Detecting and Exploiting Subproblem Tractability

Christian Bessiere, Clement Carbonnel, Emmanuel Hebrard, George Katsirelos, Toby Walsh
To appear in Proc. of IJCAI 2013, Beijing, China
Suppose you are all familiar with constraint satisfaction?

- Given: variables each with a (finite) domain of possible values and a set of constraints (relations on allowed values for tuple of vars)
- Question: does there exist assignment of values to variables so that every constraint is satisfied?
Suppose you are all familiar with constraint satisfaction?

- Given: variables each with a (finite) domain of possible values and a set of constraints (relations on allowed values for tuple of vars)
- Question: does there exist assignment of values to variables so that every constraint is satisfied?

- Graph colouring: vars = nodes, values = colours, constraints = nodes connected by an edge have different colours
Background

- Suppose you are all familiar with constraint satisfaction?
  - Given: variables each with a (finite) domain of possible values and a set of constraints (relations on allowed values for tuple of vars)
  - Question: does there exist assignment of values to variables so that every constraint is satisfied?
    - Graph colouring: vars=nodes, values=colours, constraints=nodes connected by an edge have different colours
    - SAT: vars=Boolean, values=true/false, constraints=clauses
Background

- Suppose you are all *familiar* with constraint satisfaction?
  - Given: variables each with a (finite) domain of possible values and a set of constraints (relations on allowed values for tuple of vars)
  - Question: does there exist assignment of values to variables so that every constraint is satisfied?
    - Graph colouring: vars=nodes, values=colours, constraints=nodes connected by an edge have different colours
    - SAT: vars=Boolean, values=true/false, constraints=clauses
    - Scheduling: vars=jobs, values=start times, constraints=start times respect release times, start times+job lengths respect due dates, etc.
Motivation

- Lots of research on tractable constraint problems
  - Restricted language (e.g. 2SAT)
  - Restricted constraint structure (e.g. tree)
- But solvers often perform poorly on tractable problems
  - [Petke & Jeavons 2009]
  - Little research on detecting when a (sub)problem is tractable
Motivation

- Exploit (strong) backdoors into tractable subproblems
  - Identify some key variables (backdoor) that make problem intractable
  - Branch on these to give a tractable subproblem
  - FPT algorithm in size of backdoor

- Need to detect tractable subproblems
  - Not so much work on computational question of how to identify tractable subproblems!
Motivation

- Preliminary work
  - Our methods for identifying tractable subproblems have large polynomial cost
    - E.g. $O(d^6)$ and $O(d^7)$ time
  - May be able to offset this over many instances
  - Challenge will be to reduce costs!
Outline

- Identify tractable classes
- Exploit tractable classes
Outline

- Identify tractable classes
  - Detecting set of relations that admit majority polymorphism
  - Detecting set of relations that admit conservative Mal’tsev polymorphism
- Exploit tractable classes
Outline

- Identify tractable classes
  - Detecting set of relations that admit majority polymorphism
  - Detecting set of relations that admit conservative Mal’tsev polymorphism

- Exploit tractable classes
  - FPT algorithm for idempotent classes
  - FPT algorithm for conservative classes
  - NP-hardness when we don’t know backdoor and tractable subset of language
    - But FPT in $d+k+r$
Identifying tractable class

- Constraint problems are tractable if their relations are closed under majority polymorphisms
  
  [Jeavons et al 1997]

Language closed under majority polymorphism = generalization of 2-SAT and 0/1/all constraints
Identifying tractable class

- Constraint problems are tractable if their relations are closed under majority polymorphisms
  [Jeavons et al 1997]

- Constraint problems are tractable if their relations are closed under Mal’tsev polymorphisms
  [Bulatov & Dalmau 2006]

Language closed under majority polymorphism = generalization of 2-SAT and 0/1/all constraints

Language closed under Mal’tsev polymorphism = generalization of linear equations over a field
Identifying tractable class

- Constraint problems are tractable if their relations are closed under majority polymorphisms
  [Jeavons et al 1997]

- Constraint problems are tractable if their relations are closed under Mal’tsev polymorphisms
  [Bulatov & Dalmau 2006]

- Thm: Can decide if language is closed under majority polymorphism in $O(d^7t^4)$ time
  - Proof: Build an indicator problem, repeatedly apply SAC until failure/solution.
Identifying tractable class

- Constraint problems are tractable if their relations are closed under majority polymorphisms
  [Jeavons et al 1997]

- Constraint problems are tractable if their relations are closed under Mal’tsev polymorphisms
  [Bulatov & Dalmau 2006]

- Thm: Can decide if language is closed under majority polymorphism in $O(d^7t^4)$ time
  - Proof: Build an indicator problem, repeatedly apply SAC until failure/solution.

- Thm: Can decide if language is closed under a conservative Mal’tsev polymorphism in $O(d^6)$ time
  - Proof: Build a special indicator problem, repeatedly enforce AC, merge equals, remove redundant/universal constraints until failure/sol.
Exploiting tractable class

- Inspired by cycle cutset method [Dechter & Pearl 1987]
  - Instantiate variables to cut cycles
  - Then decide backtrack free with Directional AC

- Tractable subproblem based on structure of network
  - We now do much the same with a tractable language
Exploiting tractable class

- Idempotent class
  - I.e. fixing variables, we remain within the class

- Conservative class
  - Closed under all unary constraints
  - Stronger condition, smaller FPT algorithm
Exploiting tractable class

- Idempotent class
  - Let constraint relations \( C = C_1 + C_2 \)
  - Where \( C_2 \) closed under the idempotent polymorphism
  - Instantiate all \( m \) vars in \( C_1 \)
  - Leaves tractable subproblem made from \( C_2 \) and instantiations
  - FPT in \( d+m \)

- Conservative class
Exploiting tractable class

- Idempotent class
  - Let constraint relations $C = C_1 + C_2$
  - Where $C_2$ closed under the idempotent polymorphism
  - Instantiate all $m$ vars in $C_1$
  - Leaves tractable subproblem made from $C_2$ and instantiations
  - FPT in $d+m$

- Conservative class
  - Similar algorithm
  - FPT in strictly smaller parameter, $d+k$
  - Where $k$ is minimum vertex cover of primal graph of $C_1$
Exploiting tractable class

- Assumed so far that we know which relations make up the tractable subproblem
- What if we need to search simultaneously for a backdoor and the tractable relations?
Exploiting tractable class

- Assumed so far that we know which relations make up the tractable subproblem

- What if we need to search simultaneously for a backdoor and the tractable relations?
  
  Thm: NP-hard to decide if \( C \) partitions into \( C_1 + C_2 \) such that \( C_2 \) admits a conservative majority polymorphism and \( C_1 \) has a vertex cover of at most \( k \)
  
  - In fact, \( W[2] \)-hard in \( k \)
Exploiting tractable class

- Assumed so far that we know which relations make up the tractable subproblem

- What if we need to search simultaneously for a backdoor and the tractable relations?
  
  Thm: FPT in $d+k+r$ to decide if $C$ partitions into $C_1+C_2$ such that $C_2$ admits a conservative majority polymorphism and $C_1$ has a vertex cover of at most $k$
Empirical results

- Tested instances of 4th Constraint Solver Competition
  - Limited to those without globals
  - All instances put in extensional form
  - 191 series of instances
  - Tested for existence of subproblem closed under conservative majority polymorphism
Empirical results

- Tested instances of 4th Constraint Solver Competition
  - Limited to those without globals
  - All instances put in extensional form
  - 191 series of instances
  - Tested for existence of subproblem closed under conservative majority polymorphism

- Results
  - 135 series: exhausted 8Gb of memory
  - 40 series: large backdoor
  - But a few promising series
    - E.g. 5 prime series
    - ¼ had small backdoor (0 to 6 vars out of 100)
Conclusions

- We can exploit constraint problems that are *nearly* tractable
- Branch on backdoor into a tractable language
- For such methods to be useful, we need methods to identify tractable (sub)languages
  - Propose here two polynomial methods to identify language closed under a majority polymorphism, and under a conservative Mal’tsev polymorphism
- Computing a backdoor into a language closed under a conservative majority polymorphism is $W[2]$-hard in $k$, but FPT in $d+k+r$
Questions?

- PS we’re recruiting PhD students and a PostDoc
- Good student = guaranteed funding
- Shortly after graduating -> Australian citizen