V(F)$, $\phi(v)$ is connected.
- If $u, v \in V(F)$ are adjacent in F, then $\phi(u)$ and $\phi(v)$ touch: there is a hyperedge intersecting both of them
- Every hyperedge e of H intersects the images of at most q vertices of F.

Lemma: [M. 2007] For **graphs** F and G, if $m = |E(F)|$ is sufficiently large and $k = \text{tw}(G)$, then there is a q-embedding of F in G for $q = O(m \log k / k)$.

We show:

Lemma: For a graph F and hypergraph H, if $m = |E(F)|$ is sufficiently large and H has submodular width w, then there is a q-embedding of F in H for $q = O(m/w^{\frac{1}{4}})$.

Proof: Flows, multicuts on hypergraphs, LP duality etc.

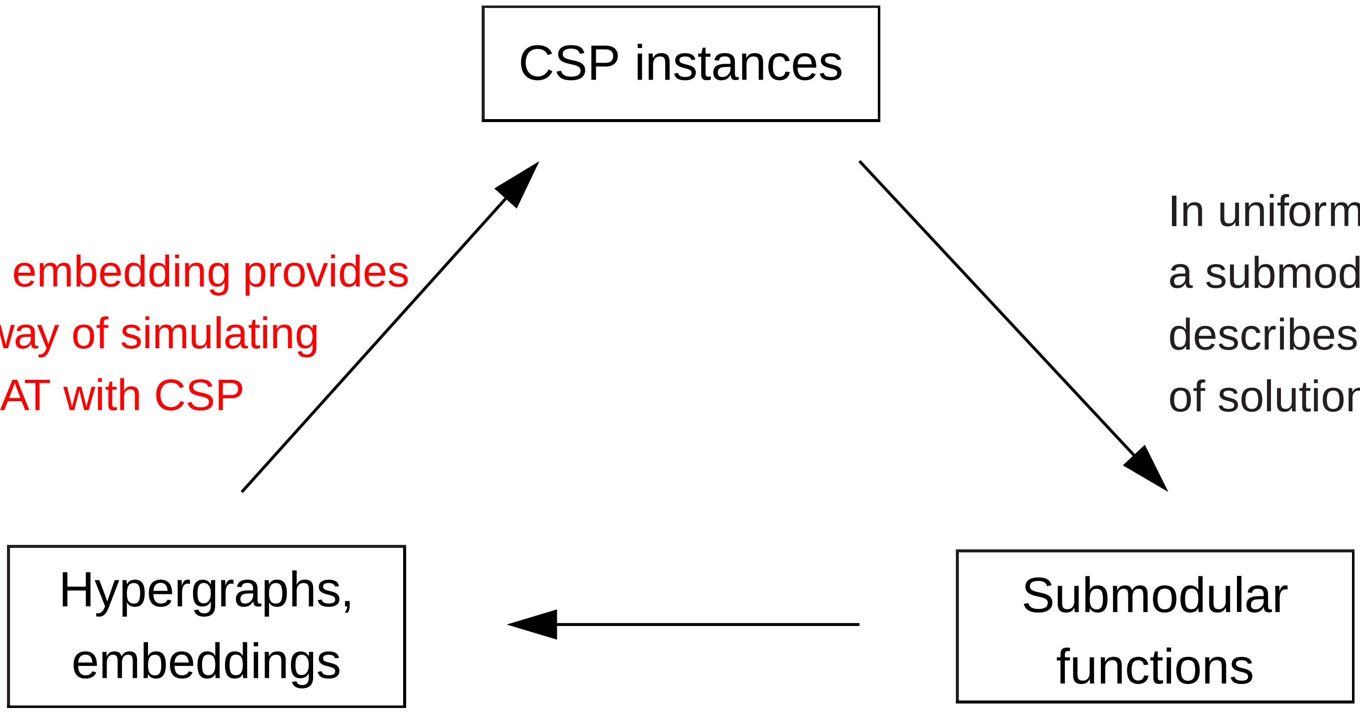
Hardness proof

Theorem: If $\mathcal{H}$ is a recursively enumerable class of hypergraphs with unbounded submodular width, then CSP($\mathcal{H}$) is not fixed-parameter tractable (assuming ETH).

Proof outline:

- Given a 3SAT instance with m clauses and n variables, we turn it into a CSP instance I_1 with $3m$ binary constraints, and domain size 3.
- We use the embedding result to find a q-embedding of the primal graph F of I_1 into some $H_k \in \mathcal{H}$ (chosen appropriately).
- We simulate I_1 by an instance I_2 whose primal graph is H_k: each edge of I_2 "sees" at most q variables of I_1, thus each constraint relation has size $\le 3^q$.
- Now the 3SAT problem can be solved by solving I_2. Calculation of the running time shows that that an FPT algorithm for CSP($\mathcal{H}$) would give a $2^{o(m)}$ algorithm for m-clause 3SAT, violating ETH.

Three battlefields



An embedding provides a way of simulating 3SAT with CSP

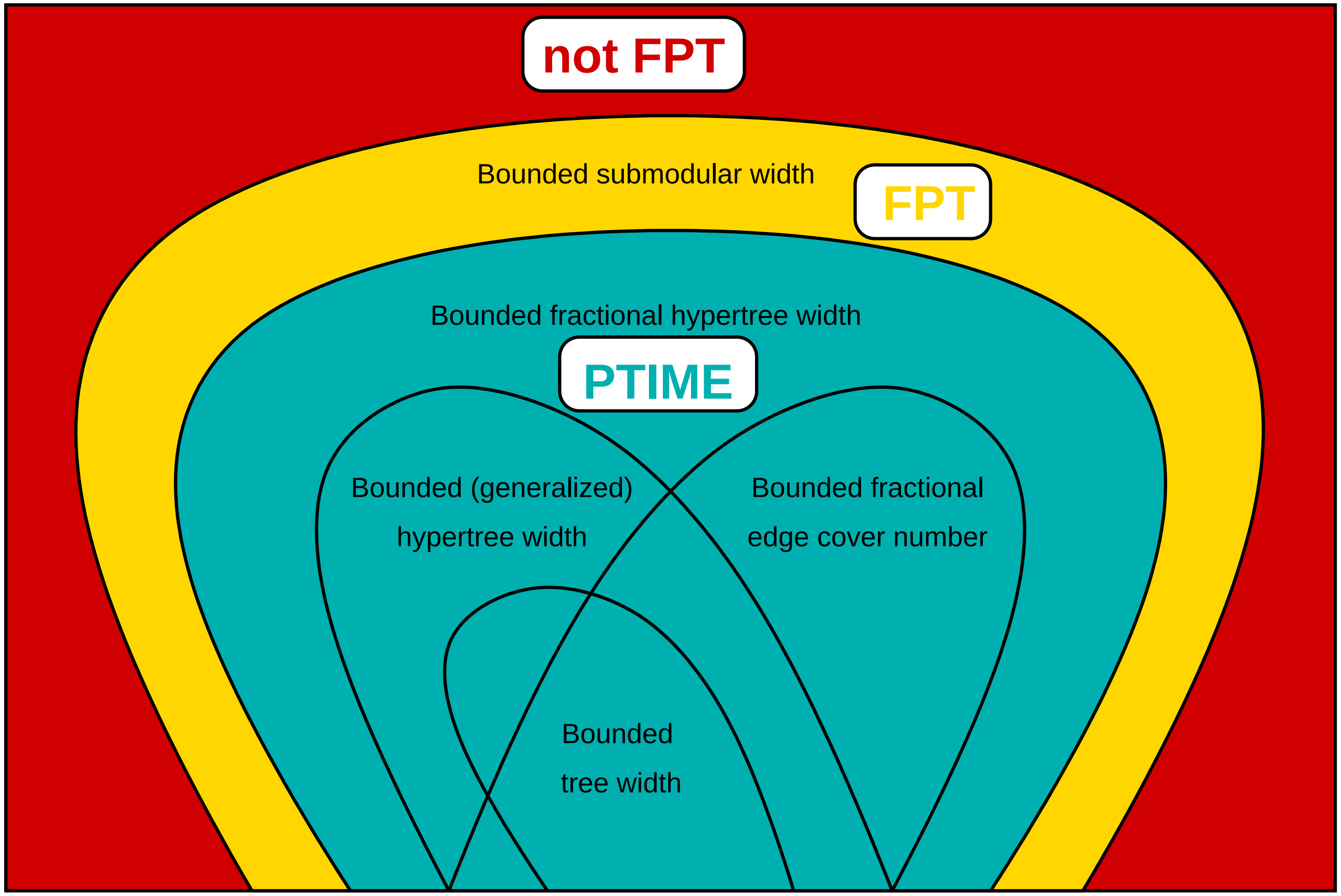
In uniform CSP instances a submodular function describes the number of solutions

Connection between fractional separators and submodular cost functions

Conclusions

- Characterization of CSP($\mathcal{H}$) with respect to fixed-parameter tractability.
- Main new definition: submodular width.
- Why fixed-parameter tractability?
- What happens in the “gray zone”?

Tractable classes


not FPT
Bounded submodular width
FPT
Bounded fractional hypertree width
PTIME
Bounded (generalized) hypertree width
Bounded fractional edge cover number
Bounded tree width