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
- Not enough to know it is tractable [Petke & Jeavons 2009]
- Detect membership to a tractable and apply the proper algorithm
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
- Not enough to know it is tractable [Petke & Jeavons 2009]
- Detect membership to a tractable and apply the proper algorithm

Problems might be *nearly-tractable*
Solid edges: “easy” constraints / Dashed edges: “hard” constraints
Solid edges: “easy” constraints / Dashed edges: “hard” constraints
Branch on $d$
Solid edges: “easy” constraints / Dashed edges: “hard” constraints
Branch on $d$
  - Remove $(a, d)$ and $(d, g)$
Motivation

- Identify a (hopefully) small number of variables
- Branch on these to give tractable subproblems
Identify a (hopefully) small number of variables
Branch on these to give tractable subproblems
find a backdoor [Williams et al. 2003]
Detecting tractable classes

Exploiting tractable classes
Contributions

- Detecting tractable classes
  - Detecting set of relations closed by a *majority polymorphism*
  - Detecting set of relations closed by a *Mal'tsev polymorphism*
- Exploiting tractable classes
Contributions

- Detecting tractable classes
  - Detecting set of relations closed by a *majority polymorphism*
  - Detecting set of relations closed by a *Mal'tsev polymorphism*

- Exploiting tractable classes
  - If given a tractable sublanguage: *easy*
  - Otherwise: *hard* (but there are positive results!)
Polymorphisms and Tractability

Constraint problems are tractable if their relations are closed under *majority* polymorphisms [Jeavons et al 1997]
- Generalization of 2-SAT and 0/1/all constraints

Constraint problems are tractable if their relations are closed under *Mal’tsev* polymorphisms [Bulatov & Dalmau 2006]
- Generalization of linear equations over a field
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$R:$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly, it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

$R :$
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

$$R: \begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}$$
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$R$:
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$

Example

- $f(x, y) = (x + y \mod 2)$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$R:$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$.
Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$.

Example

$f(x, y) = (x + y \mod 2)$

\[ R : \begin{array}{ccc}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array} \]
Polymorphism

- Operation $f$ maps $m$ values $v_1, \ldots, v_m$ to another value $f(v_1, \ldots, v_m)$
- Similarly it maps $m$ tuples $\tau_1, \ldots, \tau_m$ to another tuple $f(\tau_1, \ldots, \tau_m)$
- $f$ is a polymorphism of $R$ iff applying $f$ to tuples of $R$ does not produce new tuples

Example

- $f(x, y) = (x + y \mod 2)$

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Detecting Majority

- $f$ is a **majority** operation iff $f(v, v, w) = f(v, w, v) = f(w, v, v) = v$
Detecting Majority

- $f$ is a *majority* operation iff $f(v, v, w) = f(v, w, v) = f(w, v, v) = v$

**Theorem 1**

Majority polymorphisms can be detected in polynomial time
Detecting Majority

- $f$ is a *majority* operation iff $f(v, v, w) = f(v, w, v) = f(w, v, v) = v$
- Polymorphisms of $P$ are solutions of its *indicator problem* [Jeavons et al. 1997]
  - $P$ and its indicator problem share the same set of relations
- A CSP closed under a majority polymorphism is solved backtrack-free by maintaining *singleton arc consistency* [Chen et al. 2013]

**Theorem 1**

Majority polymorphisms can be detected in polynomial time
Detecting Majority

- \( f \) is a \textit{majority} operation iff \( f(v, v, w) = f(v, w, v) = f(w, v, v) = v \)
- Polymorphisms of \( P \) are solutions of its \textit{indicator problem} [Jeavons et al. 1997]
  - \( P \) and its indicator problem share the same set of relations
- A CSP closed under a majority polymorphism is solved backtrack-free by maintaining \textit{singleton arc consistency} [Chen et al. 2013]

Theorem 1

Majority polymorphisms can be detected in polynomial time

Proof

- Run \textit{maintain SAC} on the indicator problem
  - If success, we obtain a solution, hence a polymorphism of \( P \)
  - If at any point there is a fail, we can deduce that the indicator problem has no majority polymorphism
Detecting (conservative) Mal’tsev

- $f$ is a *Mal’tsev* operation iff $f(v, w, w) = f(w, w, v) = v$
- $f$ is *conservative* iff $f(u, v, w) \in \{u, v, w\}$

**Theorem 2**

Conservative Mal’tsev polymorphisms can be detected in polynomial time on binary relations
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism

Question: is $P$ satisfiable?
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism

Question: is $P$ satisfiable?

- Backdoor of size $k$: search tree of size $d^k$
Exploiting Tractability

- Input: A CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, a set $B$ such that $\mathcal{C} \setminus B$ has a polymorphism
- Question: is $P$ satisfiable?
  - Backdoor of size $k$: search tree of size $d^k$

Fixed Parameter Tractability

- Given a problem $A$ and a parameter $k$
- **Fixed Parameter Tractable** (FPT) iff there exists an algorithm which complexity is in $O(f(k)P(n))$
  - Any computable function $f$ of $k$ (for ex. $2^k$)
  - A polynomial $P(n)$ in the size of the problem $n$
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism

Question: is $P$ satisfiable?

- Backdoor of size $k$: search tree of size $d^k$

Fixed Parameter Tractability

- Given a problem $A$ and a parameter $k$
- **Fixed Parameter Tractable** (FPT) iff there exists an algorithm which complexity is in $O(f(k)P(n))$
  - Any computable function $f$ of $k$ (for ex. $2^k$)
  - A polynomial $P(n)$ in the size of the problem $n$
- $W[m]$-hardness by reduction from a $W[m]$-hard pair $A', k'$
Input: A CSP \( P = (X, D, C) \), a set \( B \) such that \( C \setminus B \) has a polymorphism.

Question: is \( P \) satisfiable?
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism

Question: is $P$ satisfiable?
Input: A CSP \( P = (\mathcal{X}, \mathcal{D}, \mathcal{C}) \), a set \( B \) such that \( \mathcal{C} \setminus B \) has a polymorphism.

Question: Is \( P \) satisfiable?

The polymorphism is conservative.
Exploiting Tractability

- Input: A CSP \( P = (\mathcal{X}, \mathcal{D}, \mathcal{C}) \), a set \( B \) such that \( \mathcal{C} \setminus B \) has a polymorphism
- Question: is \( P \) satisfiable?

The polymorphism is \textit{conservative}

\[ f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\} \]
**Exploiting Tractability**

- **Input:** A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism.
- **Question:** is $P$ satisfiable?

The polymorphism is *conservative*

- $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$
- Unary relations are closed under any conservative operation.
Exploiting Tractability

- Input: A CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, a set $B$ such that $\mathcal{C} \setminus B$ has a polymorphism
- Question: is $P$ satisfiable?

The polymorphism is conservative

- $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$
- Unary relations are closed under any conservative operation
- Eliminating a constraint $\leftrightarrow$ assigning all (but one) of its variables
Exploiting Tractability

- Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism
- Question: is $P$ satisfiable?

The polymorphism is conservative

- $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$
- Unary relations are closed under any conservative operation
- Eliminating a constraint $\leftrightarrow$ assigning all (but one) of its variables
- Backdoor: vertex cover of the primal graph of $B$ $[O(2^k)]$
**Exploiting Tractability**

- **Input:** A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism
- **Question:** is $P$ satisfiable?

The polymorphism is *conservative*

- $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$
- Unary relations are closed under any conservative operation
- Eliminating a constraint $\leftrightarrow$ assigning all (but one) of its variables
- Backdoor: *vertex cover* of the primal graph of $B$ [$O(2^k)$]
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism.

Question: is $P$ satisfiable?

The polymorphism is \textit{conservative}.

- $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$
- Unary relations are closed under any conservative operation.
- Eliminating a constraint $\iff$ assigning all (but one) of its variables.
- Backdoor: \textit{vertex cover} of the primal graph of $B$ $[O(2^k)]$.
Input: A CSP \( P = (X, D, C) \), a set \( B \) such that \( C \setminus B \) has a polymorphism

Question: is \( P \) satisfiable?

The polymorphism is \textit{conservative}

- \( f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\} \)
- Unary relations are closed under any conservative operation
- Eliminating a constraint \( \iff \) assigning all (but one) of its variables
- Backdoor: \textit{vertex cover} of the primal graph of \( B \) \([O(2^k)]\)
- Explore a \( d^k \) search tree
  - FPT in \( d + k \) (domain size and size of the vertex cover of \( B \))
Input: A CSP $P = (X, D, C)$, a set $B$ such that $C \setminus B$ has a polymorphism

Question: is $P$ satisfiable?

The polymorphism is *idempotent*

- $f(v, \ldots, v) = v$
- Eliminating a constraint $\iff$ assigning all its variables
- Backdoor: all variables of $B$
Exploiting Tractability

- Input: A CSP \( P = (\mathcal{X}, \mathcal{D}, \mathcal{C}) \), a set \( B \) such that \( \mathcal{C} \setminus B \) has a polymorphism
- Question: is \( P \) satisfiable?

The polymorphism is **idempotent**

- \( f(v, \ldots, v) = v \)
- Eliminating a constraint \( \leftrightarrow \) assigning all its variables
- Backdoor: all variables of \( B \)
What if we don’t know $B$?

Finding a min backdoor to majority: **Partition-Majority-CSP**

- particular case: $f$ is conservative majority, $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is binary
- compute a subset $B$ of $\mathcal{C}$ such that $\mathcal{C} \setminus B$ is closed under some majority operation and the vertex cover of $B$’s graph is minimum
What if we *don’t* know $B$?

- Finding a min backdoor to *majority*: \textsc{Partition-Majority-CSP}
  - particular case: $f$ is conservative majority, $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is binary
  - compute a subset $B$ of $\mathcal{C}$ such that $\mathcal{C} \setminus B$ is closed under some majority operation and the vertex cover of $B$’s graph is minimum

**Theorem 5**

\textsc{Partition-Majority-CSP} is NP-hard
Identifying Tractable Subproblems

- What if we *don’t* know $B$?
  - Finding a min backdoor to *majority*: Partition-Majority-CSP
    - particular case: $f$ is conservative majority, $P = (X, D, C)$ is binary
    - compute a subset $B$ of $C$ such that $C \setminus B$ is closed under some majority operation and the vertex cover of $B$'s graph is minimum

**Theorem 5**

- **Partition-Majority-CSP** is NP-hard

**Theorem 6**

- **Partition-Majority-CSP** is $W[2]$-hard when the parameter is the size of the vertex cover
What if we *don’t* know $B$?

- Finding a min backdoor to *majority*: **Partition-Majority-CSP**
  - particular case: $f$ is conservative majority, $P = (X, D, C)$ is binary
  - compute a subset $B$ of $C$ such that $C \setminus B$ is closed under some majority operation and the vertex cover of $B$’s graph is minimum

**Theorem 5**

**Partition-Majority-CSP** is NP-hard

**Theorem 6**

**Partition-Majority-CSP** is W[2]-hard when the parameter is the size of the vertex cover

**Theorem 7**

**Partition-Majority-CSP** is FPT for **domain + cover + language**
Empirical Results

- We used benchmarks from the 4th CSP Solver Competition
  - Are there *almost-majority-closed* problems?
  - If so, can we compute small backdoors in *practice*?
Empirical Results

- We used benchmarks from the 4th CSP Solver Competition
  - Are there *almost-majority-closed* problems?
  - If so, can we compute small backdoors in *practice*?

Algorithm

- Explore the possible partitions of the language of relations (branch & bound)
- Given a partition we compute the minimal vertex cover (to be used to branch & bound)
  - Cache partitions that block majority (nogoods)
  - Efficient algorithm for SAC (SAC3-SDS) [Bessiere et al. 2008]
  - Efficient algorithm for vertex cover [Balasubramanian et al. 1998]
Empirical Results

- We used benchmarks from the 4th CSP Solver Competition
  - Are there *almost-majority-closed* problems?
  - If so, can we compute small backdoors in *practice*?

- Out of 191 instances put in extensional form:
  - On 135 instances, the indicator problem is too large
  - On 40 instances, the backdoor is large (trivial)
Empirical Results

- We used benchmarks from the 4th CSP Solver Competition
  - Are there *almost-majority-closed* problems?
  - If so, can we compute small backdoors in *practice*?

- Out of 191 instances put in extensional form:
  - On 135 instances, the indicator problem is too large
  - On 40 instances, the backdoor is large (trivial)
  - On a serie of 5 prime instances we found small backdoors (0 to 6 variables out of 100)
  - On 1 driverlogw instance we found a non-trivial backdoor (22 variables out of 71)
Empirical Results

We used benchmarks from the 4th CSP Solver Competition

- Are there almost-majority-closed problems? A few
- If so, can we compute small backdoors in practice? Sometimes

Out of 191 instances put in extensional form:

- On 135 instances, the indicator problem is too large
- On 40 instances, the backdoor is large (trivial)
- On a serie of 5 prime instances we found small backdoors (0 to 6 variables out of 100)
- On 1 driverlogw instance we found a non-trivial backdoor (22 variables out of 71)
We can exploit constraint problems that are *nearly* tractable
- Compute a *tractable sub-language*
  - *Detect membership* efficiently
- Compute a *backdoor* to this sub-language
- Branch on the backdoor

Computing a majority-backdoor is W[2]-hard in the vertex cover size, however FPT in $d + k + r$
- Domain size, vertex cover size, language cardinality
Conclusion

- We can exploit constraint problems that are *nearly* tractable
  - Compute a *tractable sub-language*
    - *Detect membership* efficiently
  - Compute a *backdoor* to this sub-language
  - Branch on the backdoor
- Computing a majority-backdoor is W[2]-hard in the vertex cover size, however FPT in $d + k + r$
  - Domain size, vertex cover size, language cardinality

Questions?
Detecting (conservative) Mal’tsev

- $f$ is a **Mal’tsev** operation iff $f(v, w, w) = f(w, w, v) = v$
Detecting (conservative) Mal’tsev

- $f$ is a Mal’tsev operation iff $f(v, w, w) = f(w, w, v) = v$

Theorem 2

Conservative Mal’tsev polymorphisms can be detected in polynomial time on binary relations
Detecting (conservative) Mal’tsev

- \( f \) is a \textit{Mal’tsev} operation iff \( f(v, w, w) = f(w, w, v) = v \)
- \( f \) is \textit{conservative} iff \( f(u, v, w) \in \{u, v, w\} \) (vars of the indicator problem)
- If \( R \) is binary and Mal’tsev, then it is a set of bicliques \cite{Bulatov2002}

**Theorem 2**

Conservative Mal’tsev polymorphisms can be detected in polynomial time on binary relations
Detecting (conservative) Mal’tsev

- \( f \) is a Mal’tsev operation iff \( f(v, w, w) = f(w, w, v) = v \)
- \( f \) is conservative iff \( f(u, v, w) \in \{u, v, w\} \) (vars of the indicator problem)
- If \( R \) is binary and Mal’tsev, then it is a set of bicliques [Bulatov 2002]

Theorem 2
Conservative Mal’tsev polymorphisms can be detected in polynomial time on binary relations

Proof: only three cases

```
1 - 1 - 1
|   |   |   |
2 - 2 - 2
|   |   |   |
3 - 3 - 3
```

```
1 - 1 - 1
|   |   |   |
2 - 2 - 2
|   |   |   |
3 - 3 - 3
```

```
1 - 1 - 1
|   |   |   |
2 - 2 - 2
|   |   |   |
3 - 3 - 3
```
Given a set of relations $\Gamma$:

- CSP which solutions are polymorphisms of $\Gamma$
- A variable for each $m$-tuple of values
  - represents the image of this $m$-tuple by the polymorphism
- For each relation $R \in \Gamma$, and for each permutation of $m$ tuples $\tau_1, \ldots, \tau_m \in R$ we post the constraint $R$ on the image variables
Given a set of relations $\Gamma$:
- CSP which solutions are polymorphisms of $\Gamma$
- A variable for each $m$-tuple of values represents the image of this $m$-tuple by the polymorphism
- For each relation $R \in \Gamma$, and for each permutation of $m$ tuples $\tau_1, \ldots, \tau_m \in R$ we post the constraint $R$ on the image variables

\[
\begin{bmatrix}
1 & X_{11} & 1 & X_{10} & 1 & X_{11} \\
1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
Given a set of relations $\Gamma$:

- CSP which solutions are polymorphisms of $\Gamma$
- A variable for each $m$-tuple of values
  - represents the image of this $m$-tuple by the polymorphism
- For each relation $R \in \Gamma$, and for each permutation of $m$ tuples $\tau_1, \ldots, \tau_m \in R$ we post the constraint $R$ on the image variables

\[ R(x_{11}, x_{10}, x_{11}) \]