On the complexity of #CSP

Martin Dyer

University of Leeds

Counting, Inference, and Optimization on Graphs

Princeton

Thursday, 3rd November, 2011

(joint work with David Richerby)

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Definitions and notation

A *constraint language* Γ is a collection of named relations over a fixed finite set D, the *domain*.

An *instance* has a set of *variables* $V = \{v_1, v_2, \ldots, v_n\}$ and a finite collection of *constraints*, $\mathcal{C}$.

A *constraint* has the form $R(v_{i_1}, \ldots, v_{i_k})$, where $R \in \Gamma$ has *arity* k, and $v_{i_1}, \ldots, v_{i_k} \in V$, not necessarily distinct.

An *assignment* is a mapping $\sigma : V \to D$. It is *satisfying* if $(\sigma(v_1), \ldots, \sigma(v_k)) \in R$, for every constraint in $\mathcal{C}$.

We write $\mathrm{CSP}(\Gamma)$ for CSP with all constraints from Γ.

In *non-uniform CSP*, we regard D and Γ as being fixed constants. We measure the size of the input by the number of variables, n.

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- $\mathrm{CSP}(\Gamma)$: what is the complexity of determining any satisfying assignment for an arbitrary instance?
- $\#\mathrm{CSP}(\Gamma)$: what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with $\#\mathrm{CSP}(\Gamma)$.

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- $\mathrm{CSP}(\Gamma)$: what is the complexity of determining any satisfying assignment for an arbitrary instance?
- $\#\mathrm{CSP}(\Gamma)$: what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with $\#\mathrm{CSP}(\Gamma)$.

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- CSP(Γ): what is the complexity of determining any satisfying assignment for an arbitrary instance?
- #CSP(Γ): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with #CSP(Γ).

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- $\mathrm{CSP}(\Gamma)$: what is the complexity of determining any satisfying assignment for an arbitrary instance?
- $\#\mathrm{CSP}(\Gamma)$: what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with $\#\mathrm{CSP}(\Gamma)$.

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- CSP(Γ): what is the complexity of determining any satisfying assignment for an arbitrary instance?
- #CSP(Γ): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with #CSP(Γ).

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- $\mathrm{CSP}(\Gamma)$: what is the complexity of determining any satisfying assignment for an arbitrary instance?
- $\#\mathrm{CSP}(\Gamma)$: what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with $\#\mathrm{CSP}(\Gamma)$.

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- CSP(Γ): what is the complexity of determining any satisfying assignment for an arbitrary instance?
- #CSP(Γ): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with #CSP(Γ).

Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there *any* satisfying assignment for the given instance?
- Counting: *how many* satisfying assignments are there?

For a given Γ, we can generalise these questions as follows:

- $\mathrm{CSP}(\Gamma)$: what is the complexity of determining any satisfying assignment for an arbitrary instance?
- $\#\mathrm{CSP}(\Gamma)$: what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of n, the number of variables.

Here we will be concerned with $\#\mathrm{CSP}(\Gamma)$.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is #P-complete.

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is #P-complete.

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is #P-complete.

But . . .

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is #P-complete.

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ *is either in* FP *or is* #P*-complete.*

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ *is either in* FP *or is* #P*-complete.*

But . . .

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Dichotomy

Both for decision and counting, it was conjectured that a *dichotomy* exists, between P and NP for decision, and between FP and #P for counting.

For decision, the conjecture remains open. But, for counting, it is settled.

Theorem (Bulatov, 2008)

For all Γ, $\#\mathrm{CSP}(\Gamma)$ is either in FP or is #P-complete.

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (*congruence singularity*) isn't shown to be decidable.

Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for $\#\mathrm{CSP}(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.

  By-product: an improved algorithm for $\mathrm{CSP}(\Gamma)$ when Γ is "strongly rectangular".
- And, most importantly,

  decidability of the new criterion.

Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for $\#\mathrm{CSP}(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.

  By-product: an improved algorithm for $\mathrm{CSP}(\Gamma)$ when Γ is "strongly rectangular".
- And, most importantly,

  decidability of the new criterion.

Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for $\#\mathrm{CSP}(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.

  By-product: an improved algorithm for $\mathrm{CSP}(\Gamma)$ when Γ is "strongly rectangular".
- And, most importantly,

  decidability of the new criterion.

Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for $\#\mathrm{CSP}(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.

  By-product: an improved algorithm for $\mathrm{CSP}(\Gamma)$ when Γ is "strongly rectangular".
- And, most importantly,

  decidability of the new criterion.

Rectangularity

A relation R defined on $A \subseteq D^r$, for some r, is *rectangular* if

$$\left.\begin{array}{l}(\mathbf{a},\mathbf{c})\\(\mathbf{a},\mathbf{d})\\(\mathbf{b},\mathbf{c})\end{array}\right\}\in R \;\Rightarrow\; (\mathbf{b},\mathbf{d})\in R$$

Rectangularity

A relation R defined on $A \subseteq D^r$, for some r, is *rectangular* if

$$\left.\begin{array}{c}(\mathbf{a},\mathbf{c})\\(\mathbf{a},\mathbf{d})\\(\mathbf{b},\mathbf{c})\end{array}\right\} \in R \;\Rightarrow\; (\mathbf{b},\mathbf{d}) \in R$$

Rectangularity

A relation R defined on $A \subseteq D^r$, for some r, is *rectangular* if

$$\left.\begin{array}{l}(\mathbf{a},\mathbf{c})\\(\mathbf{a},\mathbf{d})\\(\mathbf{b},\mathbf{c})\end{array}\right\} \in R \;\Rightarrow\; (\mathbf{b},\mathbf{d}) \in R$$

$$(\mathbf{a},\mathbf{c}) \qquad (\mathbf{a},\mathbf{d})$$

$$(\mathbf{b},\mathbf{c})$$

Rectangularity

A relation R defined on $A \subseteq D^r$, for some r, is *rectangular* if

$$\left.\begin{array}{l} (\mathbf{a},\mathbf{c}) \\ (\mathbf{a},\mathbf{d}) \\ (\mathbf{b},\mathbf{c}) \end{array}\right\} \in R \;\Rightarrow\; (\mathbf{b},\mathbf{d}) \in R$$

$$\begin{array}{ccc} (\mathbf{a},\mathbf{c}) & & (\mathbf{a},\mathbf{d}) \\ & & \downarrow \\ (\mathbf{b},\mathbf{c}) & \rightarrow & (\mathbf{b},\mathbf{d}) \end{array}$$

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Strong rectangularity

A relation is *pp-definable* in Γ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in Γ.

This adds $\exists$ to the operations permissible in $\mathrm{CSP}(\Gamma)$.

Γ is *strongly rectangular* if *every* relation pp-definable in Γ is rectangular.

It's not clear that this is decidable, but we have the well known

Lemma

Γ *is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.*

In view of this, Bulatov & Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of Γ.

Notation

We use the following notation. Let $[n]$ denote $\{1, 2, \ldots, n\}$.

If $J \subseteq [n]$, then $\mathrm{pr}_J R$ is the relation R restricted to the positions in J.

Example: Suppose $D = \{0, 1\}$ and R is the ternary relation with 3-tuples:

$$\begin{array}{c}(0,1,0)\\(0,1,1)\\(1,0,1)\\(1,1,0)\\(1,1,1)\end{array}$$

then $\mathrm{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$\begin{array}{c}(0,1)\\(1,0)\\(1,1)\end{array}$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

Notation

We use the following notation. Let $[n]$ denote $\{1, 2, \ldots, n\}$.

If $J \subseteq [n]$, then $\mathrm{pr}_J R$ is the relation R restricted to the positions in J.

Example: Suppose $D = \{0, 1\}$ and R is the ternary relation with 3-tuples:

$$(0, 1, 0)$$
$$(0, 1, 1)$$
$$(1, 0, 1)$$
$$(1, 1, 0)$$
$$(1, 1, 1)$$

then $\mathrm{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$(0, 1)$$
$$(1, 0)$$
$$(1, 1)$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

Notation

We use the following notation. Let $[n]$ denote $\{1, 2, \ldots, n\}$.

If $J \subseteq [n]$, then $\mathrm{pr}_J R$ is the relation R restricted to the positions in J.

Example: Suppose $D = \{0, 1\}$ and R is the ternary relation with 3-tuples:

$$
\begin{array}{c}
(0,1,0)\\
(0,1,1)\\
(1,0,1)\\
(1,1,0)\\
(1,1,1)
\end{array}
$$

then $\mathrm{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$
\begin{array}{c}
(0,1)\\
(1,0)\\
(1,1)
\end{array}
$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

Notation

We use the following notation. Let $[n]$ denote $\{1, 2, \ldots, n\}$.

If $J \subseteq [n]$, then $\mathrm{pr}_J R$ is the relation R restricted to the positions in J.

Example: Suppose $D = \{0, 1\}$ and R is the ternary relation with 3-tuples:

$$(0,1,0)$$
$$(0,1,1)$$
$$(1,0,1)$$
$$(1,1,0)$$
$$(1,1,1)$$

then $\mathrm{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$(0,1)$$
$$(1,0)$$
$$(1,1)$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

Notation

We use the following notation. Let $[n]$ denote $\{1, 2, \ldots, n\}$.

If $J \subseteq [n]$, then $\mathrm{pr}_J R$ is the relation R restricted to the positions in J.

Example: Suppose $D = \{0, 1\}$ and R is the ternary relation with 3-tuples:

$$\begin{array}{c} (0,1,0) \\ (0,1,1) \\ (1,0,1) \\ (1,1,0) \\ (1,1,1) \end{array}$$

then $\mathrm{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$\begin{array}{c} (0,1) \\ (1,0) \\ (1,1) \end{array}$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots), (u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots), (v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots)$, $(u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots)$, $(v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots)$, $(u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots)$, $(v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots)$, $(u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots)$, $(v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots)$, $(u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots)$, $(v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A *frame* for a relation $R \subseteq D^n$ is any relation $F \subseteq R$ such that:

If, for any $0 \leq i < n$,
R contains a pair of tuples $(u_1, \ldots, u_i, a, \ldots)$, $(u_1, \ldots, u_i, b, \ldots)$,
then F contains a pair of tuples $(v_1, \ldots, v_i, a, \ldots)$, $(v_1, \ldots, v_i, b, \ldots)$.

R is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than R.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

Example

Here is a frame for the complete relation $\{0, 1, 2\}^3$.

$$\begin{array}{lll} (0,0,0) & & \\ (1,0,0) & (0,1,0) & (0,0,1) \\ (2,0,0) & (0,2,0) & (0,0,2) \end{array}$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a, b, b) = \varphi(b, b, a) = a$ is a Mal'tsev polymorphism of D^n.

Example

Here is a frame for the complete relation $\{0, 1, 2\}^3$.

$$\begin{array}{lll} (0,0,0) & & \\ (1,0,0) & (0,1,0) & (0,0,1) \\ (2,0,0) & (0,2,0) & (0,0,2) \end{array}$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a, b, b) = \varphi(b, b, a) = a$ is a Mal'tsev polymorphism of D^n.

Example

Here is a frame for the complete relation $\{0, 1, 2\}^3$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a, b, b) = \varphi(b, b, a) = a$ is a Mal'tsev polymorphism of D^n.

Example

Here is a frame for the complete relation $\{0,1,2\}^3$.

$$\begin{array}{ccc} (0,0,0) & & \\ (1,0,0) & (0,1,0) & (0,0,1) \\ (2,0,0) & (0,2,0) & (0,0,2) \end{array}$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a,b,b) = \varphi(b,b,a) = a$ is a Mal'tsev polymorphism of D^n.

Example

Here is a frame for the complete relation $\{0,1,2\}^3$.

$$\begin{array}{ccc} (0,0,0) & & \\ (1,0,0) & (0,1,0) & (0,0,1) \\ (2,0,0) & (0,2,0) & (0,0,2) \end{array}$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a,b,b) = \varphi(b,b,a) = a$ is a Mal'tsev polymorphism of D^n.

Example

Here is a frame for the complete relation $\{0,1,2\}^3$.

$$\begin{array}{lll} (0,0,0) & & \\ (1,0,0) & (0,1,0) & (0,0,1) \\ (2,0,0) & (0,2,0) & (0,0,2) \end{array}$$

It contains only 7 of the 27 3-tuples in the relation.

Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation D^n (which has $|D|^n$ n-tuples).

The complete relation D^n is trivially strongly rectangular.

For example, any function $\varphi : D^3 \to D$ satisfying $\varphi(a,b,b) = \varphi(b,b,a) = a$ is a Mal'tsev polymorphism of D^n.

Properties of frames

If F is a frame for a strongly rectangular n-ary relation R, with Mal'tsev polymorphism φ:

- $F = \emptyset$ if, and only if, $R = \emptyset$.
- We can recover R from F and φ, by taking the *closure* of F under φ. However, this will take exponential time if R has exponential size.
- In time $\mathcal{O}(n^2|F|^2)$, we can construct a *small frame* for R, which means a frame with at most $n|D|$ n-tuples, if one exists.
- If F is a small frame for R, and $\mathbf{a} \in D^n$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}(n^2)$.

Properties of frames

If F is a frame for a strongly rectangular n-ary relation R, with Mal'tsev polymorphism φ:

- $F = \emptyset$ if, and only if, $R = \emptyset$.
- We can recover R from F and φ, by taking the *closure* of F under φ. However, this will take exponential time if R has exponential size.
- In time $\mathcal{O}(n^2|F|^2)$, we can construct a *small frame* for R, which means a frame with at most $n|D|$ n-tuples, if one exists.
- If F is a small frame for R, and $\mathbf{a} \in D^n$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}(n^2)$.

Properties of frames

If F is a frame for a strongly rectangular n-ary relation R, with Mal'tsev polymorphism φ:

- $F = \emptyset$ if, and only if, $R = \emptyset$.
- We can recover R from F and φ, by taking the *closure* of F under φ. However, this will take exponential time if R has exponential size.
- In time $\mathcal{O}(n^2|F|^2)$, we can construct a *small frame* for R, which means a frame with at most $n|D|$ n-tuples, if one exists.
- If F is a small frame for R, and $\mathbf{a} \in D^n$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}(n^2)$.

Properties of frames

If F is a frame for a strongly rectangular n-ary relation R, with Mal'tsev polymorphism φ:

- $F = \emptyset$ if, and only if, $R = \emptyset$.
- We can recover R from F and φ, by taking the *closure* of F under φ. However, this will take exponential time if R has exponential size.
- In time $\mathcal{O}(n^2|F|^2)$, we can construct a *small frame* for R, which means a frame with at most $n|D|$ n-tuples, if one exists.
- If F is a small frame for R, and $\mathbf{a} \in D^n$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}(n^2)$.

Properties of frames

If F is a frame for a strongly rectangular n-ary relation R, with Mal'tsev polymorphism φ:

- $F = \emptyset$ if, and only if, $R = \emptyset$.
- We can recover R from F and φ, by taking the *closure* of F under φ. However, this will take exponential time if R has exponential size.
- In time $\mathcal{O}(n^2|F|^2)$, we can construct a *small frame* for R, which means a frame with at most $n|D|$ n-tuples, if one exists.
- If F is a small frame for R, and $\mathbf{a} \in D^n$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}(n^2)$.

Frames for strongly rectangular constraints

Let Γ be a strongly rectangular constraint language over domain D.

For an instance I of $\#\mathrm{CSP}(\Gamma)$ with n variables, the set of satisfying assignments can be considered to be an n-ary relation $\Phi \subseteq D^n$.

Then Φ is pp-definable in Γ, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in n, a *small* frame F for Φ, i.e. one with at most $n|D|$ n-tuples.

Frames for strongly rectangular constraints

Let Γ be a strongly rectangular constraint language over domain D.

For an instance I of $\#\mathrm{CSP}(\Gamma)$ with n variables, the set of satisfying assignments can be considered to be an n-ary relation $\Phi \subseteq D^n$.

Then Φ is pp-definable in Γ, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in n, a *small* frame F for Φ, i.e. one with at most $n|D|$ n-tuples.

Frames for strongly rectangular constraints

Let Γ be a strongly rectangular constraint language over domain D.

For an instance I of $\#\mathrm{CSP}(\Gamma)$ with n variables, the set of satisfying assignments can be considered to be an n-ary relation $\Phi \subseteq D^n$.

Then Φ is pp-definable in Γ, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in n, a *small* frame F for Φ, i.e. one with at most $n|D|$ n-tuples.

Frames for strongly rectangular constraints

Let Γ be a strongly rectangular constraint language over domain D.

For an instance I of $\#\mathrm{CSP}(\Gamma)$ with n variables, the set of satisfying assignments can be considered to be an n-ary relation $\Phi \subseteq D^n$.

Then Φ is pp-definable in Γ, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in n, a *small* frame F for Φ, i.e. one with at most $n|D|$ n-tuples.

Bulatov & Dalmau's idea

Let instance I have constraints $C_1, \ldots, C_m$.

For $0 \leq j \leq m$, let I_j be the sub-instance of I with all variables but only constraints $C_1, \ldots, C_j$, determining relation Φ_j.

So, $I_m = I$ and I_0 has no constraints.

Therefore $\Phi_0 = D^n$, and has a small frame (as we've seen).

We must construct, efficiently, a frame for Φ_j, given C_j and a frame F_{j-1} for Φ_{j-1}.

Bulatov & Dalmau's idea

Let instance I have constraints $C_1, \ldots, C_m$.

For $0 \leq j \leq m$, let I_j be the sub-instance of I with all variables but only constraints $C_1, \ldots, C_j$, determining relation Φ_j.

So, $I_m = I$ and I_0 has no constraints.

Therefore $\Phi_0 = D^n$, and has a small frame (as we've seen).

We must construct, efficiently, a frame for Φ_j, given C_j and a frame F_{j-1} for Φ_{j-1}.

Bulatov & Dalmau's idea

Let instance I have constraints $C_1, \ldots, C_m$.

For $0 \le j \le m$, let I_j be the sub-instance of I with all variables but only constraints $C_1, \ldots, C_j$, determining relation Φ_j.

So, $I_m = I$ and I_0 has no constraints.

Therefore $\Phi_0 = D^n$, and has a small frame (as we've seen).

We must construct, efficiently, a frame for Φ_j, given C_j and a frame F_{j-1} for Φ_{j-1}.

Bulatov & Dalmau's idea

Let instance I have constraints $C_1, \ldots, C_m$.

For $0 \leq j \leq m$, let I_j be the sub-instance of I with all variables but only constraints $C_1, \ldots, C_j$, determining relation Φ_j.

So, $I_m = I$ and I_0 has no constraints.

Therefore $\Phi_0 = D^n$, and has a small frame (as we've seen).

We must construct, efficiently, a frame for Φ_j, given C_j and a frame F_{j-1} for Φ_{j-1}.

Bulatov & Dalmau's idea

Let instance I have constraints $C_1, \ldots, C_m$.

For $0 \leq j \leq m$, let I_j be the sub-instance of I with all variables but only constraints $C_1, \ldots, C_j$, determining relation Φ_j.

So, $I_m = I$ and I_0 has no constraints.

Therefore $\Phi_0 = D^n$, and has a small frame (as we've seen).

We must construct, efficiently, a frame for Φ_j, given C_j and a frame F_{j-1} for Φ_{j-1}.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \ldots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\ldots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\ldots,k+1\}}(\Psi \wedge C), \ldots\ldots, \mathrm{pr}_{\{1,\ldots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \ldots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\ldots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\ldots,k+1\}}(\Psi \wedge C), \ldots\ldots, \mathrm{pr}_{\{1,\ldots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \ldots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\ldots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\ldots,k+1\}}(\Psi \wedge C), \ldots\ldots, \mathrm{pr}_{\{1,\ldots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \ldots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\ldots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\ldots,k+1\}}(\Psi \wedge C), \ldots\ldots, \mathrm{pr}_{\{1,\ldots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \ldots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\ldots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\ldots,k+1\}}(\Psi \wedge C), \ldots\ldots, \mathrm{pr}_{\{1,\ldots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Inductive step

Let F be a frame for the relation $\Psi = \Phi_{j-1}$ determined by the constraints $C_1, C_2, \dots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint C_j is $C = R(x_1, \dots, x_k)$.

- For each $i > k$, choose a set $T_i \subseteq F$ from which $\mathrm{pr}_{\{1,\dots,k,i\}}\Psi$ can be reconstructed.
- Remove from each T_i anything that is inconsistent with C.
- Use the resulting sets sequentially to construct "partial frames" for $\mathrm{pr}_{\{1,\dots,k+1\}}(\Psi \wedge C), \dots\dots, \mathrm{pr}_{\{1,\dots,n\}}(\Psi \wedge C) = \Phi_j \wedge C$.

The total time to construct the frame is $\mathcal{O}(n^5)$, if n is the number of variables in $\Phi_n = \Phi$, provided Γ has constant size.

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}.$$

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}.$$

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K$, $j \notin L$, or $i \notin K$, $j \in L$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}.$$

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K$, $j \notin L$, or $i \notin K$, $j \in L$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}.$$

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K$, $j \notin L$, or $i \notin K$, $j \in L$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3), (1,4), (2,3), (2,4), (3,2), (4,1)\}.$$

Block matrices

Let $A = (a_{ij})$ be a $k \times \ell$ non-negative real-valued matrix.

The matrix A has an *underlying relation*

$R_A = \{(i,j) : a_{ij} > 0\} \subseteq [k] \times [\ell]$.

A *block* of A is a set of rows $K \subset [k]$, and a set of columns $L \subset [\ell]$, such that $a_{ij} = 0$ if $i \in K$, $j \notin L$, or $i \notin K$, $j \in L$.

Example: The 4×4 matrix

$$A = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ \cline{2-4} 0 & \multicolumn{1}{|c|}{1} & 0 & 0 \\ \hline 2 & \multicolumn{1}{|c}{0} & 0 & 0 \end{array}\right]$$

has the three blocks shown, and underlying relation

$$R_A = \{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}.$$

Rank-one block matrix matrices

Lemma

Suppose A decomposes into blocks of rank 1. Then

- *R_A is a rectangular relation.*
- *we can recover A from R_A and the row and column sums of A.*

A decomposition of A into blocks of rank 1 corresponds to the existence of a row function $\alpha : [k] \to \mathbb{R}$ and a column function $\beta : [\ell] \to \mathbb{R}$ such that $a_{ij} = \alpha(i)\beta(j)$ for $(i,j) \in R_A$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}, \quad \text{with} \quad \alpha = \left[\begin{array}{c} 1 \\ 2 \\ \hline 1 \\ \hline 1 \end{array}\right], \quad \beta = \left[\begin{array}{c|c|cc} 2 & 1 & 1 & 1 \end{array}\right],$$

is a rank-one block matrix.

Rank-one block matrix matrices

Lemma

Suppose A decomposes into blocks of rank 1. Then

- R_A *is a rectangular relation.*
- *we can recover A from R_A and the row and column sums of A.*

A decomposition of A into blocks of rank 1 corresponds to the existence of a row function $\alpha : [k] \to \mathbb{R}$ and a column function $\beta : [\ell] \to \mathbb{R}$ such that $a_{ij} = \alpha(i)\beta(j)$ for $(i,j) \in R_A$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}, \quad \text{with} \quad \alpha = \left[\begin{array}{c} 1 \\ 2 \\ \hline 1 \\ \hline 1 \end{array}\right], \quad \beta = \left[\begin{array}{c|c|cc} 2 & 1 & 1 & 1 \end{array}\right],$$

is a rank-one block matrix.

Rank-one block matrix matrices

Lemma

Suppose A decomposes into blocks of rank 1. Then

- *R_A is a rectangular relation.*
- *we can recover A from R_A and the row and column sums of A.*

A decomposition of A into blocks of rank 1 corresponds to the existence of a row function $\alpha : [k] \to \mathbb{R}$ and a column function $\beta : [\ell] \to \mathbb{R}$ such that $a_{ij} = \alpha(i)\beta(j)$ for $(i,j) \in R_A$.

Example: The 4×4 matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}, \quad \text{with} \quad \alpha = \left[\begin{array}{c} 1 \\ 2 \\ \hline 1 \\ \hline 1 \end{array}\right], \quad \beta = \left[\begin{array}{c|c|cc} 2 & 1 & 1 & 1 \end{array}\right],$$

is a rank-one block matrix.

Rank-one block matrix matrices

Lemma

Suppose A decomposes into blocks of rank 1. Then

- *R_A is a rectangular relation.*
- *we can recover A from R_A and the row and column sums of A.*

A decomposition of A into blocks of rank 1 corresponds to the existence of a row function $\alpha : [k] \to \mathbb{R}$ and a column function $\beta : [\ell] \to \mathbb{R}$ such that $a_{ij} = \alpha(i)\beta(j)$ for $(i, j) \in R_A$.

Example: The 4×4 matrix

$$A = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ \cline{2-4} 0 & \multicolumn{1}{|c|}{1} & 0 & 0 \\ \cline{1-2} 2 & \multicolumn{1}{|c}{0} & 0 & 0 \end{array}\right], \quad \text{with} \quad \alpha = \left[\begin{array}{c} 1 \\ 2 \\ \hline 1 \\ \hline 1 \end{array}\right], \quad \beta = \left[\begin{array}{c|c|cc} 2 & 1 & 1 & 1 \end{array}\right],$$

is a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x,y) \ = \ \big|\{z : (x,y,z) \in R\}\big| \, .$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x,y) = \alpha(x)\beta(y)$ for $(x,y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$\big\{(1,3,1),\ (1,4,1),\ (1,4,3),\ (2,3,2),\ (2,3,4),\\ (2,4,2),\ (3,2,2),\ (4,1,2),\ (4,1,3)\big\}$$

has balance matrix

$$M \ = \ \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

which is *not* a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x,y) \;=\; \bigl|\{z : (x,y,z) \in R\}\bigr|\,.$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x,y) = \alpha(x)\beta(y)$ for $(x,y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$\bigl\{(1,3,1),\,(1,4,1),\,(1,4,3),\,(2,3,2),\,(2,3,4),\\ (2,4,2),\,(3,2,2),\,(4,1,2),\,(4,1,3)\bigr\}$$

has balance matrix

$$M \;=\; \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

which is *not* a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x, y) = \left|\{z : (x, y, z) \in R\}\right|.$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x, y) = \alpha(x)\beta(y)$ for $(x, y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1, 2, 3, 4\}$, with tuples

$$\big\{(1, 3, 1), (1, 4, 1), (1, 4, 3), (2, 3, 2), (2, 3, 4),$$
$$(2, 4, 2), (3, 2, 2), (4, 1, 2), (4, 1, 3)\big\}$$

has balance matrix

$$M = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

which is *not* a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x, y) = \left|\{z : (x, y, z) \in R\}\right| .$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x, y) = \alpha(x)\beta(y)$ for $(x, y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1, 2, 3, 4\}$, with tuples

$$\{(1, 3, 1), (1, 4, 1), (1, 4, 3), (2, 3, 2), (2, 3, 4), (2, 4, 2), (3, 2, 2), (4, 1, 2), (4, 1, 3)\}$$

has balance matrix

$$M = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

which is *not* a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x,y) \ = \ \left|\{z : (x,y,z) \in R\}\right| .$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x,y) = \alpha(x)\beta(y)$ for $(x,y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$\big\{(1,3,1),\ (1,4,1),\ (1,4,3),\ (2,3,2),\ (2,3,4),\ (2,4,2),\ (3,2,2),\ (4,1,2),\ (4,1,3)\big\}$$

has balance matrix

$$M \ = \ \left[\begin{array}{cc|cc} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ \hline 0 & \multicolumn{1}{|c|}{1} & 0 & 0 \\ \hline 2 & \multicolumn{1}{|c}{0} & 0 & 0 \end{array}\right]$$

which is *not* a rank-one block matrix.

Balance matrices

For a ternary relation R, define its *balance matrix* to be

$$M(x,y) \;=\; \left|\{z : (x,y,z) \in R\}\right| .$$

R is *balanced* if M decomposes into blocks of rank 1

(i.e. if $M(x,y) = \alpha(x)\beta(y)$ for $(x,y) \in \mathrm{pr}_{1,2}R$).

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$\big\{(1,3,1),\ (1,4,1),\ (1,4,3),\ (2,3,2),\ (2,3,4),\\ (2,4,2),\ (3,2,2),\ (4,1,2),\ (4,1,3)\big\}$$

has balance matrix

$$M \;=\; \left[\begin{array}{cc|cc} 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ \hline 0 & \multicolumn{1}{|c|}{1} & 0 & 0 \\ \hline 2 & \multicolumn{1}{|c}{0} & 0 & 0 \end{array}\right]$$

which is *not* a rank-one block matrix.

Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^i \times D^j \times D^k$ $(i, j, k \geq 1,\ i + j + k = r)$.

Example: a relation $R \subseteq D^4$ can be considered as a ternary relation over $D^2 \times D \times D$, in $4!$ ways, by permuting the 4 positions in R.

Γ is *strongly balanced* if every ternary relation derived from every relation pp-definable from Γ is balanced.

Strong balance clearly implies strong rectangularity.

Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^i \times D^j \times D^k$ ($i, j, k \geq 1$, $i + j + k = r$).

Example: a relation $R \subseteq D^4$ can be considered as a ternary relation over $D^2 \times D \times D$, in $4!$ ways, by permuting the 4 positions in R.

Γ is *strongly balanced* if every ternary relation derived from every relation pp-definable from Γ is balanced.

Strong balance clearly implies strong rectangularity.

Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^i \times D^j \times D^k$ $(i, j, k \geq 1,\ i + j + k = r)$.

Example: a relation $R \subseteq D^4$ can be considered as a ternary relation over $D^2 \times D \times D$, in $4!$ ways, by permuting the 4 positions in R.

Γ is *strongly balanced* if every ternary relation derived from every relation pp-definable from Γ is balanced.

Strong balance clearly implies strong rectangularity.

Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^i \times D^j \times D^k$ ($i, j, k \geq 1$, $i + j + k = r$).

Example: a relation $R \subseteq D^4$ can be considered as a ternary relation over $D^2 \times D \times D$, in $4!$ ways, by permuting the 4 positions in R.

Γ is *strongly balanced* if every ternary relation derived from every relation pp-definable from Γ is balanced.

Strong balance clearly implies strong rectangularity.

#P-completeness

We use the following theorem, which strengthens a theorem of Bulatov & Dalmau (2007) concerning strong rectangularity.

Theorem

If Γ is not strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is $\#\mathrm{P}$-complete.

Proof.

Via *weighted* $\#\mathrm{CSP}(\Gamma)$, using a result of Bulatov & Grohe (2005), for partition functions of graph homomorphisms.

From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on Γ implies $\#\mathrm{P}$-completeness. $\square$

#P-completeness

We use the following theorem, which strengthens a theorem of BULATOV & DALMAU (2007) concerning strong rectangularity.

Theorem

If Γ is not strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is $\#\mathrm{P}$-complete.

Proof.

Via *weighted* $\#\mathrm{CSP}(\Gamma)$, using a result of BULATOV & GROHE (2005), for partition functions of graph homomorphisms.

From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on Γ implies $\#\mathrm{P}$-completeness. □

#P-completeness

We use the following theorem, which strengthens a theorem of Bulatov & Dalmau (2007) concerning strong rectangularity.

Theorem

If Γ is not strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is $\#\mathrm{P}$-complete.

Proof.

Via *weighted* $\#\mathrm{CSP}(\Gamma)$, using a result of Bulatov & Grohe (2005), for partition functions of graph homomorphisms.

From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on Γ implies $\#\mathrm{P}$-completeness. $\square$

The counting algorithm

Suppose now that Γ is strongly balanced, and we have a given instance.

First, we compute a small frame F for set of assignments Φ, using the algorithm outlined above.

Assume there are at least two variables, so Φ is at least binary.

For $1 \leq i < j \leq n$, let

$$N_{i,j}(a) = |\{(u_1, \ldots, u_i) : (u_1, \ldots, u_n) \in \Phi \text{ and } u_j = a\}| .$$

Then the total number of satisfying assignments, $N = |\Phi|$, is

$$N = \sum_{a \in D} N_{n-1,n}(a) .$$

The counting algorithm

Suppose now that Γ is strongly balanced, and we have a given instance.

First, we compute a small frame F for set of assignments Φ, using the algorithm outlined above.

Assume there are at least two variables, so Φ is at least binary.

For $1 \leq i < j \leq n$, let

$$N_{i,j}(a) = |\{(u_1, \ldots, u_i) : (u_1, \ldots, u_n) \in \Phi \text{ and } u_j = a\}| .$$

Then the total number of satisfying assignments, $N = |\Phi|$, is

$$N = \sum_{a \in D} N_{n-1,n}(a) .$$

The counting algorithm

Suppose now that Γ is strongly balanced, and we have a given instance.

First, we compute a small frame F for set of assignments Φ, using the algorithm outlined above.

Assume there are at least two variables, so Φ is at least binary.

For $1 \leq i < j \leq n$, let

$$N_{i,j}(a) = |\{(u_1, \ldots, u_i) : (u_1, \ldots, u_n) \in \Phi \text{ and } u_j = a\}| .$$

Then the total number of satisfying assignments, $N = |\Phi|$, is

$$N = \sum_{a \in D} N_{n-1,n}(a) .$$

The counting algorithm

Suppose now that Γ is strongly balanced, and we have a given instance.

First, we compute a small frame F for set of assignments Φ, using the algorithm outlined above.

Assume there are at least two variables, so Φ is at least binary.

For $1 \leq i < j \leq n$, let

$$N_{i,j}(a) = |\{(u_1, \ldots, u_i) : (u_1, \ldots, u_n) \in \Phi \text{ and } u_j = a\}| .$$

Then the total number of satisfying assignments, $N = |\Phi|$, is

$$N = \sum_{a \in D} N_{n-1,n}(a) .$$

The counting algorithm

Suppose now that Γ is strongly balanced, and we have a given instance.

First, we compute a small frame F for set of assignments Φ, using the algorithm outlined above.

Assume there are at least two variables, so Φ is at least binary.

For $1 \leq i < j \leq n$, let

$$N_{i,j}(a) = \left|\{(u_1, \ldots, u_i) : (u_1, \ldots, u_n) \in \Phi \text{ and } u_j = a\}\right|.$$

Then the total number of satisfying assignments, $N = |\Phi|$, is

$$N = \sum_{a \in D} N_{n-1,n}(a).$$

What the $N_{i,j}$ count

If Φ is the relation with tuples in $\mathbf{u} \in D^n$:

$$
\begin{array}{cccccccccc}
(u_{1,1}, & u_{1,2}, & \cdots\cdots & , u_{1,i-1}, & u_{1,i}, & \cdots\cdots, & u_{1,j} & , \cdots\cdots & , u_{1,n}) \\
(u_{2,1}, & u_{2,2}, & \cdots\cdots & , u_{2,i-1}, & u_{2,i}, & \cdots\cdots, & u_{2,j} & , \cdots\cdots & , u_{2,n}) \\
\vdots & \vdots & \cdots\cdots & \vdots & \vdots & \cdots\cdots & \vdots & \cdots\cdots & \vdots \\
\vdots & \vdots & \cdots\cdots & \vdots & \vdots & \cdots\cdots & \vdots & \cdots\cdots & \vdots \\
(u_{N,1}, & u_{N,2}, & \cdots\cdots & , u_{N,i-1}, & u_{N,i}, & \cdots\cdots, & u_{N,j} & , \cdots\cdots & , u_{N,n})
\end{array}
$$

then $N_{i,j}(a) = \left|\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1,j\}}\Phi : u_j = a\}\right|$.

Note that $\mathrm{pr}_{\{1,\dots,i-1,j\}}\Phi$ has fewer than N tuples, in general, because many different tuples in Φ give rise to the same one in $\mathrm{pr}_{\{1,\dots,i-1,j\}}\Phi$.

What the $N_{i,j}$ count

If Φ is the relation with tuples in $\mathbf{u} \in D^n$:

$$
\begin{array}{ccccccccc}
(u_{1,1}, & u_{1,2}, & \cdots\cdots & , u_{1,i-1}, & u_{1,i}, & \cdots\cdots, & u_{1,j} & , \cdots\cdots, & u_{1,n}) \\
(u_{2,1}, & u_{2,2}, & \cdots\cdots & , u_{2,i-1}, & u_{2,i}, & \cdots\cdots, & u_{2,j} & , \cdots\cdots, & u_{2,n}) \\
\vdots & \vdots & \ldots\ldots & \vdots & \vdots & \ldots\ldots & \vdots & \ldots\ldots & \vdots \\
\vdots & \vdots & \ldots\ldots & \vdots & \vdots & \ldots\ldots & \vdots & \ldots\ldots & \vdots \\
(u_{N,1}, & u_{N,2}, & \cdots\cdots & , u_{N,i-1}, & u_{N,i}, & \cdots\cdots, & u_{N,j} & , \cdots\cdots, & u_{N,n})
\end{array}
$$

then $N_{i,j}(a) = \left|\{\mathbf{u} \in \mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi : u_j = a\}\right|$.

Note that $\mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi$ has fewer than N tuples, in general, because many different tuples in Φ give rise to the same one in $\mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi$.

What the $N_{i,j}$ count

If Φ is the relation with tuples in $\mathbf{u} \in D^n$:

$$
\begin{array}{|cccc|ccc|c|ccc}
\hline
(u_{1,1}, & u_{1,2}, & \cdots\cdots & , u_{1,i-1}, & u_{1,i} & , \cdots\cdots & , & u_{1,j} & , \cdots\cdots & , & u_{1,n}) \\
(u_{2,1}, & u_{2,2}, & \cdots\cdots & , u_{2,i-1}, & u_{2,i} & , \cdots\cdots & , & u_{2,j} & , \cdots\cdots & , & u_{2,n}) \\
\vdots & \vdots & \cdots\cdots & \vdots & \vdots & \cdots\cdots & & \vdots & \cdots\cdots & & \vdots \\
\vdots & \vdots & \cdots\cdots & \vdots & \vdots & \cdots\cdots & & \vdots & \cdots\cdots & & \vdots \\
(u_{N,1}, & u_{N,2}, & \cdots\cdots & , u_{N,i-1}, & u_{N,i} & , \cdots\cdots & , & u_{N,j} & , \cdots\cdots & , & u_{N,n}) \\
\hline
\end{array}
$$

then $N_{i,j}(a) = \left| \{ \mathbf{u} \in \mathrm{pr}_{\{1,\ldots,i-1,j\}} \Phi : u_j = a \} \right|$.

Note that $\mathrm{pr}_{\{1,\ldots,i-1,j\}} \Phi$ has fewer than N tuples, in general, because many different tuples in Φ give rise to the same one in $\mathrm{pr}_{\{1,\ldots,i-1,j\}} \Phi$.

What the $N_{i,j}$ count

If Φ is the relation with tuples in $\mathbf{u} \in D^n$:

$$
\begin{array}{|cccc|ccc|c|ccc}
\hline
(\, u_{1,1}, & u_{1,2}, & \cdots\cdots & , u_{1,i-1}, & u_{1,i} & , \cdots\cdots & , & u_{1,j} & , \cdots\cdots & , u_{1,n} &) \\
(\, u_{2,1}, & u_{2,2}, & \cdots\cdots & , u_{2,i-1}, & u_{2,i} & , \cdots\cdots & , & u_{2,j} & , \cdots\cdots & , u_{2,n} &) \\
\vdots & \vdots & \ldots\ldots & \vdots & \vdots & \ldots\;\ldots & & \vdots & \ldots\;\ldots & \vdots & \\
\vdots & \vdots & \ldots\ldots & \vdots & \vdots & \ldots\;\ldots & & \vdots & \ldots\;\ldots & \vdots & \\
(\, u_{N,1}, & u_{N,2}, & \cdots\cdots & , u_{N,i-1}, & u_{N,i} & , \cdots\cdots & , & u_{N,j} & , \cdots\cdots & , u_{N,n} &) \\
\hline
\end{array}
$$

then $N_{i,j}(a) = \left|\{\mathbf{u} \in \mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi : u_j = a\}\right|$.

Note that $\mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi$ has fewer than N tuples, in general, because many different tuples in Φ give rise to the same one in $\mathrm{pr}_{\{1,\ldots,i-1,j\}}\Phi$.

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \leq |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\dots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\dots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{array}{l}(\mathbf{u}, x, y)\\(\mathbf{u}', x, y)\\(\mathbf{u}', x', y')\end{array}\right\} \in \Lambda \;\Rightarrow\; (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \leq |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\dots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\dots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{matrix} (\mathbf{u}, x, y) \\ (\mathbf{u}', x, y) \\ (\mathbf{u}', x', y') \end{matrix}\right\} \in \Lambda \;\Rightarrow\; (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \le |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\dots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\dots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{matrix} (\mathbf{u}, x, y) \\ (\mathbf{u}', x, y) \\ (\mathbf{u}', x', y') \end{matrix}\right\} \in \Lambda \;\Rightarrow\; (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \leq |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\dots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\dots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{array}{l}(\mathbf{u}, x, y)\\(\mathbf{u}', x, y)\\(\mathbf{u}', x', y')\end{array}\right\} \in \Lambda \;\Rightarrow\; (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \leq |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\dots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\dots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\dots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{array}{l} (\mathbf{u}, x, y) \\ (\mathbf{u}', x, y) \\ (\mathbf{u}', x', y') \end{array}\right\} \in \Lambda \ \Rightarrow\ (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: rectangularity

Each $N_{1,j}$ can be calculated easily, because $|\mathrm{pr}_{1,j}\Phi| \leq |D|^2 = \mathcal{O}(1)$.

Suppose we have computed each $N_{i-1,j}$, for some i.

We consider $\Lambda = \mathrm{pr}_{\{1,\ldots,i,j\}}\Phi$ to be a ternary relation on

$$\mathrm{pr}_{\{1,\ldots,i-1\}}\Phi \times \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi.$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_i\Phi \times \mathrm{pr}_j\Phi$, the sets $\{\mathbf{u} \in \mathrm{pr}_{\{1,\ldots,i-1\}}\Phi : (\mathbf{u}, x, y) \in \Lambda\}$ are *disjoint* or *identical*.

This follows by *rectangularity*:

$$\left.\begin{array}{l}(\mathbf{u}, x, y) \\ (\mathbf{u}', x, y) \\ (\mathbf{u}', x', y')\end{array}\right\} \in \Lambda \;\Rightarrow\; (\mathbf{u}, x', y') \in \Lambda.$$

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$(x,y) \equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\}$$
$$(x,y) \not\equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\} = \emptyset.$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x,y) = |\{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x,a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$\begin{aligned}(x,y) \equiv (x',y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\}\\ (x,y) \not\equiv (x',y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\} = \emptyset.\end{aligned}$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x,y) = |\{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x,a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$\begin{aligned} (x, y) \equiv (x', y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u}, x', y') \in \Lambda\} \\ (x, y) \not\equiv (x', y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u}, x', y') \in \Lambda\} = \emptyset. \end{aligned}$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x, y) = |\{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$\begin{aligned}(x,y) \equiv (x',y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\}\\ (x,y) \not\equiv (x',y') \quad &\Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\} = \emptyset.\end{aligned}$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x,y) = |\{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x,a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation:

$$(x,y) \equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\}$$
$$(x,y) \not\equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\} = \emptyset.$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x,y) = |\{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x,a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$(x,y) \equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\}$$
$$(x,y) \not\equiv (x',y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u},x',y') \in \Lambda\} = \emptyset.$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x,y) = |\{\mathbf{u} : (\mathbf{u},x,y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x,a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Computing the $N_{i,j}$: strong balance

Using a frame F for Φ, we can determine an equivalence relation :

$$(x, y) \equiv (x', y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\} = \{\mathbf{u} : (\mathbf{u}, x', y') \in \Lambda\}$$
$$(x, y) \not\equiv (x', y') \quad \Leftrightarrow \quad \{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\} \cap \{\mathbf{u} : (\mathbf{u}, x', y') \in \Lambda\} = \emptyset.$$

Now, since Λ is pp-definable in Γ, it is balanced. Therefore, the matrix

$$M(x, y) = |\{\mathbf{u} : (\mathbf{u}, x, y) \in \Lambda\}|$$

is a rank-one block matrix, and $N_{i,j}(a) = \sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of M under the equivalence $\equiv$.

Using F again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1,i}$ and $N_{i-1,j}$.

Hence we can determine $\widehat{M}$, and then M, and finally $N_{i,j}$.

Dichotomy theorem

Therefore we have

Theorem

If Γ is strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is computable in time $\mathcal{O}(n^5)$. Otherwise, it is $\#\mathrm{P}$-complete.

We can prove that strong balance is equivalent to the *congruence singularity* criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given Γ?

The answer to this question is yes and, in fact, it is decidable in NP.

Dichotomy theorem

Therefore we have

Theorem

If Γ is strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is computable in time $\mathcal{O}(n^5)$. Otherwise, it is $\#\mathrm{P}$-complete.

We can prove that strong balance is equivalent to the *congruence singularity* criterion of BULATOV (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given Γ?

The answer to this question is yes and, in fact, it is decidable in NP.

Dichotomy theorem

Therefore we have

Theorem

If Γ is strongly balanced, then $\#\mathrm{CSP}(\Gamma)$ is computable in time $\mathcal{O}(n^5)$. Otherwise, it is $\#\mathrm{P}$-complete.

We can prove that strong balance is equivalent to the *congruence singularity* criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given Γ?

The answer to this question is yes and, in fact, it is decidable in NP.

Dichotomy theorem

Therefore we have

Theorem

If Γ *is strongly balanced, then* $\#\mathrm{CSP}(\Gamma)$ *is computable in time* $\mathcal{O}(n^5)$*. Otherwise, it is* $\#\mathrm{P}$*-complete.*

We can prove that strong balance is equivalent to the *congruence singularity* criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given Γ ?

The answer to this question is yes and, in fact, it is decidable in NP.

The question

Is the following problem decidable?

Input: a constraint language Γ
Question: is Γ strongly balanced?

And, if so, what is its computational complexity?

Note that D and Γ are not fixed parameters in this meta-problem, though they were in the dichotomy theorem.

The question

Is the following problem decidable?

Input: a constraint language Γ
Question: is Γ strongly balanced?

And, if so, what is its computational complexity?

Note that D and Γ are not fixed parameters in this meta-problem, though they were in the dichotomy theorem.

The question

Is the following problem decidable?

Input: a constraint language Γ
Question: is Γ strongly balanced?

And, if so, what is its computational complexity?

Note that D and Γ are not fixed parameters in this meta-problem, though they were in the dichotomy theorem.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in } \mathrm{FP} \\ &\implies \text{the problem isn't } \#\mathrm{P}\text{-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in } \mathrm{FP} \\ &\implies \text{the problem isn't } \#\mathrm{P}\text{-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in FP} \\ &\implies \text{the problem isn't \#P-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in FP} \\ &\implies \text{the problem isn't \#P-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in FP} \\ &\implies \text{the problem isn't \#P-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in FP} \\ &\implies \text{the problem isn't \#P-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in FP} \\ &\implies \text{the problem isn't \#P-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned}\text{strong balance} &\implies \text{almost strong balance}\\ &\implies \text{the algorithm works}\\ &\implies \text{the problem is in } \mathrm{FP}\\ &\implies \text{the problem isn't } \#\mathrm{P}\text{-complete}\\ &\implies \text{strong balance,}\end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A weaker condition

We can relax the strong balance criterion to a more useful condition which we call *almost-strong* balance.

An constraint language Γ with domain D is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^k \times D \times D$, for some k, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$\begin{aligned} \text{strong balance} &\implies \text{almost strong balance} \\ &\implies \text{the algorithm works} \\ &\implies \text{the problem is in } \mathrm{FP} \\ &\implies \text{the problem isn't } \#\mathrm{P}\text{-complete} \\ &\implies \text{strong balance,} \end{aligned}$$

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \#\mathrm{P}$.

A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

Lemma

M is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$u^2x^2vw = v^2w^2ux$$

for every 2×2 submatrix $\left(\begin{smallmatrix} u & v \\ w & x \end{smallmatrix}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$M(a,c)^2M(b,d)^2M(a,d)M(b,c) = M(a,d)^2M(b,c)^2M(a,c)M(b,d)$$

for all $a, b, c, d \in D$ and every $M = M(R)$, $R \subseteq D^k \times D \times D$.

A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

Lemma

M is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$u^2x^2vw = v^2w^2ux$$

for every 2×2 submatrix $\left(\begin{smallmatrix} u & v \\ w & x \end{smallmatrix}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$M(a,c)^2M(b,d)^2M(a,d)M(b,c) = M(a,d)^2M(b,c)^2M(a,c)M(b,d)$$

for all $a, b, c, d \in D$ and every $M = M(R)$, $R \subseteq D^k \times D \times D$.

A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

Lemma

M is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$u^2x^2vw = v^2w^2ux$$

for every 2×2 submatrix $\begin{pmatrix} u & v \\ w & x \end{pmatrix}$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$M(a,c)^2M(b,d)^2M(a,d)M(b,c) = M(a,d)^2M(b,c)^2M(a,c)M(b,d)$$

for all $a, b, c, d \in D$ and every $M = M(R)$, $R \subseteq D^k \times D \times D$.

A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

Lemma

M is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$u^2x^2vw = v^2w^2ux$$

for every 2×2 submatrix $\left(\begin{smallmatrix} u & v \\ w & x \end{smallmatrix}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$M(a,c)^2M(b,d)^2M(a,d)M(b,c) = M(a,d)^2M(b,c)^2M(a,c)M(b,d)$$

for all $a,b,c,d \in D$ and every $M = M(R)$, $R \subseteq D^k \times D \times D$.

A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

Lemma

M is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$u^2x^2vw = v^2w^2ux$$

for every 2×2 submatrix $\begin{pmatrix} u & v \\ w & x \end{pmatrix}$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$M(a,c)^2M(b,d)^2M(a,d)M(b,c) = M(a,d)^2M(b,c)^2M(a,c)M(b,d)$$

for all $a, b, c, d \in D$ and every $M = M(R)$, $R \subseteq D^k \times D \times D$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Cartesian powers

We will recast this as a problem in D^6. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^6$ to $abcdef$.

Now, using the usual definition of *Cartesian powers* of a finite structure, we can define a new constraint language Γ' over D^6, and translate the relation $R \subseteq D^k$ to $R' \subseteq (D^6)^k$, with corresponding balance matrix M'.

Our condition for Γ to be strongly balanced then becomes that

$$M'(aabbab, ccdddc) = M'(aabbab, ddcccd)$$

for all $a, b, c, d \in D$ and all balance matrices M' of these translated relations R'.

We will write $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^6$.

Automorphisms

Our condition is then that $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for all $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form) and all $M' = M'(R')$.

This means that, for every R', the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LOVÁSZ (1967), we can show that this happens if and only if, for every $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form), there exists an automorphism η of Γ' with $\eta(\bar{a}) = \bar{a}$ and $\eta(\bar{b}) = \bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

Automorphisms

Our condition is then that $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for all $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form) and all $M' = M'(R')$.

This means that, for every R', the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LOVÁSZ (1967), we can show that this happens if and only if, for every $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form), there exists an automorphism η of Γ' with $\eta(\bar{a}) = \bar{a}$ and $\eta(\bar{b}) = \bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

Automorphisms

Our condition is then that $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for all $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form) and all $M' = M'(R')$.

This means that, for every R', the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LOVÁSZ (1967), we can show that this happens if and only if, for every $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form), there exists an automorphism η of Γ' with $\eta(\bar{a}) = \bar{a}$ and $\eta(\bar{b}) = \bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

Automorphisms

Our condition is then that $M'(\bar{a}, \bar{b}) = M'(\bar{b}, \bar{c})$ for all $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form) and all $M' = M'(R')$.

This means that, for every R', the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LOVÁSZ (1967), we can show that this happens if and only if, for every $\bar{a}$, $\bar{b}$, $\bar{c}$ (of appropriate form), there exists an automorphism η of Γ' with $\eta(\bar{a}) = \bar{a}$ and $\eta(\bar{b}) = \bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

Decidability

Theorem

Strong balance is decidable in NP.

Proof.

First verify that Γ is strongly rectangular. If not, answer **no**. If so:

Construct Γ' and, for each $\bar{a}, \bar{b}, \bar{c}$ of the required form, nondeterministically guess a function $\eta\colon D^6 \to D^6$. Check that these functions are the required automorphisms. If so, answer **yes**, otherwise answer **no**. □

As a corollary, we have

Theorem

Congruence singularity is decidable in NP.

Decidability

Theorem

Strong balance is decidable in NP*.*

Proof.

First verify that Γ is strongly rectangular. If not, answer **no**. If so:

Construct Γ' and, for each $\bar{a}, \bar{b}, \bar{c}$ of the required form, nondeterministically guess a function $\eta\colon D^6 \to D^6$. Check that these functions are the required automorphisms. If so, answer **yes**, otherwise answer **no**. $\square$

As a corollary, we have

Theorem

Congruence singularity is decidable in NP*.*

Decidability

Theorem

Strong balance is decidable in NP.

Proof.

First verify that Γ is strongly rectangular. If not, answer **no**. If so:

Construct Γ' and, for each $\bar{a}, \bar{b}, \bar{c}$ of the required form, nondeterministically guess a function $\eta\colon D^6 \to D^6$. Check that these functions are the required automorphisms. If so, answer **yes**, otherwise answer **no**. □

As a corollary, we have

Theorem

Congruence singularity is decidable in NP.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.

It is not difficult to show that strong balance is reducible to the *graph isomorphism* problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

Open questions and further work

- Can the algorithm for handling strongly rectangular relations be made more efficient ? It is $\mathcal{O}(n^5)$, but there seems no reason why is should be worse than matrix multiplication: $\mathcal{O}(n^{2.376})$. This computation dominates the counting algorithm.
- What about *weighted* #CSP ?

  Bulatov, Dyer, Goldberg, Jalsenius, Jerrum & Richerby (2010) extended the dichotomy to *rational*-weighted #CSP.

  Cai, Chen & Lu (2011) have extended this result to *algebraic* weights by developing the approach used here.

  More generally, *negative* or *complex* weights can be considered. Dichotomies were known only in special cases, but Cai & Chen (2011) have recently obtained the "ultimate" generalisation: to *complex algebraic* weights.

Open questions and further work

- Can the algorithm for handling strongly rectangular relations be made more efficient ? It is $\mathcal{O}(n^5)$, but there seems no reason why is should be worse than matrix multiplication: $\mathcal{O}(n^{2.376})$. This computation dominates the counting algorithm.
- What about *weighted* #CSP ?

  Bulatov, Dyer, Goldberg, Jalsenius, Jerrum & Richerby (2010) extended the dichotomy to *rational*-weighted #CSP.

  Cai, Chen & Lu (2011) have extended this result to *algebraic* weights by developing the approach used here.

  More generally, *negative* or *complex* weights can be considered. Dichotomies were known only in special cases, but Cai & Chen (2011) have recently obtained the "ultimate" generalisation: to *complex algebraic* weights.

Open questions and further work

- What new or known special cases (for restricted classes of Γ) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.

- What can be said if restrictions are placed on the *instance*? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.

- What can be said for *approximate* counting? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a *trichotomy* for the Boolean domain.

- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in P (e.g. via a special case of GI)?

Open questions and further work

- What new or known special cases (for restricted classes of Γ) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.

- What can be said if restrictions are placed on the *instance*? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.

- What can be said for *approximate* counting? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a *trichotomy* for the Boolean domain.

- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in P (e.g. via a special case of GI)?

Open questions and further work

- What new or known special cases (for restricted classes of Γ) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.

- What can be said if restrictions are placed on the *instance*? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.

- What can be said for *approximate* counting? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a *trichotomy* for the Boolean domain.

- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in P (e.g. via a special case of GI)?

Open questions and further work

- What new or known special cases (for restricted classes of Γ) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.
- What can be said if restrictions are placed on the *instance*? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.
- What can be said for *approximate* counting? It seems unlikely that a simple dichotomy exists, but D, GOLDBERG AND JERRUM (2010) have given a *trichotomy* for the Boolean domain.
- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in P (e.g. via a special case of GI)?