

What is a tractable class?

Hubie Chen

Dagstuhl - October 2009

Left-hand Side Constraint Satisfaction



- ▶ This talk: we formalize constraint satisfaction as the problem of deciding, given a pair of rel. structures (\mathbf{A}, \mathbf{B}) , if there exists a hom. $\mathbf{A} \rightarrow \mathbf{B}$
- ▶ Will discuss *left-hand side* restrictions.
For a set of structures \mathcal{A} , define $\text{CSP}(\mathcal{A}, -)$ to be the CSP restricted to instances (\mathbf{A}, \mathbf{B}) where $\mathbf{A} \in \mathcal{A}$



- ▶ Thm (Freuder '90): Suppose \mathcal{A} has bounded treewidth. Then, there exists a polytime alg. that, given an instance of $\text{CSP}(\mathcal{A}, -)$, correctly decides it (yes/no).
- ▶ Can be shown by computing a tree decomp., and then using dynamic programming
- ▶ This can be shown by k -consistency alg – alg that outputs "unsat" or "?"
On such a class $\text{CSP}(\mathcal{A}, -)$, have:
 k -consistency outputs "unsat" \Leftrightarrow instance unsat



- ▶ This result has been extended; extension can be described using following notion...
- ▶ Every relational structure \mathbf{A} has associated to it a structure $\text{core}(\mathbf{A})$
- ▶ One definition: $\text{core}(\mathbf{A})$ is a smallest induced substructure that is homomorphically equivalent to \mathbf{A}
- ▶ Key fact: $\text{tw}(\text{core}(\mathbf{A})) \leq \text{tw}(\mathbf{A})$, and there can be arbitrary difference

Getting to the core

- ▶ Thm (Dalmou, Kolaitis, Vardi '02 (DKV)): Suppose $\text{core}(\mathcal{A})$ has bounded treewidth. Then, there exists a polytime alg. that, given an instance of $\text{CSP}(\mathcal{A}, -)$, correctly decides it (yes/no).
- ▶ Once again! This can be shown by k -consistency alg – alg that outputs "unsat" or "?"
On such a class $\text{CSP}(\mathcal{A}, -)$, have:
 k -consistency outputs "unsat" \Leftrightarrow instance unsat
- ▶ Not known if it is possible to efficiently compute/approximate tree decomp. of $\text{core}(\mathbf{A})$ here...
- ▶ Tractable class defined using tree decompositions, but no tree decompositions ever computed!

On the membership problem

- ▶ As discussed and studied explicitly by DKV, there is a difference between these two tractability results!
- ▶ The complexity of deciding membership in the corresponding classes differs
- ▶ For each $k \geq 1$, there exists an alg that decides if given \mathbf{A} has $\text{tw}(\mathbf{A}) \leq k$
- ▶ Thm (DKV): For each $k \geq 2$, it is NP-complete to decide if given \mathbf{A} has $\text{tw}(\text{core}(\mathbf{A})) \leq k$

Tractability

- ▶ Gottlob, Leone, Scarcello (Artificial Intelligence '00):
For each CSP-decomposition method D , the class $C(D, k)$ is a tractable class of CSPs because the following important tasks are tractable:
 1. Checking membership of a CSP C in $C(D, k)$, and computing a corresponding CSP decomposition for C .
 2. Solving the CSP C . In turn, this task usually consists of the following two subtasks . . .



Strong tractability

- ▶ Let $f : \Sigma^* \rightarrow \{\text{yes}, \text{no}\}$ be a decision problem.
- ▶ Def: A set of instances $S \subseteq \Sigma^*$ is *strongly tractable* (with respect to f) if:
 1. there exists a polytime alg that decides, given $x \in \Sigma^*$, whether or not $x \in S$
(membership problem tractable)
 2. there exists a polytime alg that, given $x \in S$, computes the value $f(x)$
- ▶ Remarks:
 - ▶ Example. For any k , when $\mathcal{A} = \{\mathbf{A} \mid \text{tw}(\mathbf{A}) \leq k\}$, have that $\text{CSP}(\mathcal{A}, -)$ is a strongly tractable class
 - ▶ Definition readily extendible to function problems, etc.
 - ▶ Why *strong*? Strongest version of tractability discussed in this talk
 - ▶ But also, hard (for me?) to think of stronger tractability notion

Strong tractability: usable!



- ▶ Strong tractability results are usable!
- ▶ In the wild (in the real world), we are handed a CSP instance
 - ▶ would like to exploit tractable structure if it is there, but otherwise act gracefully
- ▶ With a strongly tractable class S :
Given an arbitrary instance x , check if $x \in S$.
 - ▶ if so, solve x using polytime alg
 - ▶ if not, solve x using other methods / exhaustive method

Promise tractability

- ▶ Let $f : \Sigma^* \rightarrow \{\text{yes, no}\}$ be a decision problem.
- ▶ Def: A set of instances $S \subseteq \Sigma^*$ is *promise tractable* (with respect to f) if:
 - ▶ there exists a polytime alg that, given $x \in S$, computes the value $f(x)$
- ▶ Example:
 - ▶ (DKV) For any k , when $\text{tw}(\text{core}(\mathcal{A})) \leq k$ have that $\text{CSP}(\mathcal{A}, -)$ is a promise tractable class

Bounded arity: closing the case

- ▶ Let \mathcal{A} be a set of relational structures.
- ▶ Thm (DKV): If $\text{core}(\mathcal{A})$ has bounded treewidth, then $\text{CSP}(\mathcal{A}, -)$ is promise tractable
- ▶ If we assume that \mathcal{A} has bounded arity, then...
- ▶ Thm (Grohe '07): If $\text{CSP}(\mathcal{A}, -)$ is promise tractable, then $\text{core}(\mathcal{A})$ has bounded treewidth!
(unless $W[1]$ contained in non-uniform *FPT*)

What about unbounded arity?

- ▶ Gottlob, Leone & Scarcello ('02) introduced complexity measures on hypergraphs:
hypertree width (htw), coverwidth (cw)
- ▶ $\text{cw}(\mathbf{A}) \leq \text{htw}(\mathbf{A}) \leq \text{tw}(\mathbf{A})$
- ▶ So, bounded tw \Rightarrow bounded htw \Rightarrow bounded cw
- ▶ Thm (GLS): For each $k \geq 1$, for the class $\mathcal{A}_k = \{\mathbf{A} \mid \text{htw}(\mathcal{A}) \leq k\}$, $\text{CSP}(\mathcal{A}_k, -)$ is strongly tractable.
Indeed, shown that a hypertree decomposition can be computed on $\mathbf{A} \in \mathcal{A}$
- ▶ Original line of work left open the tractability of bounded coverwidth

Taming coverwidth

- ▶ (Chen & Dalmau '05): introduced consistency algorithm for coverwidth, building on ideas in (DKV)
- ▶ Thm: If $\text{cw}(\text{core}(\mathcal{A}))$ bounded, then $\text{CSP}(\mathcal{A}, -)$ is promise tractable



- ▶ Again: tractable class defined by tree decompositions, but algorithm never computes tree decompositions
- ▶ Any way to eliminate the need for a promise?

Depromisfying the algorithm

- ▶ Algorithm which decides bounded $\text{cw}(\mathcal{A})$ instances correctly, but otherwise acts gracefully
- ▶ Self-reducibility style algorithm which makes repeated calls to our consistency algorithm
- ▶ Algorithm (Chen & Dalmau '05):
 1. Run consistency alg; if alg says "unsat", output "no"
 2. Pick any variable v ; try to find a value b such that setting $v = b$ allows consistency to be established
If such a value found, fix $v = b$, and repeat.
If no such value found, output "decline"
 3. If we have set all variables, then we have a satisfying assignment!
Output "yes". (Can also output sat. assignment.)
- ▶ Observe:
 - ▶ If instance x has bounded cw , algorithm correctly answers "yes" or "no".
 - ▶ If algorithm answers "yes" or "no", it does so correctly.

No-promise tractability

- ▶ Let $f : \Sigma^* \rightarrow \{\text{yes}, \text{no}\}$ be a decision problem.
- ▶ Def: A set of instances $S \subseteq \Sigma^*$ is *no-promise tractable* (with respect to f) if there exists a polytime alg M that, given an instance $x \in \Sigma^*$, outputs a value in $\{\text{yes}, \text{no}, \text{decline}\}$, where:
 1. if $M(x) \in \{\text{yes}, \text{no}\}$, then $M(x) = f(x)$
 2. if $x \in S$, then $M(x) \in \{\text{yes}, \text{no}\}$
- ▶ Via algorithm on previous slide, get: when $\text{cw}(\mathcal{A})$ bounded, $\text{CSP}(\mathcal{A}, -)$ is no-promise tractable
- ▶ Thm (Adler, Gottlob, Grohe '07): $\text{cw}(\mathbf{A}) \leq 3 * \text{htw}(\mathbf{A}) + 1$
Also implies no-promise tractability of coverwidth!

Summary

strongly tractable \Rightarrow no-promise tractable \Rightarrow promise tractable

bd $\text{tw}(\mathcal{A})$

bd $\text{tw}(\text{core}(\mathcal{A}))$

bd $\text{cw}(\mathcal{A})$

bd $\text{cw}(\text{core}(\mathcal{A}))$

- ▶ Can we move some of these results over “to the left”?
- ▶ Thm (Gottlob, Miklos, Schwentick '07): For bounded $\text{cw}(\mathcal{A})$, no!
For each $k \geq 3$, deciding if a given struct \mathbf{A} has $\text{cw}(\mathbf{A}) \leq k$ is NP-hard.

Which is better?

- ▶ Which is better: strong tractability or no-promise tractability?
- ▶ Feeling (?) of some: strong tractability better, since have exact polytime characterization of class
- ▶ From strong tractability algorithms, we get:
 - ▶ "yes" - we know $f(x) = \text{yes}$, and $x \in S$
 - ▶ "no" - we know $f(x) = \text{no}$, and $x \in S$
 - ▶ " $x \notin S$ " - we know $x \notin S$
- ▶ From a no-promise tractability algorithm, we get:
 - ▶ "yes" - we know $f(x) = \text{yes}$, but do not know if $x \in S$
 - ▶ "no" - we know $f(x) = \text{no}$, but do not know if $x \in S$
 - ▶ "decline" - we know $x \notin S$
- ▶ Do we care whether or not $x \in S$ if we already know the answer $f(x)$?

Which is better?

- ▶ Let $f : \Sigma^* \rightarrow \{\text{yes, no}\}$ be a decision problem, $S \subseteq \Sigma^*$ be a set of instances
- ▶ Observation: S is no-promise tractable \Leftrightarrow a superset $T \supseteq S$ is strongly tractable
 - ▶ Proof (\Leftarrow): run T -membership algorithm, then (if possible) decision algorithm
 - ▶ Proof (\Rightarrow): define $T = \{x \in \Sigma^* \mid M(x) \neq \text{decline}\}$. Algorithm M serves as both membership algorithm and decision algorithm.

- ▶ Personal perspective on some structural tractability results, and notions of tractability
- ▶ I want to suggest that explicitly defining these different flavors of tractability facilitates the understanding, interpretation, and comparison of the discussed results

1. What can we say about unbounded arity?
(Marx '09): tractability of bounded fractional hypertree width, defined by (Grohe & Marx '06)
2. What about bounded arity...
 - ▶ Under what conditions on \mathcal{A} is $\text{CSP}(\mathcal{A}, -)$ no-promise tractable?
 - ▶ Bounded treewidth sufficient, but can we do any better?
 - ▶ What about function problem of outputting a sat assignment if one exists?
 - ▶ Again, bounded treewidth sufficient, but can we do any better?
 - ▶ Grohe negative result applies to both problems