The Symmetry Gap in Combinatorial Optimization

Anuj Dawar

Based on joint papers with Matthew Anderson, Bjarki Holm and Pengming Wang

University of Cambridge
Department of Computer Science and Technology

Logic and Computational Complexity
Shonan, 21 September 2017

Fixed-Point Logic with Counting

FPC—*Fixed-Point with Counting* is an extension of first-order logic with a *recursion operator* and a mechanism for *counting*.

If $\varphi(x)$ is a formula with free variable x, then $\#x\varphi$ is a term denoting the number of elements satisfying φ.

Formulae of FPC:

- all atomic formulae as in FP;
- $\tau_1 < \tau_2$; $\tau_1 = \tau_2$ where τ_i is a term of numeric sort;
- $\exists x\, \varphi$; $\exists \nu\, \varphi$; where ν is a variable ranging over numbers up to the size of the domain;
- $[\mathbf{lfp}_{X,\mathbf{x},\nu}\varphi](\mathbf{t})$; and
- $\varphi \wedge \psi$; $\neg\varphi$.

Fixed-Point Logic with Counting

FPC is the class of *decision problems* definable in *fixed-point logic with counting*.

> *The decision problems are (isomorphism-closed) classes (or properties) of finite structures (such as graphs, Boolean formulas, systems of equations).*

Every problem in FPC is in P;

Expressive Power of FPC

Most *"obviously"* polynomial-time algorithms can be expressed in FPC.

This includes P-complete problems such as

> CVP—*the Circuit Value Problem*
> *Input: a circuit, i.e. a labelled DAG with source labels from* $\{0, 1\}$*, internal node labels from* $\{\vee, \wedge, \neg\}$*.*
> *Decide: what is the value at the output gate.*

CVP is expressible in FPC.

It is expressible in FPC also for circuits that may include *threshold or counting gates*.

Expressive Power of FPC

Many non-trivial polynomial-time algorithms can be expressed in FPC:

FPC captures all of P over any *proper minor-closed class of graphs* **(Grohe 2010)**

But some cannot be expressed:

- There are polynomial-time decidable properties of graphs that are not definable in FPC. **(Cai, Fürer, Immerman, 1992)**
- *XOR-Sat*, or more generally, solvability of a system of linear equations over a finite field cannot be expressed in FPC. **(Atserias, Bulatov, D. 2009)**

Some NP-complete problems are *provably* not in FPC, including *Sat*, *Hamiltonicity* and *3-colouraiblity*.

Circuit Complexity

A *language* $L \subseteq \{0,1\}^*$ can be described by a family of *Boolean functions*:

$$(f_n)_{n \in \omega} : \{0,1\}^n \to \{0,1\}.$$

Each f_n may be computed by a *circuit* C_n made up of

- Gates labeled by Boolean operators: $\wedge, \vee, \neg$,
- Boolean inputs: $x_1, \ldots, x_n$, and
- A distinguished gate determining the output.

If there is a polynomial $p(n)$ bounding the *size* of C_n, i.e. the number of gates in C_n, the language L is in the class P/poly.

If, in addition, the function $n \mapsto C_n$ is computable in *polynomial time*, L is in P.

Note: For these classes it makes no difference whether the circuits only use $\{\wedge, \vee, \neg\}$ or a richer basis with *threshold* or *majority* gates.

Symmetric Circuits

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *symmetric* if it is invariant under *all* permutations of its inputs.

A Boolean function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is *graph-invariant* if it is invariant under the natural action of S_n on its inputs.

A circuit C computing $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is *symmetric* if the action of any permutation in S_n on the inputs can be extended to an *automorphism* of C.

A graph property is in FPC *if, and only if,* it is decided by a P-uniform family of *symmetric* circuits using *symmetric gates*.

Weisfeiler-Leman Equivalences

The k-*dimensional Weisfeiler-Leman* equivalence relation is an *overapproximation* of the isomorphism relation.

If G, H are n-vertex graphs and $k < n$, we have:

$$G \cong H \quad \Leftrightarrow \quad G \equiv^n H \quad \Rightarrow \quad G \equiv^{k+1} H \quad \Rightarrow \quad G \equiv^k H.$$

$G \equiv^k H$ is decidable in time $n^{O(k)}$.

It has many equivalent characterisations arising from

- *combinatorics* **(Babai)**
- *logic* **(Immerman-Lander)**
- *algebra* **(Weisfeiler; Holm)**
- *linear optimization* **(Atserias-Maneva; Malkin)**

Weisfeiler-Leman Equivalences

$G \equiv^k H$ iff G and H cannot be distinguished by a sentence of first-order logic with *counting quantifiers* using only $k+1$ variables.

G and H are not distinguished by the coarsest partition of the k-tuples of G into classes $P_1, \ldots, P_t$ satisfying:

> *two tuples* $\mathbf{u}$ *and* $\mathbf{v}$ *in the same class* P_i *cannot be distinguished by counting the number of substitutions we can make in them to get a tuple in class* P_j*.*

Graph Isomorphism Integer Program

Yet another way of approximating the *graph isomorphism relation* is obtained by considering it as a *0/1 linear program*.

If A and B are adjacency matrices of graphs G and H, then $G \cong H$ if, and only if, there is a *permutation matrix* P such that:

$$PAP^{-1} = B \quad \text{or, equivalently} \quad PA = BP$$

Introducing a variable x_{ij} for each entry of P and adding the constraints:

$$\sum_i x_{ij} = 1 \quad \text{and} \quad \sum_j x_{ij} = 1$$

we get a system of equations that has a *0-1 solution* if, and only if, G and H are isomorphic.

Fractional Isomorphism

To the system of equations:

$$PA = BP; \quad \sum_i x_{ij} = 1 \quad \text{and} \quad \sum_j x_{ij} = 1$$

add the inequalities

$$0 \leq x_{ij} \leq 1.$$

Say that G and H are *fractionally isomorphic* ($G \cong^f H$) if the resulting system has *any real solution*.

$G \cong^f H$ if, and only if, $G \equiv^1 H$.

(Ramana, Scheiermann, Ullman 1994)

Sherali-Adams Hierarchy

If we have any *linear program* for which we seek a *0-1 solution*, we can relax the constraint and admit *fractional solutions*.

The resulting linear program can be solved in *polynomial time*, but admits solutions which are not solutions to the original problem.

Sherali and Adams (1990) define a way of *tightening* the linear program by adding a number of *lift and project* constraints.

Sherali-Adams Hierarchy

The kth *lift-and-project* of a linear program is defined as follows:

For each constraint $\mathbf{a}^T\mathbf{x} \leq b$ in the linear program, and each set I of variables with $|I| < k$ and $J \subseteq I$, multiply the constraint by

$$\prod_{i \in I \setminus J} x_i \prod_{j \in J} (1 - x_j)$$

and then *linearize* by replacing x_i^2 by x_i and $\prod_{j \in K} x_j$ by a new variable y_K for each set K (along with constraints: $y_\emptyset = 1$, $y_{\{x\}} = x$ and $y_K \leq y_{K'}$ for $K' \subseteq K$).

Say that $G \cong^{f,k} H$ if the kth lift-and-project of the *isomorphism program* on G and H admits a solution.

Sherali-Adams Isomorphism

For each k

$$G \equiv^k H \quad \Rightarrow \quad G \cong^{f,k} H \quad \Rightarrow \quad G \equiv^{k-1} H$$

(Atserias, Maneva 2012)

For $k > 2$, the reverse implications fail. **(Grohe, Otto 2012)**

Counting Width

For any class of structures $\mathcal{C}$, we define its *counting width* $\nu_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ so that

> *$\nu_{\mathcal{C}}(n)$ is the least k such that $\mathcal{C}$ restricted to structures with at most n elements is closed under $\equiv^k$.*

Every class in FPC has counting width bounded by a *constant*.

3-Sat, *XOR-Sat*, *3-Colourability* all have counting width $\Omega(n)$.

FPC-Reductions

If $\mathcal{C} \leq_{\mathsf{FPC}} \mathcal{D}$ then

$$\nu_{\mathcal{D}} = \Omega(\nu_{\mathcal{C}}^{1/d}).$$

If the reduction takes $\mathcal{C}$-instances to $\mathcal{D}$-instances of *linear size*, then

$$\nu_{\mathcal{D}} = \Omega(\nu_{\mathcal{C}}).$$

Known linear lower bounds follow from $\nu_{\mathsf{XOR\text{-}Sat}} = \Omega(n)$.

Linear Programming

Linear Programming is an important algorithmic tool for solving a large variety of optimization problems.

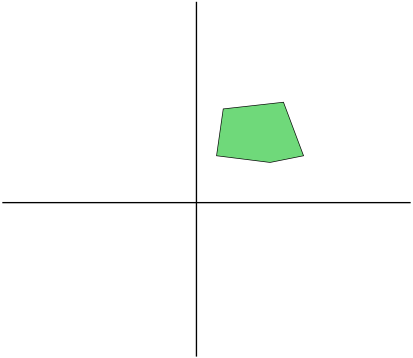
It was shown by **(Khachiyan 1980)** that linear programming problems can be solved in polynomial time.
We have a set C of *constraints* over a set V of *variables*.
Each $c \in C$ consists of $a_c \in \mathbb{Q}^V$ and $b_c \in \mathbb{Q}$.

Feasibility Problem: Given a linear programming instance, determine if there is an $x \in \mathbb{Q}^V$ such that:

$$a_c^T x \leq b_c \quad \text{for all } c \in C$$

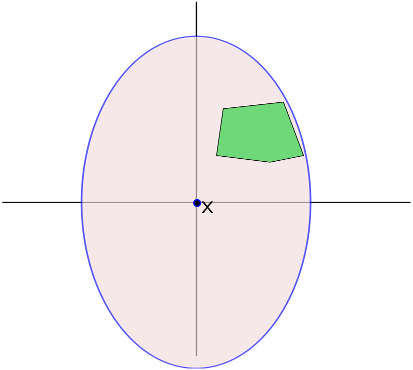
We show that this, and the corresponding *optimization problem* are expressible in FPC.

Ellipsoid Method



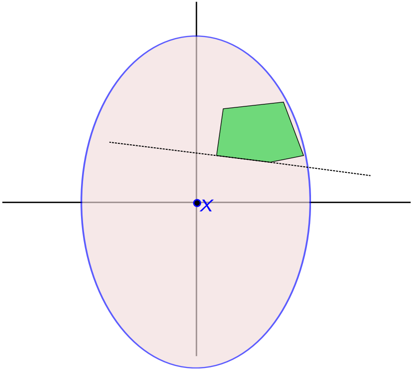
The set of constraints determines a *polytope*

Ellipsoid Method



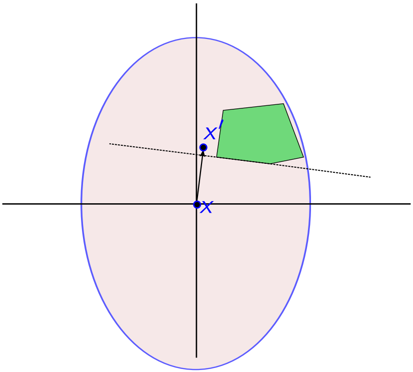
Start at the origin and calculate an *ellipsoid* enclosing it.

Ellipsoid Method



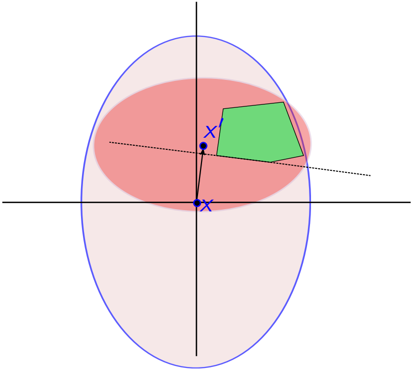
If the centre is not in the polytope, choose a constraint it *violates*.

Ellipsoid Method



Calculate a new *centre*.

Ellipsoid Method



And a new ellipsoid around the centre of at most *half* the volume.

Ellipsoid Method in FPC

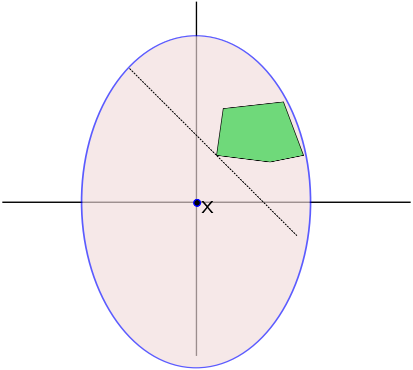
We can encode all the calculations involved in FPC.

This relies on expressing algebraic manipulations of *unordered* matrices.

What is not obvious is how to *choose* the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some *separating hyperplane*.

Ellipsoid Method in FPC



Ellipsoid Method in FPC

We can encode all the calculations involved in FPC.

This relies on expressing algebraic manilpulations of *unordered* matrices.

What is not obvious is how to *choose* the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some *separating hyperplane*.

So, we can take:

$$(\sum_{c \in S} a_c)^T x \leq \sum_{c \in S} b_c$$

where S is the *set* of all violated constraints.

Separation Oracle

More generally, the ellipsoid method can be used, even when the *constraint matrix* is not given explicitly, as long as we can always determine a *separating hyperplane*.

In particular, the polytope represented may have *exponentially many* facets.

We show that as long as the *separation oracle* can be defined in FPC, the corresponding *optimization problem* can be solved in FPC.

Graph Matching

Recall, in a *graph* $G = (V, E)$ a matching $M \subset E$ is a set of edges such that each vertex is incident on *at most* one edge in M.

(Blass, Gurevich, Shelah 1999) showed that for *bipartite* graphs this is definable in FPC.

We consider the more general problem of determining the *maximum weight* of a matching in a *weighted graph*:

$$G = (V, E) \quad w : E \to \mathbb{Q}_{\geq 0}$$

The Matching Polytope

(Edmonds 1965) showed that the problem of finding a *maximum weight matching* in $G = (V, E)$ $\quad w : \mathbb{Q}^E_{\geq 0}$ can be expressed as the following linear programming problem

$$\max w^\top y \qquad \text{subject to}$$

$$Ay \leq 1^V,$$

$$y_e \geq 0, \;\; \forall e \in E,$$

$$\sum_{e \in E \cap W^2} y_e \leq \frac{1}{2}(|W| - 1), \;\; \forall W \subseteq V \text{ with } |W| \text{ odd},$$

Matching in FPC

We show that a *separation oracle* for this polytope is definable by an FPC formula interpreted in the weighted graph G.

As a consequence, there is an FPC formula defining the *size* of the maximum matching in G.

Note that this does not allow us to define an *actual* matching.

Finite Valued CSPs

Finite Valued CSPs generalize various Max-CSP problems.

Such a problem is given by a finite domain D and a collection Γ of functions $f : D^k \to \mathbb{Q}_+$.

An instance is a set V of variables along with constraints $c = (\mathbf{x}, f, w)$ for $\mathbf{x} \in V^k$ and $w \in \mathbb{Q}_+$.

Find an assignment $h : V \to D$ which minimizes

$$\sum_c wf(h\mathbf{x})$$

As usual, we get a decision problem by including a threshold t.

Linear Programming Relaxations

Each instance I of (D, Γ) can be turned into a linear program $\mathrm{BLP}(I)$:
Set of variables V, domain D, constraints $c = (x, R)$

$$\max \sum_{c \in C} \sum_{d \in R^{\Gamma}} \lambda_{c,d} \quad \text{where } c = (x, R), \text{ s.t.}$$

$$\sum_{d \in D^{|x|}; d_i = a} \lambda_{c,d} = \mu_{x_i, a} \qquad \forall c \in C, a \in D, i \in [|x|]$$

$$\sum_{a \in D} \mu_{v,a} = 1 \qquad \forall v \in V$$

Lift and Project Hierarchies

Given a *polytope* $\mathcal{K}$ for *integer* optimization problem, we can get a better approximation of the *convex hull* of the integer points by means of *lift-and-project* programs.

The general idea is to add new variables $y_{x_1,\ldots,x_t}$ to denote the product $x_1 \cdots x_t$ and add linear (or semi-definite) constraints to try and force this meaning.

We get hierarchies as t increases:

- *Sherali-Adams*: $\mathsf{SA}_t(\mathcal{K})$
- *Lovasz-Schrijver*: $\mathsf{LS}_t(\mathcal{K})$
- *Lasserre*: $\mathsf{Las}_t(\mathcal{K})$

Of these, the last is the strongest.

Dichotomy

For each Γ and t, there is an FPC interpretation that takes an instance I of $\mathrm{CSP}(\Gamma)$ to the tth level of the Lasserre hierarchy over $\mathrm{BLP}(I)$.

For every finite-valued CSP (D, Γ), the counting width of the corresponding decision problem is either $O(1)$ or $\Omega(n)$.

In the former case the problem can be solved exactly by its *linear programming relaxation*.

Every finite-valued CSP (D, Γ) that is *not* solved by its basic linear programming relaxation is not solved exactly by $o(n)$-level Lasserre lift-and-project.

Symmetry Gap in Constraint Optimization

Gaussian elimination is *not* a symmetric algorithm.

And, it cannot be made symmetric—this is established by the *XOR-Sat* lower bounds.

The ellipsoid method can be made symmetric.

Lower bounds on combinatorial optimization methods can be derived from this gap.

References

[1] Anderson, M., and Dawar, A. On symmetric circuits and fixed-point logics. *Theory Comput. Syst.* (2017).
[2] Anderson, M., Dawar, A., and Holm, B. Solving linear programs without breaking abstractions. *J. ACM* (2015).
[3] Dawar, A. The nature and power of fixed-point logic with counting. *ACM SIGLOG News* (2015).
[4] Dawar, A. On symmetric and choiceless computation. *TTCS* (2015).
[5] Dawar, A., and Wang, P. A definability dichotomy for finite valued CSPs. *CSL* (2015).
[6] Dawar, A., and Wang, P. Definability of semidefinite programming and Lasserre lower bounds for CSPs. *LICS* (2017).