Constraint Solving via Fractional Edge Covers

Martin Grohe and Dániel Marx

Humboldt-Universität zu Berlin
Institut für Informatik

Symposium on Discrete Algorithms (SODA) 2006
January 22, 2006
Many natural problems can be expressed as a **Constraint Satisfaction Problem**, where a conjunction of clauses has to be satisfied.

\[ I = C_1(x_1, x_2, x_3) \land C_2(x_2, x_4) \land C_3(x_1, x_3, x_4) \]

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.
Constraint Satisfaction Problems (CSP)

Many natural problems can be expressed as a Constraint Satisfaction Problem, where a conjunction of clauses has to be satisfied.

\[ I = C_1(x_1, x_2, x_3) \land C_2(x_2, x_4) \land C_3(x_1, x_3, x_4) \]

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.

**Example:** 3-COLORING is a CSP problem.
**Variables:** vertices, **Domain:** \{1, 2, 3\}, **Constraints:** one for each edge.
Tractable structures

Structural properties that can make a CSP instance tractable:

- tree width
- hypertree width [Gottlob et al. '99]
- fractional edge cover number
- fractional hypertree width
Representation issues

How are the constraints represented in the input?

- full truth table
- listing the satisfying tuples
- formula/circuit
- oracle
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 & AI.
Constraints are known databases, “satisfying” means “appears in the database.”
**Tree width**

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

**Tree decomposition:** Bags of 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.

**Tree width:** width of the best decomposition.

**Fact:** Tree width = 1 \( \iff \) graph is a forest.
**Bounded tree width graphs**

Many problems are polynomial-time solvable for bounded tree width graphs:

- **Vertex Coloring**
- **Edge Coloring**
- **Hamiltonian Cycle**
- **Maximum Clique**
- **Vertex Disjoint Paths**
Bounded tree width graphs

Many problems are polynomial-time solvable for bounded tree width graphs:

- **Vertex Coloring**
- **Edge Coloring**
- **Hamiltonian Cycle**
- **Maximum Clique**
- **Vertex Disjoint Paths**

Usually, if a problem can be solved on trees by bottom-up dynamic programming, then the same approach works for bounded tree width graphs.
**CSP and tree width**

**Primal (Gaifman) graph:** vertices are the variables, and two vertices are connected if they appear in a common constraint.

**Fact:** For every $w$, there is a polynomial-time algorithm solving CSP instances where the primal graph have tree width at most $w$. 
**CSP and tree width**

**Primal (Gaifman) graph:** vertices are the variables, and two vertices are connected if they appear in a common constraint.

**Fact:** For every $w$, there is a polynomial-time algorithm solving CSP instances where the primal graph have tree width at most $w$.

This result is best possible.

CSP($\mathcal{G}$): the problem restricted to instances where the primal graph is in $\mathcal{G}$.

**Theorem:** [Grohe '03]

CSP($\mathcal{G}$) is polynomial-time solvable $\iff$ $\mathcal{G}$ has bounded tree width (assuming $\text{FPT} \neq \text{W}[1]$).
CSP and hypergraphs

Hypergraph: edges are arbitrary subsets of vertices.

Hypergraph of a CSP instance: vertices are the variables, each constraint is an edge.
Hypergraph: edges are arbitrary subsets of vertices.

Hypergraph of a CSP instance: vertices are the variables, each constraint is an edge.

Considering the hypergraph instead of the primal graph makes the complexity analysis more precise.

\[ I_1 = C(x_1, x_2, \ldots, x_n) \text{ vs.} \]
\[ I_2 = C(x_1, x_2) \land C(x_1, x_3) \land \cdots \land C(x_{n-1}, x_n) \]

\( I_1, I_2 \) have the same primal graph \( K_n \), but \( I_1 \) is always easy, \( I_2 \) can be hard.
Hypergraph: edges are arbitrary subsets of vertices.

Hypergraph of a CSP instance: vertices are the variables, each constraint is an edge.

Considering the hypergraph instead of the primal graph makes the complexity analysis more precise.

\[ I_1 = C(x_1, x_2, \ldots, x_n) \text{ vs. } I_2 = C(x_1, x_2) \land C(x_1, x_3) \land \cdots \land C(x_{n-1}, x_n) \]

\( I_1, I_2 \) have the same primal graph \( K_n \), but \( I_1 \) is always easy, \( I_2 \) can be hard.

**Observation:** If there is a constraint that covers every variable, then we have to test at most \( \| I \| \) possible assignments.

**Observation:** If the variables can be covered by \( k \) constraints, then we have to test at most \( \| I \|^k \) possible assignments.
In a **hypertree decomposition** [Gottlob et al. ’99] of width $w$, bags of vertices are arranged in a tree structure such that

1. If $u$ and $v$ are connected by an edge, then there is a bag containing both of them.
2. For every vertex $v$, the bags containing $v$ form a connected subtree.
3. For each bag, there are $w$ edges (called the **guards**) that cover the bag.

**Hypertree width**: width of the best decomposition.
In a **hypertree decomposition** [Gottlob et al. '99] of width $w$, bags of vertices are arranged in a tree structure such that

1. If $u$ and $v$ are connected by an edge, then there is a bag containing both of them.
2. For every vertex $v$, the bags containing $v$ form a connected subtree.
3. For each bag, there are $w$ edges (called the **guards**) that cover the bag.

**Hypertree width:** width of the best decomposition.

**Footnote:** This is actually called generalized hypertree width for historical reasons.
**Hypertree width**

**Theorem:** [Gottlob et al. ’99] For every $w$, there is a polynomial-time algorithm for solving CSP on instances with hypergraphs having hypertree width at most $w$.

**Algorithm:** Bottom up dynamic programming. There are at most $||I||^w$ possible satisfying assignments for each bag.
**Hypertree width**

**Theorem:** [Gottlob et al. ’99] For every $w$, there is a polynomial-time algorithm for solving CSP on instances with hypergraphs having hypertree width at most $w$.

**Algorithm:** Bottom up dynamic programming. There are at most $||I||^w$ possible satisfying assignments for each bag.

**Generalization:** Is there some more general property that makes the number of satisfying assignments of a bag polynomial?
An edge cover of a hypergraph is a subset of the edges such that every vertex is covered by at least one edge.

\( \varrho(H) \): size of the smallest edge cover.

A fractional edge cover is a weight assignment to the edges such that every vertex is covered by total weight at least 1.

\( \varrho^*(H) \): smallest total weight of a fractional edge cover.
An edge cover of a hypergraph is a subset of the edges such that every vertex is covered by at least one edge.

$\varrho(H)$: size of the smallest edge cover.

A fractional edge cover is a weight assignment to the edges such that every vertex is covered by total weight at least 1.

$\varrho^*(H)$: smallest total weight of a fractional edge cover.

$\varrho(H) = 2$
(Fractional) edge covering

An edge cover of a hypergraph is a subset of the edges such that every vertex is covered by at least one edge.
\( g(H) \): size of the smallest edge cover.

A fractional edge cover is a weight assignment to the edges such that every vertex is covered by total weight at least 1.
\( g^*(H) \): smallest total weight of a fractional edge cover.

\[
\begin{align*}
g(H) &= 2 \\
g^*(H) &= 1.5
\end{align*}
\]
**Fractional edge covering**

**Lemma:** (trivial) If the hypergraph of the instance has edge covering number $w$, then there are at most $\|I\|^w$ satisfying assignments.

**Lemma:** If the hypergraph of the instance has fractional edge covering number $w$, then there are at most $\|I\|^w$ satisfying assignments.

This can be shown using the following combinatorial lemma:

**Shearer’s Lemma:** Let $H = (V, E)$ be a hypergraph, and let $A_1, A_2, \ldots, A_p$ be (not necessarily distinct) subsets of $V$ such that each $v \in V$ is contained in at least $q$ of the $A_i$’s. Denote by $E_i$ the edge set of the hypergraph projected to $A_i$. Then

$$|E| \leq \prod_{i=1}^{p} |E_i|^{1/q}.$$
Lemma: (trivial) If the hypergraph of the instance has edge covering number $w$, then there are at most $\|I\|^w$ satisfying assignments.

Lemma: If the hypergraph of the instance has fractional edge covering number $w$, then there are at most $\|I\|^w$ satisfying assignments, and they can be enumerated in $\|I\|^{O(w)}$ time.

This can be shown using the following combinatorial lemma:

Shearer’s Lemma: Let $H = (V, E)$ be a hypergraph, and let $A_1, A_2, \ldots, A_p$ be (not necessarily distinct) subsets of $V$ such that each $v \in V$ is contained in at least $q$ of the $A_i$’s. Denote by $E_i$ the edge set of the hypergraph projected to $A_i$. Then

$$|E| \leq \prod_{i=1}^{p} |E_i|^{1/q}.$$
**Fractional hypertree width**

In a fractional hypertree decomposition of width $w$, bags of vertices are arranged in a tree structure such that

1. If $u$ and $v$ are connected by an edge, then there is a bag containing both of them.
2. For every vertex $v$, the bags containing $v$ form a connected subtree.
3. A fractional edge cover of weight $w$ is given for each bag.

**Fractional hypertree width**: width of the best decomposition.
Fractional hypertree width

In a fractional hypertree decomposition of width $w$, bags of vertices are arranged in a tree structure such that

1. If $u$ and $v$ are connected by an edge, then there is a bag containing both of them.
2. For every vertex $v$, the bags containing $v$ form a connected subtree.
3. A fractional edge cover of weight $w$ is given for each bag.

**Fractional hypertree width:** width of the best decomposition.

**Theorem:** For every $w$, there is a polynomial-time algorithm for solving CSP if a fractional hypertree decomposition of width at most $w$ is given in the input.

Currently we do not know if deciding fractional hypertree width $\leq w$ is possible in polynomial time for every fixed value of $w$. 
Law enforcement on graphs

Robber and Cops Game: $k$ cops try to capture a robber in the graph.

- In each step, the cops can move from vertex to vertex arbitrarily with helicopters.
- The robber moves infinitely fast, and sees where the cops will land.
- The robber cannot go through the vertices blocked by the cops.
Robber and Cops Game: $k$ cops try to capture a robber in the graph.

- In each step, the cops can move from vertex to vertex arbitrarily with helicopters.
- The robber moves infinitely fast, and sees where the cops will land.
- The robber cannot go through the vertices blocked by the cops.

**Theorem:** [Seymour and Thomas '93]

$k$ cops can win the game $\iff$ the tree width of the graph is at most $k - 1$.

The winner of the game can be determined in $n^{O(k)}$ time $\Rightarrow$ tree width $\leq k$ can be checked in polynomial time for fixed $k$. 

"Law enforcement on graphs"
Robber and Marshals:
Played on a hypergraph, a marshal can occupy an edge blocking all the vertices of the edge at the same time.

**Theorem:** [Adler et al. ’05] $k$ marshals can win the game if hypertree width is $\leq k$, and they cannot win the game if hypertree width is $\geq 3k + 1$.

$\Rightarrow n^{O(k)}$ algorithm for approximating the hypertree width.
Robber and Marshals:
Played on a hypergraph, a marshal can occupy an edge blocking all the vertices of the edge at the same time.

Theorem: [Adler et al. ’05] \( k \) marshals can win the game if hypertree width is \( \leq k \), and they cannot win the game if hypertree width is \( \geq 3k + 1 \).

\[ \Rightarrow n^{O(k)} \] algorithm for approximating the hypertree width.

Robber and Army:
A general has \( k \) battalions. A battalion can be divided arbitrarily, each part can be assigned to an edge. A vertex is blocked if it is covered by one full battalion.

Theorem: \( k \) battalions can win the game if fractional hypertree width is \( \leq k \), and they cannot win the game if fractional hypertree width is \( \geq 3k + 2 \).

We don’t know how to turn this result into an algorithm (there are too many army positions).
Conclusions

- CSP where constraints are represented as lists of satisfying tuples.
- Bounded tree width and bounded hypertree width make the problem polynomial-time solvable.
- New: Bounded fractional edge cover number.
- New: Rational hypertree width.
- Open: Finding fractional hypertree decompositions.
- Robber and Army Game: equivalent to fractional hypertree width (up to a constant factor).
- Are there other classes of hypergraphs where CSP is easy? Can we prove that bounded fractional hypertree width is best possible?