Outline

1. Constraint Programming
2. Robustness
3. Super Solutions
4. Applications
Outline

1. Constraint Programming
2. Robustness
3. Super Solutions
4. Applications
Constraint Satisfaction Problem

- **Variables**: finite discrete domain ($\subseteq \mathbb{Z}$)
- **Constraints**: any polynomial-time checkable relation
  - Logical or arithmetic operators $\{\neq, >, \leq, \lor, \Rightarrow, \ldots\}$
  - Linear or non-linear equations
  - Standard subproblems
    - Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - NP-hard: Hitting set, Bin packing, Linear equality, ...
- **Inference mecanism**: **Propagation!**
  - Instead of finding a solution to these constraints, we look for inconsistent values, and remove them
Constraint Satisfaction Problem

- Variables: finite discrete domain \(\subseteq \mathbb{Z}\)
- Constraints: any polynomial-time checkable relation
  - Logical or arithmetic operators \(\{\neq, >, \leq, \text{or, } \Rightarrow, \ldots\}\)
  - Linear or non-linear equations
  - Standard subproblems
    - Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: Propagation!
  - Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

\[
\begin{align*}
x & \overset{<}{\rightarrow} y & \neq & z \\
\{1, 2, 3\} & \{1, 2\} & \{1, 2, 3\}
\end{align*}
\]
Constraint Satisfaction Problem

- Variables: finite discrete domain (\(\subseteq \mathbb{Z}\))
- Constraints: any polynomial-time checkable relation
  - Logical or arithmetic operators \{\neq, >, \leq, or, \Rightarrow, \ldots\}
  - Linear or non-linear equations
  - Standard subproblems
    - Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation!**
  - Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

\[
\begin{array}{c}
  x < y \\
  \neq \\
  z
\end{array}
\]

\[
\{1\} \quad \{2\} \quad \{1, 2, 3\}
\]
Constraint Satisfaction Problem

- Variables: finite discrete domain ($\subseteq \mathbb{Z}$)
- Constraints: any polynomial-time checkable relation
  - Logical or arithmetic operators $\{\neq, >, \leq, \text{or}, \Rightarrow, \ldots\}$
  - Linear or non-linear equations
  - Standard subproblems
    - Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation**!
  - Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

\[
x < y \neq z
\]
\[
\{1\} \quad \{2\} \quad \{1, 3\}
\]
Outline

1 Constraint Programming

2 Robustness
   - Context Free Robustness?
   - Stability
   - Super solutions

3 Super Solutions

4 Applications
### Solution Robustness

- Satisfaction, Optimisation: find a solution
- Uncertainty
  - Unexpected change in the data

## Change

A change can be seen as an additional constraint

## Solution Robustness

A solution $\sigma$ is more robust than $\sigma'$ iff the probability that a change invalidates $\sigma$ is lower than the probability that it invalidates $\sigma'$
Solution Robustness

A solution $\sigma$ is more robust than $\sigma'$ iff the probability that a change invalidates $\sigma$ is lower than the probability that it invalidates $\sigma'$.

Is it possible to characterise solution robustness without assumptions on the changes?

▶ Not with the definition above ⋆

⋆ There are exactly as many changes that invalidate $\sigma$ and not $\sigma'$ as the opposite

⋆ Requires a probability distribution, or any kind of information on the possible changes
Solution Robustness

A solution $\sigma$ is more robust than $\sigma'$ iff the probability that a change invalidates $\sigma$ is lower than the probability that it invalidates $\sigma'$.

Is it possible to characterise solution robustness without assumptions on the changes?
Context Free Robustness?

Solution Robustness
A solution \( \sigma \) is more robust than \( \sigma' \) iff the probability that a change invalidates \( \sigma \) is lower than the probability that it invalidates \( \sigma' \).

- *Is it possible to characterise solution robustness without assumptions on the changes?*
  - Not with the definition above
    - There are exactly as many changes that invalidate \( \sigma \) and not \( \sigma' \) as the opposite
    - Requires a probability distribution, or any kind of information on the possible changes
Stability

Consider the following Boolean satisfaction problem:

\[ \sum_{i=1}^{n} x_i \geq k \]

111...11 or 111100...00?

the solution assigning every variable to 1 seems more “robust” than assigning only \( \{x_1, \ldots, x_k\} \) to 1:

- If the change involves less than \( n - k \) variables, then simply re-assigning these variables in any consistent way yields a new solution
- With the second solution, we may need to re-assign variables that were not involved in the change
Stability

- Consider the following Boolean satisfaction problem:

\[ \sum_{i=1}^{n} x_i \geq k \]

- 111...11 or 111100...00?
- the solution assigning every variable to 1 seems more “robust” than assigning only \( \{x_1, \ldots, x_k\} \) to 1:
  - If the change involves less than \( n - k \) variables, then simply re-assigning these variables in any consistent way yields a new solution.
  - With the second solution, we may need to re-assign variables that were not involved in the change.

Stability

A solution \( \sigma \) is more stable than \( \sigma' \) iff a change requires less repairs than on \( \sigma' \) to obtain a solution consistent with the change.
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a **break** and denote its size by $a$.
- We call any solution of $\Phi \& c$ a **repair**, and denote the size of its difference with respect to $\sigma$ by $b$.

\[
\sum_{i=1}^{n} x_i \geq 6
\]

\[
\sigma \begin{array}{ccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a break and denote its size by $a$.
- We call any solution of $\Phi \& c$ a repair, and denote the size of its difference with respect to $\sigma$ by $b$.

\[ \sum_{i=1}^{n} x_i \geq 6 \]

$\sigma$ $1$ $1$ $1$ $1$ $1$ $1$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$ $0$

$\beta$ $0$ $0$ $0$ $1$

break
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a **break** and denote its size by $a$.
- We call any solution of $\Phi$ & $c$ a **repair**, and denote the size of its difference with respect to $\sigma$ by $b$.

\[
\sum_{i=1}^{n} x_i \geq 6
\]

\[
\sigma \begin{array}{ccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

\[
\beta \begin{array}{ccccccccccc}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

\[
\text{break}
\]
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a break and denote its size by $a$.
- We call any solution of $\Phi \& c$ a repair, and denote the size of its difference with respect to $\sigma$ by $b$.

$$\sum_{i=1}^{n} x_i \geq 6$$

$\sigma$ 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$\beta$ 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0

break repair
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a break and denote its size by $a$.
- We call any solution of $\Phi \& c$ a repair, and denote the size of its difference with respect to $\sigma$ by $b$.

\[
\sum_{i=1}^{n} x_i \geq 6
\]

$\sigma$ 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

$\beta$ 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0

$\beta$ (worst case)

Break

Repair
Let $\Phi$ be a problem, $\sigma$ be a solution and $c$ be a change.

We identify $c$ with the set of variables that need to be changed in $\sigma$ to satisfy $c$.

- We call this set of variables a break and denote its size by $a$.
- We call any solution of $\Phi \& c$ a repair, and denote the size of its difference with respect to $\sigma$ by $b$.

$$\sum_{i=1}^{n} x_i \geq 6$$

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| $\sigma$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\beta$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

The red region indicates the break.
Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution $\sigma$ is a $(a, b)$-super solution iff for all breaks of size $a$ or less, there exists a repair of size $b$ or less.
Super Solutions

Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution $\sigma$ is a $(a, b)$-super solution iff for all breaks of size $a$ or less, there exists a repair of size $b$ or less.

- $1^n$ is a $(a, 0)$-super solution of $\sum_{i=1}^{n} x_i \geq k$ for all $a \leq n - k$
Super Solutions

Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution $\sigma$ is a $(a, b)$-super solution iff for all breaks of size $a$ or less, there exists a repair of size $b$ or less.

- $1^n$ is a $(a, 0)$-super solution of $\sum_{i=1}^{n} x_i \geq k$ for all $a \leq n - k$
- Four Ph.D. Thesis on the topic!
  - Emmanuel Hebrard “Robust Solutions for Constraint Satisfaction and Optimisation under Uncertainty” (2006)
  - Alan Holland “Risk Management for Combinatorial Auctions” (2005)
Outline

1. Constraint Programming

2. Robustness

3. Super Solutions
   - Complexity
   - Getting Super solutions

4. Applications
Super-CSP is intractable

**CSP**
Given a CSP $\Phi$, does $\Phi$ admit a solution?

**Super-CSP**
Given a CSP $\Phi$ and two ints $a, b$, does $\Phi$ admit a $(a, b)$-super solution?

- NP-hard trivial reduction from CSP
- in $\text{NEXP}$: exponential number of witnesses, each NP-hard to check
Super-CSP is intractable

CSP
Given a CSP $\Phi$, does $\Phi$ admit a solution?

Super-CSP
Given a CSP $\Phi$ and two ints $a$, $b$, does $\Phi$ admit a $(a, b)$-super solution?

- NP-hard trivial reduction from CSP
- in NEXP: exponential number of witnesses, each NP-hard to check

Super solution checking
Given a CSP $\Phi$, a sol $\sigma$ and an int $b$, is $\sigma$ a $(1, b)$-super solution of $\Phi$?
Super-CSP is intractable

**CSP**
Given a CSP $\Phi$, does $\Phi$ admit a solution?

**Super-CSP**
Given a CSP $\Phi$ and two ints $a, b$, does $\Phi$ admit a $(a, b)$-super solution?

- NP-hard trivial reduction from CSP
- in NEXP: exponential number of witnesses, each NP-hard to check

**Super solution checking**
Given a CSP $\Phi$, a sol $\sigma$ and an int $b$, is $\sigma$ a $(1, b)$-super solution of $\Phi$?
- NP-complete, reduction from K-Clique
(a,b)-Super-CSP is NP-hard

(a,b)-Super-CSP

Given a CSP \( \Phi \), does \( \Phi \) admit a \((a, b)\)-super solution?

- Membership to NP (witnessed by itself AND the polynomial set of repairs)
- If (1,b)-Super-CSP is NP-hard then (1,b+1)-Super-CSP is NP-hard
  - Hence NP-complete for all \( a, b \)
Finding Super Solution of Tractable-CSP

- **SAT tractable classes** [Ginsberg, Parkes and Roy 2006]
  - (1,b)-Horn-SAT is NP-complete for all $b$
  - (1,b)-2-SAT is polynomial for $b \leq 1$ and NP-complete otherwise
  - (a,b)-Affine-SAT is polynomial for all $a, b$

- **CSP tractable classes** [Hebrard, Hnich and Walsh 2006]
  - (a,b)-Class-0-CSP is NP-hard for all $a, b$
  - (1,b)-Tree-CSP is polynomial for $b = 0$, NP-complete for $b > 1$ and open for $b = 1$
  - (1,b)-Majority-CSP is NP-complete for $b > 1$, and open otherwise
Finding Super Solutions

- Simplest case: \((1, 0)\)-super solutions
  - For each variable, there is an alternative value for this variable
Finding Super Solutions

- Simplest case: \((1, 0)\)-super solutions
  - For each variable, there is an alternative value for this variable

\[ x, y, z \in \{1, 2, 3\} \]
\[ x \neq y, \ y \leq z \]
Finding Super Solutions

- Simplest case: \((1, 0)\)-super solutions
  - For each variable, there is an alternative value for this variable

\[
x, y, z \in \{1, 2, 3\} \\
x \neq y, \ y \leq z
\]

\[\langle x = 3, y = 1, z = 3 \rangle\] is a \((1, 0)\)-super solution, since:

- \[\langle x = 2, y = 1, z = 3 \rangle\] is a solution
- \[\langle x = 3, y = 2, z = 3 \rangle\] is a solution
- \[\langle x = 3, y = 1, z = 2 \rangle\] is a solution
Reformulation $(\mathcal{P} + \mathcal{P})$

Given a solution $\sigma$, its restriction to $x_1, x_2, x_3, x_4$ is a $(1, 0)$-super solution.

Its restriction to $y_1, x_2, x_3, x_4$ is a repair for the break $\{x_1\}$.

Its restriction to $x_1, y_2, x_3, x_4$ is a repair for the break $\{x_2\}$.

Its restriction to $x_1, x_2, y_3, x_4$ is a repair for the break $\{x_3\}$.

Its restriction to $x_1, x_2, x_3, y_4$ is a repair for the break $\{x_4\}$.
Reformulation \((\mathcal{P} + \mathcal{P})\)

Given a solution \(\sigma\), its restriction to \(x_1, x_2, x_3, x_4\) is a \((1, 0)\)-super solution,

\[
\begin{align*}
\{x_1\} &\quad \text{is a repair for the break } \{x_1\} \\
\{x_2\} &\quad \text{is a repair for the break } \{x_3\} \\
\{x_3\} &\quad \text{is a repair for the break } \{x_4\}
\end{align*}
\]
Reformulation \((\mathcal{P} + \mathcal{P})\)

\[
\begin{align*}
x_1 &\rightarrow x_2 &\rightarrow x_3 &\rightarrow x_4 \\
y_1 &\not\equiv &\quad &\quad \\
\end{align*}
\]
Reformulation \((\mathcal{P} + \mathcal{P})\)

Given a solution \(\sigma\), its restriction to \(x_1, x_2, x_3, x_4\) is a (1, 0)-super solution;
its restriction to \(y_1, x_2, x_3, x_4\) is a repair for the break \(\{x_1\}\);
its restriction to \(x_1, y_2, x_3, x_4\) is a repair for the break \(\{x_2\}\);
its restriction to \(x_1, x_2, y_3, x_4\) is a repair for the break \(\{x_3\}\);
its restriction to \(x_1, x_2, x_3, y_4\) is a repair for the break \(\{x_4\}\).
Given a solution $\sigma$, its restriction to $x_1, x_2, x_3, x_4$ is a $(1, 0)$-super solution
- its restriction to $y_1, x_2, x_3, x_4$ is a repair for the break $\{x_1\}$
- its restriction to $x_1, y_2, x_3, x_4$ is a repair for the break $\{x_2\}$
- its restriction to $x_1, x_2, y_3, x_4$ is a repair for the break $\{x_3\}$
- its restriction to $x_1, x_2, x_3, y_4$ is a repair for the break $\{x_4\}$
Reformulation \((\mathcal{P} \times \mathcal{P}) / \text{Super Arc-Consistency}\)

- **Arc-Consistency**: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency

![Diagram](image-url)
Reformulation $(\mathcal{P} \times \mathcal{P}) / \text{Super Arc-Consistency}$

- **Arc-Consistency**: Each value has a support
  - Local inconsistency $\Rightarrow$ global inconsistency

![Diagram](image-url)
Reformulation \((\mathcal{P} \times \mathcal{P}) / \) Super Arc-Consistency

- **Arc-Consistency**: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency
- "Super" Arc-Consistency: Each value has a support
  - And an alternative support
Reformulation \((\mathcal{P} \times \mathcal{P})\) / Super Arc-Consistency

- Arc-Consistency: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - And an alternative support
Reformulation \((\mathcal{P} \times \mathcal{P})\) / Super Arc-Consistency

- **Arc-Consistency**: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency

- “Super” Arc-Consistency: Each value has a support
  - And an alternative support
Reformulation \((\mathcal{P} \times \mathcal{P}) / \text{Super Arc-Consistency}\)

- **Arc-Consistency**: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency
- **“Super” Arc-Consistency**: Each value has a support
  - And an alternative support
Reformulation \((\mathcal{P} \times \mathcal{P}) / \text{Super Arc-Consistency}\)

- **Arc-Consistency**: Each value has a support
  - Local inconsistency \(\Rightarrow\) global inconsistency
- **“Super” Arc-Consistency**: Each value has a support
  - And an alternative support
- Reformulation with stronger propagation
- Reformulation of the algorithm for Arc-Consistency
Combinatorial Auction [Holland & O’Sullivan]
Combinatorial Auction [Holland & O’Sullivan]

- 3 pictures to sell
  A: 
  B: 
  C: 

Winner Selection Problem

▶ A \{0,1\} variable for each bid

▶ Bernard A, B takes the value 1 iff Bernard's bid on items A and B wins

maximize: the sum of the bids

subject to: no item is attributed more than once
Combinatorial Auction [Holland & O’Sullivan]

3 pictures to sell  A: ![Picture A]  B: ![Picture B]  C: ![Picture C]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>
Combinatorial Auction [Holland & O’Sullivan]

- 3 pictures to sell

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>

Winner Selection Problem

- A \{0, 1\} variable for each bid
- Bernard_{A,B} takes the value 1 iff Bernard’s bid on items A and B wins

- maximize: the sum of the bids
- subject to: no item is attributed more than once
Combinatorial Auction

- 3 pictures to sell

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>
Combinatorial Auction

- 3 pictures to sell: A:  B:  C:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>

(1) Bernard gets A and Mouhamad gets B&C (500000€)
Combinatorial Auction

- 3 pictures to sell
  - A: [Image]
  - B: [Image]
  - C: [Image]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Bernard gets A and Mouhamad gets B&C (500000€)
  - Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or A&B to Ziad and C to Bernard (400000€)
3 pictures to sell

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>500000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
</tbody>
</table>

(1) Bernard gets A and Mouhamad gets B&C (500000€)
  ▶ Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or A&B to Ziad and C to Bernard (400000€)
    ★ Either lose money or take back picture A from its winner (Bernard)
Combinatorial Auction

3 pictures to sell

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>500000</td>
<td>500000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>

(1) Bernard gets A and Mouhamad gets B & C (500000€)
   - Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or A & B to Ziad and C to Bernard (400000€)
     - Either lose money or take back picture A from its winner (Bernard)

(2) Bernard gets C and Mouhamad gets A & B
Combinatorial Auction

3 pictures to sell: A: B: C:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernard</td>
<td>200000</td>
<td>0</td>
<td>200000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
<td>250000</td>
</tr>
<tr>
<td>Mouhamad</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>300000</td>
<td>200000</td>
<td>300000</td>
<td>400000</td>
</tr>
<tr>
<td>Ziad</td>
<td>50000</td>
<td>50000</td>
<td>100000</td>
<td>200000</td>
<td>150000</td>
<td>150000</td>
<td>400000</td>
</tr>
</tbody>
</table>

(1) Bernard gets A and Mouhamad gets B&C (500000€)
   ▶ Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or A&B to Ziad and C to Bernard (400000€)
      ★ Either lose money or take back picture A from its winner (Bernard)

(2) Bernard gets C and Mouhamad gets A&B
   ▶ Mouhamad withdraw his bid: Ziad replace him (400000€)
   ▶ Bernard withdraw his bid: Ziad replace him (400000€)
Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - Robustness, stochastic reasoning (probabilistic super solutions)
Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - Robustness, stochastic reasoning (probabilistic super solutions)
- Tradeoff between stability and (expected) utility
Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - Robustness, stochastic reasoning (probabilistic super solutions)
- Tradeoff between stability and (expected) utility
- In Combinatorial auctions, impact of the stability on the truthfulness of the agents
  - A bid is truthful if it is a correct assessment of the utility of the item to the agent
  - Auction systems should incentivize truthful bidding
    - In Vickrey auctions, truthfull bidding is a dominating strategy
Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - Robustness, stochastic reasoning (probabilistic super solutions)
- Tradeoff between stability and (expected) utility
- In Combinatorial auctions, impact of the stability on the truthfulness of the agents
  - A bid is truthful if it is a correct assessment of the utility of the item to the agent
  - Auction systems should incentivize truthful bidding
    - In Vickrey auctions, truthfull bidding is a dominating strategy
    - Not anymore with stable Vickrey auctions! (though it is possible to make it so)
Questions?

Coffee break, yet?
**Arc-Consistency**

**Support**

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent.
  - Propagation until reaching a fix point

```
    x       y       z
  1 <--- 1 <--- 1
  2     2     2
  3  --- 3  --- 3
```
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point

\[
\begin{array}{ccc}
  x & \quad & y \\
  1 & \xrightarrow{1} & 2 \\
  2 & \xrightarrow{2} & 3 \\
\end{array}
\quad
\begin{array}{ccc}
  y & \quad & z \\
  1 & \xrightarrow{1} & 2 \\
  2 & \xrightarrow{2} & 3 \\
\end{array}
\]
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point

![Graph showing arc-consistency](image_url)
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point
**Arc-Consistency**

**Support**

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent
  - Propagation until reaching a fix point

![Diagram of arc-consistency](image-url)
Arc-Consistency

Support

- A support $\sigma$ of a value $v$ for a constraint $c$ is a solution of this constraint such that every value is itself arc-consistent.
  - Propagation until reaching a fix point

![Graph showing arc-consistency](image)
Reformulation \((P \times P)\)
Reformulation ($\mathcal{P} \times \mathcal{P}$)
Reformulation \((\mathcal{P} \times \mathcal{P})\)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x'</th>
<th>y'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>\langle 1, 2 \rangle</td>
<td>\langle 1, 2 \rangle</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>\langle 1, 3 \rangle</td>
<td>\langle 1, 3 \rangle</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>\langle 2, 1 \rangle</td>
<td>\langle 2, 1 \rangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\langle 2, 3 \rangle</td>
<td>\langle 2, 3 \rangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\langle 3, 1 \rangle</td>
<td>\langle 3, 1 \rangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\langle 3, 2 \rangle</td>
<td>\langle 3, 2 \rangle</td>
</tr>
</tbody>
</table>
Reformulation \((\mathcal{P} \times \mathcal{P})\)

\[
\begin{array}{cc}
x & y \\
1 & 1 \\
2 & 2 \\
3 & 3 \\
\end{array}
\quad\quad
\begin{array}{cc}
x' & y' \\
\langle 1, 2 \rangle & \langle 1, 2 \rangle \\
\langle 1, 3 \rangle & \langle 1, 3 \rangle \\
\langle 2, 1 \rangle & \langle 2, 1 \rangle \\
\langle 2, 3 \rangle & \langle 2, 3 \rangle \\
\langle 3, 1 \rangle & \langle 3, 1 \rangle \\
\langle 3, 2 \rangle & \langle 3, 2 \rangle \\
\end{array}
\]
Reformulation \((\mathcal{P} \times \mathcal{P})\)

\[
\begin{array}{cc}
\text{x} & \text{y} \\
1 & 1 \\
2 & 2 \\
3 & 3 \\
\end{array}
\]

\[
\begin{array}{cc}
\langle 1, 2 \rangle & \langle 1, 2 \rangle \\
\langle 1, 3 \rangle & \langle 1, 3 \rangle \\
\langle 2, 1 \rangle & \langle 2, 1 \rangle \\
\langle 2, 3 \rangle & \langle 2, 3 \rangle \\
\langle 3, 1 \rangle & \langle 3, 1 \rangle \\
\langle 3, 2 \rangle & \langle 3, 2 \rangle \\
\end{array}
\]
Reformulation \((\mathcal{P} \times \mathcal{P})\)

\[
\begin{array}{ccc}
  & x & y \\
1 & \leftrightarrow & 1 \\
2 & \leftrightarrow & 2 \\
3 & \leftrightarrow & 3 \\
\end{array}
\]

\[
\begin{array}{ccc}
  & x' & y' \\
\langle 1, 2 \rangle & \leftrightarrow & \langle 1, 2 \rangle \\
\langle 1, 3 \rangle & \leftrightarrow & \langle 1, 3 \rangle \\
\langle 2, 1 \rangle & \leftrightarrow & \langle 2, 1 \rangle \\
\langle 2, 3 \rangle & \leftrightarrow & \langle 2, 3 \rangle \\
\langle 3, 1 \rangle & \leftrightarrow & \langle 3, 1 \rangle \\
\langle 3, 2 \rangle & \leftrightarrow & \langle 3, 2 \rangle \\
\end{array}
\]
This model allows stronger propagation!
This model allows stronger propagation!

However, the domain size is quadratic
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values.
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values.
Super-Arc-Consistency

“super”-values ⊆ “repair”-values ⊆ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values
Super-Arc-Consistency

"super"-values $\subseteq$ "repair"-values $\subseteq$ domain

- Each "repair"-value must have a support in the set of "super"-values
- Each "super"-value must have a support in the set of "super"-values, and another in the set of "repair"-values

![Diagram](https://via.placeholder.com/150)
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values

```
x            y            z
   1 ← 1 ← 1
   2 ← 2 ← 2
   3 ← 3 ← 3
```

```
x            y            z
   1 ← 1 ← 1
   2 ← 2 ← 2
   3 ← 3 ← 3
```
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values.
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values.
Super-Arc-Consistency

“super”-values \( \subseteq \) “repair”-values \( \subseteq \) domain

- Each “repair”-value must have a support in the set of “super”-values.
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values.
Super-Arc-Consistency

“super”-values ⊆ “repair”-values ⊆ domain

- Each “repair”-value must have a support in the set of “super”-values.
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values.

![Graph showing the relationships between variables x, y, and z, with values 1 and 2 connected by arcs.](image)
Super-Arc-Consistency

“super”-values $\subseteq$ “repair”-values $\subseteq$ domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values

The only $(1, 0)$-super solution is $\langle 1, 1, 2 \rangle$
Constraint Satisfaction Problem

- Variables: finite discrete domain \( \subseteq \mathbb{Z} \)
- Constraints: any polynomial-time checkable relation
  - Any fixed arity relation
    - Logical or arithmetic operators \( \{\neq, >, \leq, \text{or}, \Rightarrow, \ldots\} \)
  - Linear or non-linear equations
  - Standard subproblems
    - Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - NP-hard: Hitting set, Bin packing, Linear equality, ...
CSP and Propagation

- Basic idea:
  - any NP-hard problem can be formulated as a conjunction of subproblems (constraints)
  - each constraint is easy (either polynomial or well understood)
- How to use that to solve the composite problem?
  - Propagation!
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- $\text{MATCHING}([x_1, \ldots, x_7], \{1, \ldots, 9\})$

\[
\begin{align*}
x_1 : & \quad \{8, 9\} \\
x_2 : & \quad \{1, 2, 6, 7, 8, 9\} \\
x_3 : & \quad \{8, 9\} \\
x_4 : & \quad \{1, 5, 6, 8, 9\} \\
x_5 : & \quad \{1, 2, 6, 7, 8, 9\} \\
x_6 : & \quad \{4, 5, 8, 9\}
\end{align*}
\]
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- $\text{MATCHING}([x_1, \ldots, x_7], \{1, \ldots, 9\})$

$x_1 : \{8, 9\}$
$x_2 : \{1, 2, 6, 7, 8, 9\}$
$x_3 : \{8, 9\}$
$x_4 : \{1, 5, 6, 8, 9\}$
$x_5 : \{1, 2, 6, 7, 8, 9\}$
$x_6 : \{4, 5, 8, 9\}$

Propagation

- $\text{MATCHING}([x_1, x_3], \{8, 9\})$
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- $\text{MATCHING}([x_1, \ldots, x_7], \{1, \ldots, 9\})$

$x_1: \{ \begin{array}{c} 8 \quad 9 \end{array} \}$
$x_2: \{1 \quad 2 \quad 6 \quad 7 \} $
$x_3: \{ \begin{array}{c} 8 \quad 9 \end{array} \}$
$x_4: \{1 \quad 5 \quad 6 \} $
$x_5: \{1 \quad 2 \quad 6 \quad 7 \} $
$x_6: \{ \begin{array}{c} 4 \quad 5 \end{array} \}$

Propagation

- $\text{MATCHING}([x_1, x_3], \{8, 9\})$
Example: Kakuro

\[ \sum_{i=1}^{7} x_i = 39 \]

**MATCHING**\((\{x_1, \ldots, x_7\}, \{1, \ldots, 9\})\)

\[
\begin{align*}
x_1 & : \{8 \ 9\} \\
x_2 & : \{1 \ 2 \ 6 \ 7\} \\
x_3 & : \{8 \ 9\} \\
x_4 & : \{1 \ 5 \ 6\} \\
x_5 & : \{1 \ 2 \ 6 \ 7\} \\
x_6 & : \{4 \ 5\}
\end{align*}
\]

**Propagation**

\[ \sum_{i=1}^{6} x_i = 39 \]

\[ \Rightarrow \min(x_2) \geq 39 - \sum_{i \neq 2} \max(x_i) \]

\[ \Rightarrow \min(x_2) \geq 3, (\& \ \min(x_5) \geq 3 \ \& \ \min(x_4) \geq 2) \]
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- MATCHING(\{x_1, \ldots, x_7\}, \{1, \ldots, 9\})

\begin{align*}
x_1 & : \quad \{8, 9\} \\
x_2 & : \quad \{6, 7\} \\
x_3 & : \quad \{8, 9\} \\
x_4 & : \quad \{5, 6\} \\
x_5 & : \quad \{6, 7\} \\
x_6 & : \quad \{4, 5\}
\end{align*}

Propagation

- $\sum_{i=1}^{6} x_i = 39$
  - $\Rightarrow \min(x_2) \geq 39 - \sum_{i\neq2} \max(x_i)$
  - $\Rightarrow \min(x_2) \geq 3, (\& \min(x_5) \geq 3 \& \min(x_4) \geq 2)$
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- $\text{MATCHING}([x_1, \ldots, x_7], \{1, \ldots, 9\})$

- $x_1 : \{8 \ 9\}$
- $x_2 : \{6 \ 7\}$
- $x_3 : \{8 \ 9\}$
- $x_4 : \{5 \ 6\}$
- $x_5 : \{6 \ 7\}$
- $x_6 : \{4 \ 5\}$

Propagation

- $\text{MATCHING}([x_2, x_5], \{6, 7\})$
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- \text{MATCHING}($\{x_1, \ldots, x_7\}, \{1, \ldots, 9\}$)

\[
\begin{align*}
    x_1 & : \quad \{ 8, 9 \} \\
    x_2 & : \quad \{ 6, 7 \} \\
    x_3 & : \quad \{ 8, 9 \} \\
    x_4 & : \quad \{ 5 \} \\
    x_5 & : \quad \{ 6, 7 \} \\
    x_6 & : \quad \{ 4, 5 \}
\end{align*}
\]

Propagation

- \text{MATCHING}($\{x_2, x_5\}, \{6, 7\}$)
Example: Kakuro

- $\sum_{i=1}^{7} x_i = 39$
- $\text{MATCHING}([x_1, \ldots, x_7], \{1, \ldots, 9\})$

$x_1 : \{\begin{array}{c} 8 \\ 9 \end{array}\}$
$x_2 : \{\begin{array}{c} 6 \\ 7 \end{array}\}$
$x_3 : \{\begin{array}{c} 8 \\ 9 \end{array}\}$
$x_4 : \{\begin{array}{c} 5 \end{array}\}$
$x_5 : \{\begin{array}{c} 6 \\ 7 \end{array}\}$
$x_6 : \{\begin{array}{c} 4 \end{array}\}$

Propagation

- $\text{MATCHING}([x_4], \{5\})$