Constraint Satisfaction Problems

George Osipov

Linköping University

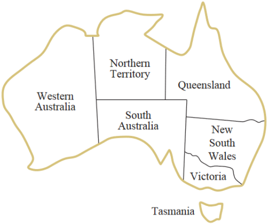
June 19, 2021

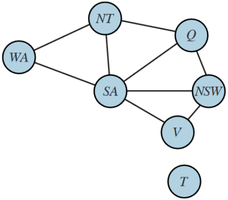
Example 1: Linear Equations

Does this system of linear equations modulo 2 have a solution?

$$
\begin{aligned}
x_1 + x_2 + x_5 &= 1 \pmod 2 \\
x_1 + x_3 + x_4 &= 1 \pmod 2 \\
x_2 + x_3 + x_5 &= 0 \pmod 2 \\
x_2 + x_3 + x_4 &= 1 \pmod 2
\end{aligned}
$$

Example 2: Coloring





Can we color all states of Australia with red, green, and blue so that no two neighboring states are assigned the same color?

Example 3: Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Can we fill in the blanks with digits $1, 2, \ldots, 9$ so that no digit appears twice in any row, column or 3×3 box?

Common Framework – CSP

Constraint Satisfaction Problem (CSP)

INSTANCE: (V, D, C), where

- V is the set of variables,
- D is the set of values,
- C is the set of constraints.

QUESTION: Is there an assignment of values to the variables that satisfies all constraints?

Example 1: Linear Equations as CSP

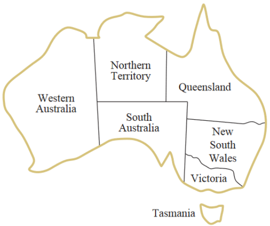
Does this system of linear equations modulo 2 have a solution?

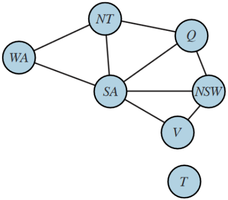
$$
\begin{aligned}
x_1 + x_2 + x_5 &= 1 \mod 2 \\
x_1 + x_3 + x_4 &= 1 \mod 2 \\
x_2 + x_3 + x_5 &= 0 \mod 2 \\
x_2 + x_3 + x_4 &= 1 \mod 2
\end{aligned}
$$

Variables: $\{x_1, x_2, x_3, x_4, x_5\}$.
Values: $\{0, 1\}$.
Constraints are $x + y + z = 0 \text{mod} 2$, $x + y + z = 1 \text{mod} 2$.

Example 2: Coloring as CSP

Can we color all states of Australia with red, green, and blue so that no two neighboring states are assigned the same color?





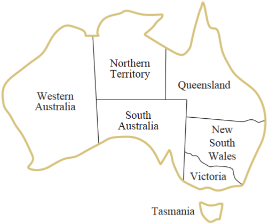
Variables: $\{WA, NT, SA, NSW, V, SA, T\}$.

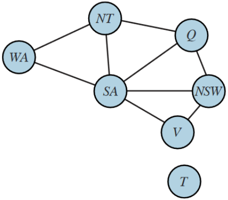
Values: {red, green, blue}.

Constraints: $WA \neq NT$, $WA \neq SA$, $NT \neq SA$, $\ldots$

Example 2: Coloring as CSP

Can we color all states of Australia with red, green, and blue so that no two neighboring states are assigned the same color?



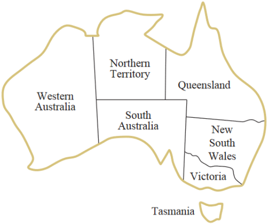


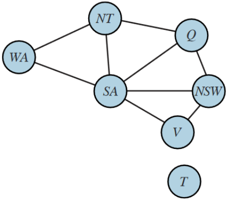
Variables: $\{WA, NT, SA, NSW, V, SA, T\}$.

Values: {red, green, blue}.

Constraints: $WA \neq NT$, $WA \neq SA$, $NT \neq SA$, $\ldots$

Can we color all states of Australia with red, green, and blue so that no two neighboring states are assigned the same color?





Variables: $\{WA, NT, SA, NSW, V, SA, T\}$.

Values: {red, green, blue}.

Constraints: $WA \neq NT$, $WA \neq SA$, $NT \neq SA$, $\ldots$

Example 3: Sudoku as CSP

Can we fill in the blanks with digits $1, 2, \ldots, 9$ so that no digit appears twice in any row, column or 3×3 box?

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Variables: `A1`, `A2`, ..., `I9` (81 in total).

Values: $1, 2, 3, 4, 5, 6, 7, 8, 9$.

Constraints are of two types: (i) $\texttt{B1} = 9, \ldots, \texttt{H9} = 9$ and (ii) $\texttt{AllDiff(A1}, \ldots, \texttt{A9)}$, $\texttt{AllDiff(A1}, \ldots, \texttt{I1)}$, $\texttt{AllDiff(A1}, \ldots, \texttt{C3)}$, ...

Example 3: Sudoku as CSP

Can we fill in the blanks with digits $1, 2, \ldots, 9$ so that no digit appears twice in any row, column or 3×3 box?

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Variables: `A1`, `A2`, ..., `I9` (81 in total).

Values: $1, 2, 3, 4, 5, 6, 7, 8, 9$.

Constraints are of two types: (i) $\texttt{B1} = 9, \ldots, \texttt{H9} = 9$ and (ii) $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{A9})$, $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{I1})$, $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{C3})$, ...

Example 3: Sudoku as CSP

Can we fill in the blanks with digits $1, 2, \ldots, 9$ so that no digit appears twice in any row, column or 3×3 box?

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Variables: `A1`, `A2`, ..., `I9` (81 in total).

Values: $1, 2, 3, 4, 5, 6, 7, 8, 9$.

Constraints are of two types: (i) `B1` $= 9, \ldots,$ `H9` $= 9$ and (ii) `AllDiff(A1,...,A9)`, `AllDiff(A1,...,I1)`, `AllDiff(A1,...,C3)`, ...

Example 3: Sudoku as CSP

Can we fill in the blanks with digits $1, 2, \ldots, 9$ so that no digit appears twice in any row, column or 3×3 box?

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

Variables: `A1`, `A2`, ..., `I9` (81 in total).

Values: $1, 2, 3, 4, 5, 6, 7, 8, 9$.

Constraints are of two types: (i) $\texttt{B1} = 9, \ldots, \texttt{H9} = 9$ and (ii) $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{A9})$, $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{I1})$, $\texttt{AllDiff}(\texttt{A1}, \ldots, \texttt{C3})$, ...

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$:

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-Coloring:

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- LINEAR EQUATIONS OVER $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-COLORING: yes! Use constraint propagation (next slide).

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-Coloring: yes! Use constraint propagation (next slide).
- 3-Coloring:

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- LINEAR EQUATIONS OVER $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-COLORING: yes! Use constraint propagation (next slide).
- 3-COLORING: probably not.

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-Coloring: yes! Use constraint propagation (next slide).
- 3-Coloring: probably not.
- CSP in general:

How to solve CSPs

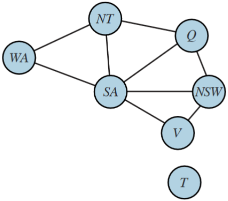
- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-Coloring: yes! Use constraint propagation (next slide).
- 3-Coloring: probably not.
- CSP in general: probably not.

How to solve CSPs

- Brute force: try all $|D|^{|V|}$ assignments (applicable only if $|D|$ is finite).
- Can we do better than that?
- Linear equations over $\mathbb{F}_2$: yes! Gaussian elimination.
- 2-Coloring: yes! Use constraint propagation (next slide).
- 3-Coloring: probably not.
- CSP in general: probably not.
- probably stands for "unless P=NP" (more on that later).

Local Consistency: Example

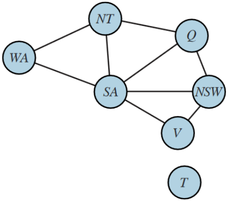
Solving 2-Coloring with $(2, 3)$-consistency algorithm.



1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

Local Consistency: Example

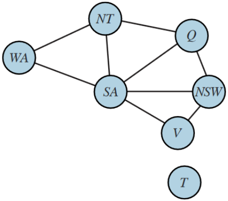
Solving 2-Coloring with $(2,3)$-consistency algorithm.



1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

Local Consistency: Example

Solving 2-Coloring with $(2,3)$-consistency algorithm.

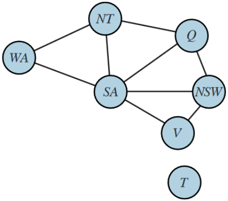


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	V
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Local Consistency: Example

Solving 2-Coloring with $(2,3)$-consistency algorithm.

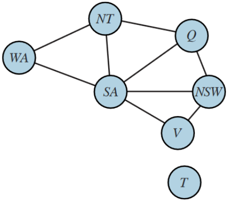


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	V
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Local Consistency: Example

Solving 2-COLORING with $(2,3)$-consistency algorithm.

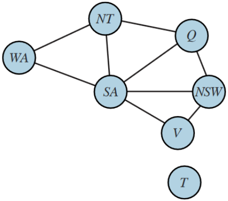


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	V
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Local Consistency: Example

Solving 2-Coloring with $(2,3)$-consistency algorithm.

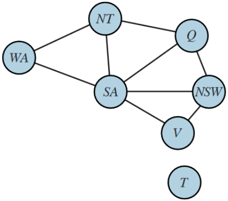


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	NT
0	1	0
0	1	1
1	0	0
1	0	1

Local Consistency: Example

Solving 2-Coloring with $(2, 3)$-consistency algorithm.

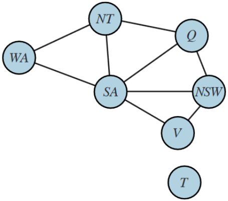


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	NT
0	1	0
0	1	1
1	0	0
1	0	1

Local Consistency: Example

Solving 2-Coloring with $(2, 3)$-consistency algorithm.

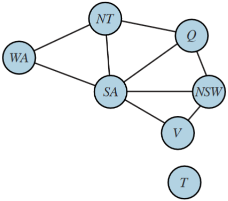


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.

WA	SA	NT
0	1	0
0	1	1
1	0	0
1	0	1

Local Consistency: Example

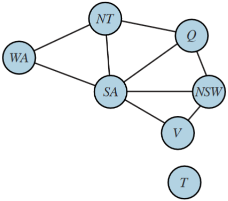
Solving 2-Coloring with $(2,3)$-consistency algorithm.



1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.
4. If no record changed, stop.
5. Else, go to step 2.

Local Consistency: Example

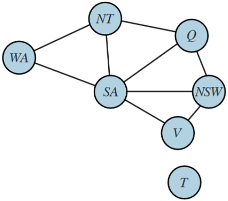
Solving 2-Coloring with $(2, 3)$-consistency algorithm.



1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.
4. If no record changed, stop.
5. Else, go to step 2.

Local Consistency: Example

Solving 2-Coloring with $(2, 3)$-consistency algorithm.

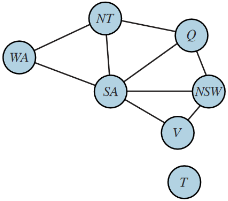


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.
4. If no record changed, stop.
5. Else, go to step 2.

- If some record is empty, inconsistency has been detected.

Local Consistency: Example

Solving 2-Coloring with $(2,3)$-consistency algorithm.

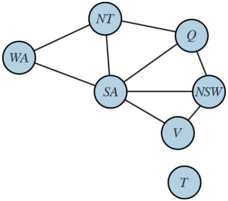


1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.
4. If no record changed, stop.
5. Else, go to step 2.

- If some record is empty, inconsistency has been detected.
- For 2-Coloring, $(2,3)$-consistency test is sufficient.

Local Consistency: Example

Solving 2-Coloring with $(2,3)$-consistency algorithm.



1. Keep record of values for each pair.
2. Consider all triple of variables.
3. Remove 2-incosistent assignments.
4. If no record changed, stop.
5. Else, go to step 2.

- If some record is empty, inconsistency has been detected.
- For 2-Coloring, $(2,3)$-consistency test is sufficient.
- Running time: $O(|V|^3)$.

Local Consistency

- Which CSPs are locally consistent?
- Restrict the set of values and the set of possible constraints.
- $\mathrm{CSP}(\{0,1\}; \neq)$ is $(2,3)$-consistent.
- $\mathrm{CSP}(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is not (ℓ, k)-consistent for any fixed ℓ, k.

Theorem (Feder, Vardi '98)

CSP is solvable by local consistency if and only if it does not have the "ability to count".

Local Consistency

- Which CSPs are locally consistent?
- Restrict the set of values and the set of possible constraints.
- CSP($\{0,1\}; \neq$) is $(2,3)$-consistent.
- CSP($\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1$) is not (ℓ, k)-consistent for any fixed ℓ, k.

Theorem (Feder, Vardi '98)

CSP is solvable by local consistency if and only if it does not have the "ability to count".

Local Consistency

- Which CSPs are locally consistent?
- Restrict the set of values and the set of possible constraints.
- CSP($\{0,1\}; \neq$) is $(2,3)$-consistent.
- CSP($\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1$) is not (ℓ, k)-consistent for any fixed ℓ, k.

Theorem (Feder, Vardi '98)

CSP is solvable by local consistency if and only if it does not have the "ability to count".

- Which CSPs are locally consistent?
- Restrict the set of values and the set of possible constraints.
- $\mathrm{CSP}(\{0,1\}; \neq)$ is $(2,3)$-consistent.
- $\mathrm{CSP}(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is not (ℓ, k)-consistent for any fixed ℓ, k.

Theorem (Feder, Vardi '98)

CSP is solvable by local consistency if and only if it does not have the "ability to count".

Local Consistency

- Which CSPs are locally consistent?
- Restrict the set of values and the set of possible constraints.
- $\mathrm{CSP}(\{0,1\}; \neq)$ is $(2,3)$-consistent.
- $\mathrm{CSP}(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is not (ℓ, k)-consistent for any fixed ℓ, k.

Theorem (Feder, Vardi '98)

CSP is solvable by local consistency if and only if it does not have the "ability to count".

Tractability of CSP

- (ℓ, k)-consistent CSPs are solvable in $O(|V|^k)$ time.
- Gaussian elimination runs in $O(|C|^3)$ time.
- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.

Tractability of CSP

- (ℓ, k)-consistent CSPs are solvable in $O(|V|^k)$ time.
- Gaussian elimination runs in $O(|C|^3)$ time.
- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.

Tractability of CSP

- (ℓ, k)-consistent CSPs are solvable in $O(|V|^k)$ time.
- Gaussian elimination runs in $O(|C|^3)$ time.
- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.

Tractability of CSP

- (ℓ, k)-consistent CSPs are solvable in $O(|V|^k)$ time.
- Gaussian elimination runs in $O(|C|^3)$ time.
- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.

Tractability of CSP

- (ℓ, k)-consistent CSPs are solvable in $O(|V|^k)$ time.
- Gaussian elimination runs in $O(|C|^3)$ time.
- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.

Computational Complexity

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are "the most difficult" in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

Computational Complexity

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are "the most difficult" in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

Computational Complexity

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are "the most difficult" in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

Computational Complexity

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are "the most difficult" in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are “the most difficult” in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

Computational Complexity

A computational problem is

- in P if finding a solution takes polynomial time.
- in NP if verifying a solution takes polynomial time.

Technical (but important) note: in the worst case.

- P $\subseteq$ NP. Whether the inclusion is strict is unknown.
- NP-complete problems are "the most difficult" in NP. Solving one in polynomial time means P=NP.
- Hence, if P$\neq$NP, then NP-complete problems are not solvable in polynomial time.

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P≠NP, and prove NP-completeness.
- CSP($\{0,1\}; \neq$) is in P.
- CSP($\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1$) is in P.
- CSP($\{0,1\}; x + y + z = 1$) is NP-complete.
- CSP($\{0,1,2\}; \neq$) is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" CSP($\{0,1\}; x + y + z = 1$) *is solvable in polynomial time.*

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P≠NP, and prove NP-completeness.
- CSP($\{0,1\}; \neq$) is in P.
- CSP($\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1$) is in P.
- CSP($\{0,1\}; x + y + z = 1$) is NP-complete.
- CSP($\{0,1,2\}; \neq$) is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" CSP($\{0,1\}; x + y + z = 1$) *is solvable in polynomial time.*

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.
- $\mathrm{CSP}(\{0,1\};\neq)$ is in P.
- $\mathrm{CSP}(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is in P.
- $\mathrm{CSP}(\{0,1\}; x + y + z = 1)$ is NP-complete.
- $\mathrm{CSP}(\{0,1,2\};\neq)$ is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" $\mathrm{CSP}(\{0,1\}; x + y + z = 1)$ *is solvable in polynomial time.*

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P≠NP, and prove NP-completeness.
- CSP$(\{0,1\}; \neq)$ is in P.
- CSP$(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is in P.
- CSP$(\{0,1\}; x + y + z = 1)$ is NP-complete.
- CSP$(\{0,1,2\}; \neq)$ is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" CSP$(\{0,1\}; x + y + z = 1)$ *is solvable in polynomial time.*

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P$\neq$NP, and prove NP-completeness.
- CSP$(\{0,1\};\neq)$ is in P.
- CSP$(\{0,1\}; x\oplus y\oplus z = 0, x\oplus y\oplus z = 1)$ is in P.
- CSP$(\{0,1\}; x+y+z=1)$ is NP-complete.
- CSP$(\{0,1,2\};\neq)$ is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" CSP$(\{0,1\}; x+y+z=1)$ *is solvable in polynomial time.*

Tractability of CSP

- Which CSPs are solvable in polynomial time?
- Don't know how to answer such questions unconditionally.
- Instead, assume P≠NP, and prove NP-completeness.
- $\mathrm{CSP}(\{0,1\}; \neq)$ is in P.
- $\mathrm{CSP}(\{0,1\}; x \oplus y \oplus z = 0, x \oplus y \oplus z = 1)$ is in P.
- $\mathrm{CSP}(\{0,1\}; x + y + z = 1)$ is NP-complete.
- $\mathrm{CSP}(\{0,1,2\}; \neq)$ is NP-complete.

Conjecture (Feder, Vardi '98)

Any CSP that cannot "simulate" $\mathrm{CSP}(\{0,1\}; x + y + z = 1)$ *is solvable in polynomial time.*

Dichotomy Conjecture

Conjecture (Feder, Vardi '98)

Any CSP with finite domain is either in P or NP-complete.

- If P$\subsetneq$NP, there are many artificial problems in between.
- In this sense, CSP is a natural computational problem.
- Proved in 2017 by Bulatov and Zhuk (independently).
- Both proofs use *universal algebra*.
- Intuitively, CSP is in P if allowed constraints share "symmetries", and NP-complete otherwise.

Dichotomy Conjecture

Conjecture (Feder, Vardi '98)

Any CSP with finite domain is either in P or NP-complete.

- If P$\subsetneq$NP, there are many artificial problems in between.
- In this sense, CSP is a natural computational problem.
- Proved in 2017 by Bulatov and Zhuk (independently).
- Both proofs use *universal algebra*.
- Intuitively, CSP is in P if allowed constraints share "symmetries", and NP-complete otherwise.

Dichotomy Conjecture

Conjecture (Feder, Vardi '98)

Any CSP with finite domain is either in P or NP-complete.

- If P$\subsetneq$NP, there are many artificial problems in between.
- In this sense, CSP is a natural computational problem.
- Proved in 2017 by Bulatov and Zhuk (independently).
- Both proofs use *universal algebra*.
- Intuitively, CSP is in P if allowed constraints share "symmetries", and NP-complete otherwise.

Dichotomy Conjecture

Theorem (Bulatov, Zhuk, '17)

Any CSP with finite domain is either in P or NP-complete.

- If P$\subsetneq$NP, there are many artificial problems in between.
- In this sense, CSP is a natural computational problem.
- Proved in 2017 by Bulatov and Zhuk (independently).
- Both proofs use *universal algebra*.
- Intuitively, CSP is in P if allowed constraints share "symmetries", and NP-complete otherwise.

Dichotomy Conjecture

Theorem (Bulatov, Zhuk, '17)

Any CSP with finite domain is either in P or NP-complete.

- If P$\subsetneq$NP, there are many artificial problems in between.
- In this sense, CSP is a natural computational problem.
- Proved in 2017 by Bulatov and Zhuk (independently).
- Both proofs use *universal algebra*.
- Intuitively, CSP is in P if allowed constraints share "symmetries", and NP-complete otherwise.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - ▷ Look for approximate solutions (MaxCSP).
  - ▷ Use heuristics (like local consistency)
  - ▷ Design parameterized algorithms.
  - ▷ Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - Look for approximate solutions (MaxCSP).
  - Use heuristics (like local consistency)
  - Design parameterized algorithms.
  - Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - ▷ Look for approximate solutions (MaxCSP).
  - ▷ Use heuristics (like local consistency)
  - ▷ Design parameterized algorithms.
  - ▷ Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - ▷ Look for approximate solutions (MaxCSP).
  - ▷ Use heuristics (like local consistency)
  - ▷ Design parameterized algorithms.
  - ▷ Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - Look for approximate solutions (MaxCSP).
  - Use heuristics (like local consistency)
  - Design parameterized algorithms.
  - Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - Look for approximate solutions (MaxCSP).
  - Use heuristics (like local consistency)
  - Design parameterized algorithms.
  - Design moderately super-polynomial algorithms.

Coping with NP-completeness

- Many interesting CSPs are NP-complete. What to do?
- A problem is in P if we can find an exact solution for any instance in polynomial time.
- We have to give up something
  - Look for approximate solutions (MaxCSP).
  - Use heuristics (like local consistency)
  - Design parameterized algorithms.
  - Design moderately super-polynomial algorithms.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - ▷ CSP($\mathbb{Q}$; <) aka Digraph Acyclicity.
  - ▷ CSP($\mathbb{Q}$; <, =, >) aka Point Algebra.
  - ▷ CSP($\mathbb{Q}$; $x - y \leq a \mid a \in \mathbb{Q}$) aka Simple Temporal Problem.
  - ▷ CSP($\mathbb{Q}$; linear inequalities).
  - ▷ CSP($\mathbb{Z}$; linear inequalities).
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - $\text{CSP}(\mathbb{Q}; <)$ aka Digraph Acyclicity.
  - $\text{CSP}(\mathbb{Q}; <, =, >)$ aka Point Algebra.
  - $\text{CSP}(\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q})$ aka Simple Temporal Problem.
  - $\text{CSP}(\mathbb{Q};$ linear inequalities$)$.
  - $\text{CSP}(\mathbb{Z};$ linear inequalities$)$.
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - CSP($\mathbb{Q}; <$) aka Digraph Acyclicity.
  - CSP($\mathbb{Q}; <, =, >$) aka Point Algebra.
  - CSP($\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q}$) aka Simple Temporal Problem.
  - CSP($\mathbb{Q}$; linear inequalities).
  - CSP($\mathbb{Z}$; linear inequalities).
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - $\mathrm{CSP}(\mathbb{Q}; <)$ aka Digraph Acyclicity.
  - $\mathrm{CSP}(\mathbb{Q}; <, =, >)$ aka Point Algebra.
  - $\mathrm{CSP}(\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q})$ aka Simple Temporal Problem.
  - $\mathrm{CSP}(\mathbb{Q};$ linear inequalities$)$.
  - $\mathrm{CSP}(\mathbb{Z};$ linear inequalities$)$.
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - ▷ $\mathrm{CSP}(\mathbb{Q}; <)$ aka Digraph Acyclicity.
  - ▷ $\mathrm{CSP}(\mathbb{Q}; <, =, >)$ aka Point Algebra.
  - ▷ $\mathrm{CSP}(\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q})$ aka Simple Temporal Problem.
  - ▷ $\mathrm{CSP}(\mathbb{Q};$ linear inequalities$)$.
  - ▷ $\mathrm{CSP}(\mathbb{Z};$ linear inequalities$)$.
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - CSP($\mathbb{Q}; <$) aka Digraph Acyclicity.
  - CSP($\mathbb{Q}; <, =, >$) aka Point Algebra.
  - CSP($\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q}$) aka Simple Temporal Problem.
  - CSP($\mathbb{Q}$; linear inequalities).
  - CSP($\mathbb{Z}$; linear inequalities).
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - $\mathrm{CSP}(\mathbb{Q}; <)$ aka Digraph Acyclicity.
  - $\mathrm{CSP}(\mathbb{Q}; <, =, >)$ aka Point Algebra.
  - $\mathrm{CSP}(\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q})$ aka Simple Temporal Problem.
  - $\mathrm{CSP}(\mathbb{Q}; \text{linear inequalities})$.
  - $\mathrm{CSP}(\mathbb{Z}; \text{linear inequalities})$.
- There is no dichotomy for infinite-domain CSPs.

Infinite Domain CSPs

- What about CSPs with infinite domains?
- Many interesting problems:
  - CSP($\mathbb{Q}; <$) aka Digraph Acyclicity.
  - CSP($\mathbb{Q}; <, =, >$) aka Point Algebra.
  - CSP($\mathbb{Q}; x - y \leq a \mid a \in \mathbb{Q}$) aka Simple Temporal Problem.
  - CSP($\mathbb{Q}$; linear inequalities).
  - CSP($\mathbb{Z}$; linear inequalities).
- There is no dichotomy for infinite-domain CSPs.

Thank You!