Introduction

Operations and Properties
  Minimalization
  Consistencies and Backbone Variables

Complexity
  Previous Results
  Our Results

Creating Minimal CSP instances
  Generator
  Empirical Tests

Conclusion
Constraint Satisfaction Problem

A CSP instance:

- A set $V = \{v_1, \ldots, v_n\}$ of $n$ variables.
- A set $\{A_1, \ldots, A_n\}$ of $n$ domains. For all $i$, $A_i$ contains the $d_i$ possible values that $v_i$ can take.
- A set $C$ of constraints specifying the $k$-tuples of values that are allowed, and the ones that are forbidden, with $k$ being the arity of the instance.

Question:
Is there a solution, that is a set of $n$ values, one in each domain, that satisfies all the constraints?
Conventions

- We also consider trivial constraints. There are $\binom{n}{k}$ constraints. The constraint hypergraph is complete. Each $k$-tuple of values over $k$ different domains is either allowed or forbidden.
- An allowed $k$-tuple of values is a compatible $k$-tuple. A forbidden $k$-tuple of values is an incompatible $k$-tuple.
- A tuple of more than $k$ values which does not contain any incompatible $k$-tuple is a partial solution.
- A subset of a complete solution is completetable.
• Arity is 2.
• 3 variables.
• 2 values in each domain.
• 3 constraints $\rightarrow$ 12 couples of values: 7 are compatible (lines), 5 are incompatible.
Example

Two solutions:

- \((a, d, f)\)
- \((b, c, e)\)
A Minimal CSP instance:

- A CSP instance.
- Every compatible $k$-tuple is completable.
NOT a Minimal CSP instance

(b, f) is not completable.
A Minimal CSP instance

Three solutions:

- \((a, d, f)\): completes \((a, d)\), \((a, f)\) and \((d, f)\).
- \((b, c, e)\): completes \((b, c)\), \((b, e)\) and \((c, e)\).
- \((b, c, f)\): completes \((b, f)\) and \((c, f)\).
What is the Problem?

Decision Problem
Is there a solution?

Search Problem
Exhibit a solution.
What is the Problem?

Search Problem
Exhibit a solution.
When not using the Minimal CSP

- Customer: I want a car.
When not using the Minimal CSP

- **Customer:** I want a car.
- **Salesperson:** No problem, we have a lot of cars. Some of them have automatic transmission, the others have manual transmission. Which one do you prefer?
When not using the Minimal CSP

- Customer: I want a car.
- Salesperson: No problem, we have a lot of cars. Some of them have automatic transmission, the others have manual transmission. Which one do you prefer?
- Customer: Automatic.
When not using the Minimal CSP

- Customer: I want a car.
- Salesperson: No problem, we have a lot of cars. Some of them have automatic transmission, the others have manual transmission. Which one do you prefer?
- Customer: Automatic.
- Salesperson: Some of our cars are green, the rest are orange. Which color do you prefer?
When not using the Minimal CSP

- Customer: I want a car.
- Salesperson: No problem, we have a lot of cars. Some of them have automatic transmission, the others have manual transmission. Which one do you prefer?
- Customer: Automatic.
- Salesperson: Some of our cars are green, the rest are orange. Which color do you prefer?
- Customer: Green.
When not using the Minimal CSP

- Customer: I want a car.
- Salesperson: No problem, we have a lot of cars. Some of them have automatic transmission, the others have manual transmission. Which one do you prefer?
- Customer: Automatic.
- Salesperson: Some of our cars are green, the rest are orange. Which color do you prefer?
- Customer: Green.
- Salesperson: Sorry, we do not have any green car with automatic transmission.
Configuration Problems

- A car salesperson.
- Several cars to sell.
- Cars are characterized by their type of transmission, color, engine, price range.
- Several possibilities for each of the characteristics (automatic or manual, green or orange, . . .).
- Customers can make choices. Each possible sequence of choices must be associated to at least one car.
Configuration Problems

- A seller.
- Several items to sell.
- Several options characterize the items.
- Several choices for each characteristic.
- Each possible sequence of choices must be associated to at least one item.

On the Minimal Constraint Satisfaction Problem
Introduction
Configuration Problems

• A seller.
• Several items to sell.
• Several options characterize the items.
• Several choices for each characteristic.
• Each possible sequence of choices must be associated to at least one item.

Buying a product online

• Cars (make, color, engine, . . . )
• Flights (airline, destination, number of stops, . . . )
• Computers (brand, operating system, video card, . . . )
Configuration Problems

- A seller $\leftrightarrow$ A CSP instance
- Several items to sell $\leftrightarrow$ Several solutions to the instance
- Characteristics $\leftrightarrow$ Variables
- Several possible choices for each option $\leftrightarrow$ Several values for each variable
- Each possible sequence of choices must be associated to at least one item $\leftrightarrow$ Each partial solution must be completable
When using the Minimal CSP

Queries that we can answer in polynomial time:

- Is there at least one green automatic car? Decision, $O(d)$
- At what time is the earliest flight from Cork to London? Optimization, $O(d)$
- Is there a computer such that \((\text{hard disk memory})/100+\text{RAM} \geq 16\text{GB}\)? Decision, $O(d^2)$

No need to solve the entire instance, just look at one constraint.
A CSP instance with empty domains is minimal.
A CSP instance with only incompatible $k$-tuples is minimal.

We only study Minimal CSP instances with non-empty domains and such that each value belongs to at least one compatible $k$-tuple.
Introduction

Operations and Properties
  Minimalization
  Consistencies and Backbone Variables

Complexity
  Previous Results
  Our Results

Creating Minimal CSP instances
  Generator
  Empirical Tests

Conclusion
Definition
To *minimalize* a CSP instance is to convert all non-completable compatibilities into incompatibilities.
Minimalizing a CSP instance

Definition
To *minimalize* a CSP instance is to convert all non-completable compatibilities into incompatibilities.

Lemma
All CSP instances are minimalizable. The result of minimalization is unique.

Proof (Uniqueness)
Converting a compatibility into an incompatibility does not change the set of solutions.
Example

Original instance

- Two solutions $(a, d, f)$ and $(b, c, e)$.
- $(b, f)$ is compatible and not completable.
Example

Minimalized instance

- Two solutions \((a, d, f)\) and \((b, c, e)\).
- Every compatible couple of values is completable.
Proposition
Minimalizing a CSP instance is NP-hard.

Proof
• Reduction from the CSP.
• Get a CSP instance as input. Minimalize it.
• If something remains, return yes. Otherwise, return no.

Corollary
Minimalizing is NP-hard for every set of CSP instances that is NP-Complete in the general decision problem.
Definition
A CSP instance $I$ is embedded in another CSP instance $I'$ if $I$ and $I'$ share the same variables and domains, and every compatibility in $I$ is a compatibility in $I'$.

Lemma
Every CSP instance is embedded in a Minimal CSP instance. The embedding is not unique, even for a given number of compatibilities.

Proof (Existence)
Convert all incompatibilities into compatibilities.
Example

Original instance

- Two solutions \((a, d, f)\) and \((b, c, e)\).
- \((b, f)\) is compatible and not completable.
Example 1

Embedding instance 1

- Three solutions \((a, d, f), (b, c, e)\) and \((b, c, f)\).
- Every compatible couple of values is completable.
Example 2

Embedding instance 2

- Three solutions \((a, d, f), (b, c, e)\) and \((b, d, f)\).
- Every compatible couple of values is completable.
(i, j)-consistency

Definition (Freuder 1985)
A CSP instance I is \((i, j)\)-consistent if for any set \(V_1\) of \(i\) variables, and any set \(V_2\) of \(j\) variables such that \(V_1 \cap V_2 = \emptyset\), any compatible \(i\)-tuple on \(V_1\) can be extended to a compatible \((i + j)\)-tuple on \(V_1 \cup V_2\).

Examples
- Arc consistency \(\iff\) \((1,1)\)-consistency
- Path consistency \(\iff\) \((2,1)\)-consistency
- Path inverse consistency \(\iff\) \((1,2)\)-consistency
Lemma
Let $I$ be a $k$-ary Minimal CSP instance.
$\forall i \leq k, \forall j \leq (n - i), I$ is $(i, j)$-consistent.

Proof/Remark
It is actually an alternate definition of minimality.
**i-wise consistent**

**Definition**
A CSP instance \( I \) is \( i \)-wise consistent if for any set 
\( C_0 = \{ c_1, c_2, \ldots, c_i \} \) of \( i \) constraints of \( I \), for any \( 1 \leq j \leq i \), any compatible \( k \)-tuple on \( c_j \) can be extended to a compatible tuple on \( C_0 \).

**Lemma**
Let \( I \) be a Minimal CSP instance. 
\( \forall i, I \) is \( i \)-wise consistent.
Minimal CSP and Backbone Variables

Definition
Let \( I \) be a CSP instance and \( v \) a variable of \( I \). \( v \) is a **backbone variable** if it takes the same value in all solutions to \( I \).

Lemma
If \( I \) is a Minimal CSP instance, then the only backbone variables of \( I \) are single-valued variables.

Proof
In a Minimal CSP, every value belongs to all solutions.
Introduction

Operations and Properties
  Minimalization
  Consistencies and Backbone Variables

Complexity
  Previous Results
  Our Results

Creating Minimal CSP instances
  Generator
  Empirical Tests

Conclusion
The Minimal CSP

• Everything is in a solution.
• Consistent for many different notions of consistency.
• Can be used to quickly answer many queries.

⇒ Minimalization: very useful, albeit hard.
   (Not the main topic of the seminar)

The Question
Since Minimal CSP instances look so easy, why study their complexity?
Because they are hard.

Theorem (Gottlob 2011)
The Minimal Constraint Satisfaction Problem is NP-hard.
Proof for Binary Case

- Reduction from 3-SAT.
- \( I \) a SAT instance, \( n \) clauses, 3 literals in each clause.
- \( v \) a variable \( \rightarrow v_1, v_2, v_3, v_4, v_5 \) five variables.
- \( l \) a literal in a clause \( \rightarrow \) all ten combinations of \( l_h \lor l_i \lor l_j \).
- \( c \) a clause with 3 literals
  \[ \rightarrow 10^3 = 1000 \text{ clauses with } 3 \times 3 = 9 \text{ literals}. \]
- SAT instance, 1000\( n \) clauses, 9 literals in each clause
  \( \rightarrow \) CSP instance, 1000\( n \) variables, 9 values in each domain.
- CSP instance is either minimal or unsatisfiable.
Proof for General Arity $k$

- Reduction from 3-SAT.
- SAT instance, $n$ clauses, 3 literals in each clause.
  $\rightarrow$ CSP instance, $(\binom{2k+1}{k})^3 \times n$ variables, $3(k+1)$ values in each domain.
- CSP instance is either minimal or unsatisfiable.
Corollary

Minimal CSP is NP-hard, even when bounding the size of the domains by $3(k + 1)$. 
Theorem
For binary CSP instances such that the treewidth of the constraint graph is equal to $h$:
$(i, n - i)$-consistency can be established in $O((n \times d^{h+1})^{i+1})$.

Corollary
Minimalizing binary CSP instances can be done in $O((n \times d^{h+1})^3)$, with $h$ the treewidth of the constraint graph.
Bounded Domain Size

Old Bound (Gottlob 2011)

\[ 3(k + 1) \rightarrow 9 \]
Bounded Domain Size

Old Bound (Gottlob 2011)
3(k + 1) → 9

Proposition
Minimal CSP is NP-hard, even when bounding the size of the domains by \( d \geq 3 \).

Proof

- Reduction from Minimal CSP.
- One domain \( \{a_1, a_2, \ldots, a_m\} \) of size \( m \)
  \( \rightarrow \) two domains \( \{a_1, a_2, x\} \) and \( \{\overline{x}, a_3, \ldots, a_m\} \)
  of size 3 and \( m - 1 \).
Boolean Domains

Proposition
Boolean Minimal CSP is NP-hard
**Proposition**

Boolean Minimal CSP is NP-hard

**Proof**

- Reduction from 3-Minimal CSP.
- One domain \(\{a_1, a_2, a_3\}\) of size 3
  \(\rightarrow\) three domains \(\{a_1, \overline{a_1}\}\), \(\{a_2, \overline{a_2}\}\) and \(\{a_3, \overline{a_3}\}\) of size 2.
Boolean Domains

**Proposition**
Boolean Minimal CSP is NP-hard, even when the arity is bounded by $k \geq 3$.

**Proof**
- Reduction from 3-Minimal CSP.
- One domain $\{a_1, a_2, a_3\}$ of size 3
  $\rightarrow$ three domains $\{a_1, \overline{a_1}\}$, $\{a_2, \overline{a_2}\}$ and $\{a_3, \overline{a_3}\}$ of size 2.
Binary Boolean Minimal CSP

Proposition
Binary Boolean Minimal CSP is polynomial.

Proof
Subset of Binary Boolean CSP, which is polynomial from 2-SAT.
The Minimal CSP Dichotomy Theorem

A $k$-ary Minimal CSP when the size of the domains is bounded by $d$ is NP-hard if and only if ($d \geq 3$ or ($d = 2$ and $k \geq 3$)).

<table>
<thead>
<tr>
<th>$d$</th>
<th>$k$</th>
<th>$2$</th>
<th>$\geq 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>tractable</td>
<td>tractable</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>tractable</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td></td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
</tbody>
</table>
Minimal CSP and general CSP

Corollary
Minimal CSP and the general CSP are NP-hard for the exact same values of $d$ and $k$. 
Introduction

Operations and Properties
  Minimalization
  Consistencies and Backbone Variables

Complexity
  Previous Results
  Our Results

Creating Minimal CSP instances
  Generator
  Empirical Tests

Conclusion
Theorem (Gottlob 2011)
Minimal CSP is NP-hard.

Proof
SAT instance, $n$ clauses, 3 literals in each clause
→ CSP instance, $1000n$ variables, 9 values in each domain
Theorem (Gottlob 2011)
Minimal CSP is NP-hard.

Proof
SAT instance, \( n \) clauses, 3 literals in each clause
\[ \rightarrow \]
CSP instance, 1000\( n \) variables, 9 values in each domain

Question
Are there hard minimal instances of reasonable size?
Theorem (Gottlob 2011)
Minimal CSP is NP-hard.

Proof
SAT instance, \( n \) clauses, 3 literals in each clause
\( \rightarrow \) CSP instance, \( 1000n \) variables, 9 values in each domain

Question
Are there hard minimal instances of reasonable size?

Remark
It would be nice to have a way to efficiently generate random, reasonable, parameterizable, minimal instances.
Definition
A $k$-ary CSP instance with $n$ variables and $d$ values in each domain is of size $(d, n, k)$.

Definition
The tightness of a $k$-ary CSP instance is the number of its incompatible $k$-tuples divided by its total number of $k$-tuples.

What we want
- Input $d$, $n$, $k$, $t$.
- Output: A random Minimal instance of size $(d, n, k)$ and tightness $t$. 
General CSP
Can generate constraint by constraint, even tuple by tuple.
Can edit an individual constraint, or even an individual tuple.

Minimal CSP
Modifying even a single tuple jeopardizes minimality.
All constraints must be considered at once.
A Few Ideas

Reduction from other CSP instances

- Far too big instances!
- “Control” over the size, but not over the tightness.
A Few Ideas

Reduction from other CSP instances

- Far too big instances!
- “Control” over the size, but not over the tightness.

Minimalization

- NP-hard $\implies$ not efficient.
- No control over the size or tightness.
Definition
A **bare Minimal CSP instance** is a CSP instance where all constraints are equality constraints.

Lemma
Bare minimal instances are the minimal instances with the lowest possible tightness.
The Generator

**Input:** $d, n, k$ integers, and $t \in [0, 1]$.

**Output:** Minimal CSP instance of size $(d, n, k)$, with a tightness (approximately) equal to $t$.

**Algorithm**

1. Generate a bare Minimal CSP instance $I$ of size $(d, n, k)$.
2. $t_i \leftarrow$ tightness of $I$.
3. While ($t_i > t$)
   3.1 Add solution to $I$.
   3.2 $t_i \leftarrow$ tightness of $I$.
4. Return $I$. 
• The first $n$-tuple in the lexicographical order is a solution.
• For low values of $t$, the algorithm does not always terminate.
The Improved Generator

Input: $d, n, k$ integers, and $t \in [0, 1]$.

Output: Minimal CSP instance of size $(d, n, k)$, with a tightness (approximately) equal to $t$.

Algorithm

1. Generate a bare Minimal CSP instance $I$ of size $(d, n, k)$.
2. Randomize the ordering of the values.
3. $t_i \leftarrow$ tightness of $I$.
4. While ($t_i > t$)
   4.1 $C \leftarrow$ most constrained constraint of $I$.
   4.2 $A \leftarrow$ random incompatible $k$-tuple from $C$.
   4.3 Add solution containing $A$ to $I$.
   4.4 $t_i \leftarrow$ tightness of $I$.
5. Return $I$. 
Final Tightness

Worst case

- Each iteration: one more compatibility in each constraint.
- Final deviation: $\frac{1}{d^k}$.

Average case

- Each iteration: (current tightness) more compatibility in each constraint.
- Final deviation: $\frac{t}{d^k}$. 
Question

Is the generator refined enough to detect hard Minimal CSP instances?
Question
Is the generator refined enough to detect hard Minimal CSP instances?

Software

- Numberjack.
- Solvers:
  - Binary instances: Mistral.
  - 3+-ary instances: MiniSat.
- Stuff computed:
  - Is there a solution?
  - Number of nodes needed to find a solution.

⇒ Runtimes do not matter.
(a) Size (10, 76, k = 2)
On the Minimal Constraint Satisfaction Problem
Empirical Tests

(b) Size \((15, 50, k = 2)\)
(c) Size $(27, 28, k = 2)$
(d) Size \((6, 24, k = 3)\)
What Correlates Tightness and Hardness?

Phase Transitions

- CSP: number of satisfiable instances.
- Satisfiable CSP: number of backbone variables.
Definition
A \textit{p-step instance} of a CSP instance \( I \) is an instance obtained from \( I \) after making \( p \) assignments and propagating \((1, k - 1)\)-consistency after each assignment.
What Correlates Tightness and Hardness?

Phase Transitions

- CSP: number of satisfiable instances.
- Satisfiable CSP: number of backbone variables.
- Minimal CSP: number of satisfiable \((k + 1)\)-step instances.

Why \(k + 1\)?

All \(k\)-step instances are satisfiable.
(e) Size $(10, 76, k = 2)$
(f) Size $(15, 50, k = 2)$

On the Minimal Constraint Satisfaction Problem
Empirical Tests

10/14/2015  Slide 61/69
(g) Size (27, 28, $k = 2$)
(h) Size (6, 24, k = 3)
Conjecture

- In a Minimal CSP instance, the first $k$ choices are free.
- The satisfiability of $(k + 1)$-step instances represents the probability of making a right $(k + 1)^{th}$ choice.
- A lot of compatibilities means a large search space.

$\Rightarrow$ The hardest instances occur at low number of satisfiable $(k + 1)$-step instances and high number of compatibilities.
Introduction

Operations and Properties
  Minimalization
  Consistencies and Backbone Variables

Complexity
  Previous Results
  Our Results

Creating Minimal CSP instances
  Generator
  Empirical Tests

Conclusion
Our Contributions: Complexity Theorem

The Minimal CSP Dichotomy Theorem

$k$-ary Minimal CSP when the size of the domains is bounded by $d$ is NP-hard if and only if ($d \geq 3$ or ($d = 2$ and $k \geq 3$)).

<table>
<thead>
<tr>
<th>$d$</th>
<th>$k$</th>
<th>2</th>
<th>$\geq 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>tractable</td>
<td>tractable</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>tractable</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td></td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
</tbody>
</table>
Our Contributions: Generator

Input: $d, n, k$ integers, and $t \in [0, 1]$.
Output: Minimal CSP instance of size $(d, n, k)$, with a tightness equal to $t' \in [t - \frac{1}{dk}, t]$.

Algorithm

1. Generate a bare Minimal CSP instance $I$ of size $(d, n, k)$.
2. Randomize the ordering of the values.
3. $t_i \leftarrow$ tightness of $I$.
4. While ($t_i > t$)
   4.1 $C \leftarrow$ most constrained constraint of $I$.
   4.2 $A \leftarrow$ random incompatible $k$-tuple from $C$.
   4.3 Add solution containing $A$ to $I$.
   4.4 $t_i \leftarrow$ tightness of $I$.
5. Return $I$. 
(Very informal) Summary

- Minimal CSP definition: everything is part of a solution.
- It looks very easy, but it is actually hard.
Thank you very much for your attention!