Definable Ellipsoid Method, Sums-of-Squares Proofs, and the Isomorphism Problem

Albert Atserias, Joanna Ochremiak

LICS'18, Oxford
12th July 2018

Algorithm analysis through proof complexity

Objective: Identify tractable instances of hard problems.

Algorithms that compute:

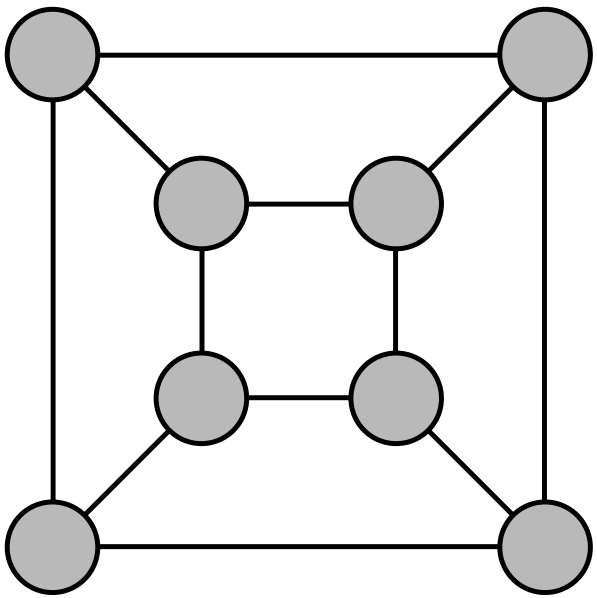
- an answer and
- a certificate/proof that the answer is correct.

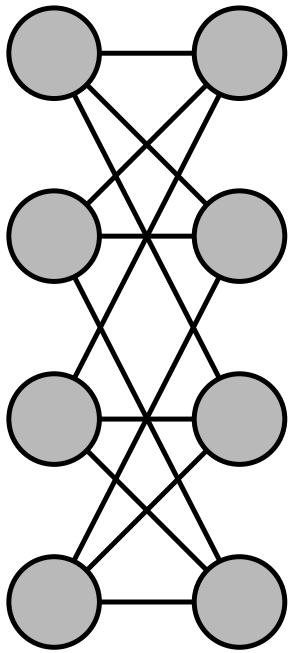
Approach: Study algorithms by analysing proof systems.

The graph isomorphism problem

Input: graphs G and H

Question: are G and H isomorphic?





Algebraic and mathematical-programming techniques

Step 1: encode an instance as a system of equations,

Step 2: solve the system.

Algebraic and mathematical-programming techniques

Step 1: encode an instance as a system of equations,

~~**Step 2:** solve the system.~~

Step 2: determine if there EXISTS a solution.

We only want to know if there EXISTS an isomorphism.

Step 1: equations

Input: graphs G and H

Compute: a system of equations $\mathrm{ISO}(G,H)$

$$\begin{cases} x_{vw}^2 - x_{vw} = 0 & \text{for every } v \in V(G), w \in V(H) \\ \sum_{w \in V(H)} x_{vw} - 1 = 0 & \text{for every } v \in V(G) \\ \sum_{v \in V(G)} x_{vw} - 1 = 0 & \text{for every } w \in V(H) \\ x_{vw}x_{v'w'} = 0 & \text{if } (v,v') \in E(G), (w,w') \notin E(H) \\ x_{vw}x_{v'w'} = 0 & \text{if } (v,v') \notin E(G), (w,w') \in E(H) \end{cases}$$

$$\text{SOLUTION} \iff \text{ISOMORPHISM}$$

Solving systems of polynomial equations is intractable.

Algebraic and mathematical-programming techniques

Step 1: encode an instance as a system of equations,

~~Step 2: solve the system.~~

~~Step 2: determine if there exists a solution.~~

Step 2: APPROXIMATELY determine if there exists a solution.

We can approximate using proof systems!

Step 2: computing a proof

Step 2: computing a proof that there is no solution

always correct

Output:

- if the algorithm finds a proof $\rightarrow$ *"no isomorphism"*
- otherwise $\rightarrow$ *"I do not know"*

Which pairs of non-isomorphic graphs the algorithms DISTINGUISH?

output *"no isomorphism"*

Proofs

Step 2: compute a proof that there is no solution

different **type of proof** $\leftrightarrow$ different algorithm

Algorithms:

- linear programming
- Gröbner basis
- semidefinite programming

Semidefinite Proofs

$$\begin{cases} x^2 + y + 2 = 0 \\ x - y^2 + 3 = 0 \end{cases}$$

$$-6\cdot(x^2 + y + 2) + 2\cdot(x - y^2 + 3) + \frac{1}{3} + 2\left(y + \frac{3}{2}\right)^2 + 6\left(x - \frac{1}{6}\right)^2 = -\mathbf{1}$$

Semidefinite Proofs

$$\begin{cases} x^2 + y + 2 = 0 \\ x - y^2 + 3 = 0 \end{cases}$$

A semidefinite proof that there is no solution:

$$-6\cdot(x^2 + y + 2) + 2\cdot(x - y^2 + 3) + \frac{1}{3} + 2\left(y + \frac{3}{2}\right)^2 + 6\left(x - \frac{1}{6}\right)^2 = -\mathbf{1}$$

Semidefinite Proofs

$$\begin{cases} x^2 + y + 2 = 0 \\ x - y^2 + 3 = 0 \end{cases}$$

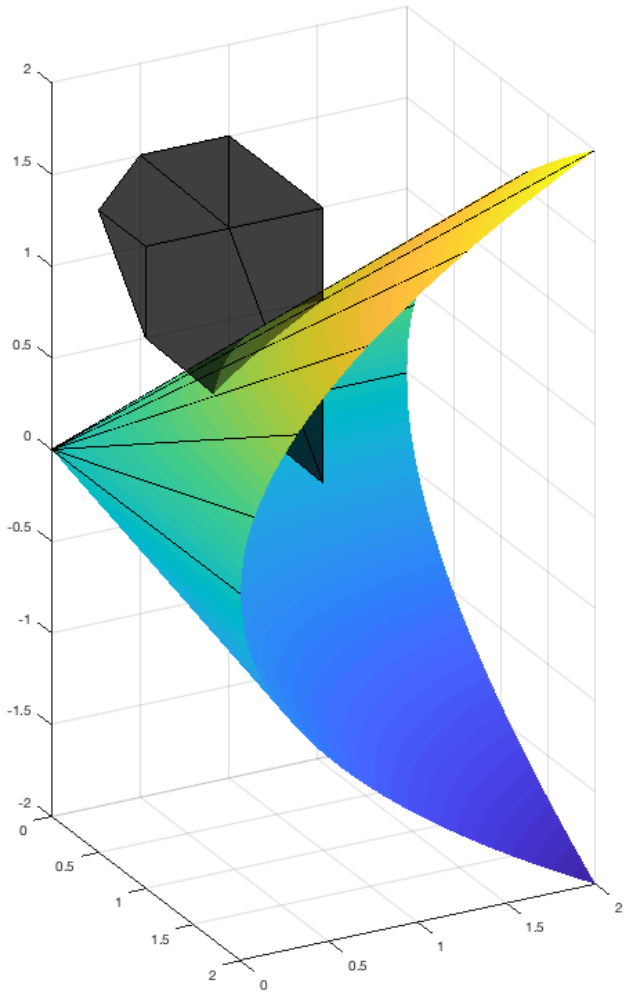
A semidefinite proof that there is no solution:

$$-6\cdot(x^2 + y + 2) + 2\cdot(x - y^2 + 3) + \frac{1}{3} + 2\left(y + \frac{3}{2}\right)^2 + 6\left(x - \frac{1}{6}\right)^2 = -\mathbf{1}$$

arbitrary polynomials sum of squares of polynomials

degree of the proof $\rightarrow$ max degree of polynomials on the left

Finding Semidefinite Proofs



Proofs

Step 2: compute a proof that there is no solution

restriction of semidefinite programming
finding a proof: linear inequalities

Algorithms:

- linear programming
- Gröbner basis
- semidefinite programming

computing a generating set
in the ideal of polynomials

Proofs

Step 2: compute a proof that there is no solution

Algorithms: Techniques:

- linear programming **hierarchy of algorithms**
- Gröbner basis **hierarchy of algorithms**
- semidefinite programming **hierarchy of algorithms**

degree of polynomials
in the proof

Summary of the setting

Algebraic and mathematical-programming techniques:

Step 1: encode an instance as a system of equations,

Step 2: compute a proof that there is no solution

always correct

Output:

- if the algorithm finds a proof → *"no isomorphism"*
- otherwise → *"I do not know"*

Which pairs of non-isomorphic graphs the algorithms DISTINGUISH?

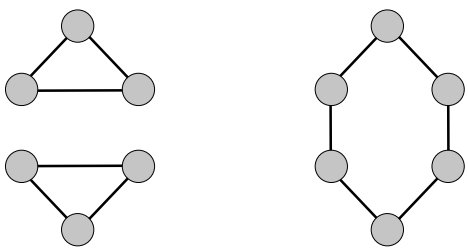
output *"no isomorphism"*

Known before

Theorem [Babai, Kučera'80]. Linear programming degree 2 distinguishes almost all graphs.

outputs *"I do not know"*

It does not distinguish:



Known before

For every pair of non-isomorphic graphs G and H:

Linear programming degree d distinguishes G and H.

$\Downarrow$ **[Berkholz, Grohe'15]**

Gröbner basis degree d distinguishes G and H.

$\Downarrow$ **[Berkholz'18]**

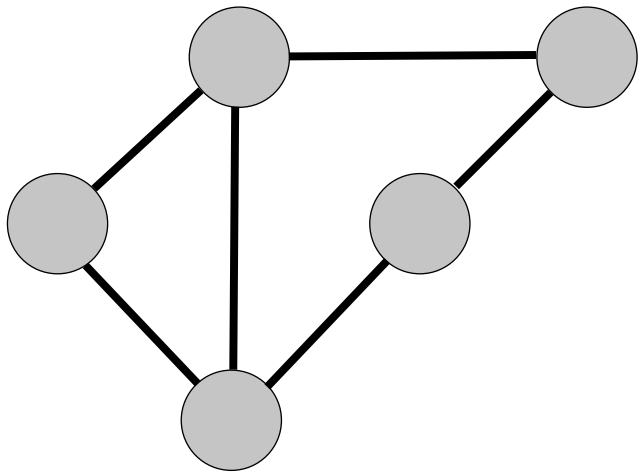
Semidefinite programming degree $2d$ distinguishes G and H.

Does semidefinite programming distinguish more graphs than linear programming?

Hope: yes!

Semidefinite programming much more powerful for many problems.

Example: MAX CUT

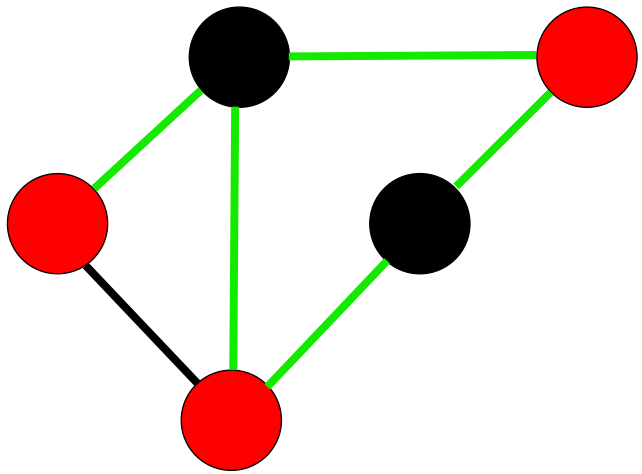


Semidefinite programming: best known efficient approximation
Linear programming: very bad approximation

Hope: yes!

Semidefinite programming much more powerful for many problems.

Example: MAX CUT



Semidefinite programming: best known efficient approximation
Linear programming: very bad approximation

All algorithms are equally powerful!

For every pair of non-isomorphic graphs G and H:

Linear programming degree d distinguishes G and H.

$\Downarrow$ **[Berkholz, Grohe'15]**

Gröbner basis degree d distinguishes G and H.

$\Downarrow$ **[Berkholz'18]**

Semidefinite programming degree $2d$ distinguishes G and H.

$\Downarrow$ **this work**

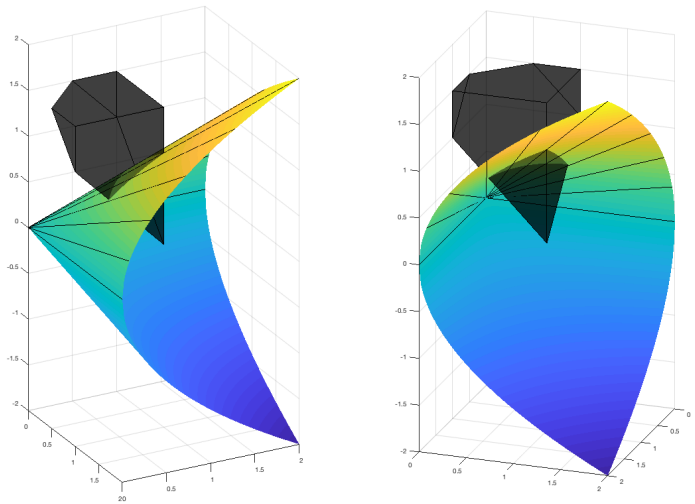
Linear programming degree cd distinguishes G and H.

constant independent from d

All algorithms are equally powerful!

Theorem. For the graph isomorphism problem all three algorithmic techniques are equally powerful, up to a constant factor loss in the degree.

Proof: Finding a semidefinite proof is expressible in counting logic.



Remark: For linear programming and Gröbner basis, showed independently in **[Grädel, Grohe, Pago, Pakusa'18]**.

Collapse

For every pair of non-isomorphic graphs G and H:

Semidefinite programming degree $2d$ distinguishes G and H.

$\Downarrow$

Counting logic $C^{cd}_{\infty\omega}$ distinguishes G and H.

$\Updownarrow$ **[Atserias, Maneva'13] [Malkin'14]**

Linear programming degree cd distinguishes G and H.

Finding semidefinite proofs in $C^{\omega}_{\infty\omega}$

Fact. Existence of semidefinite proofs reduces to feasibility of SDPs.

Is *polytop* $\cap$ *cone of positive semidefinite matrices* non-empty?

Finding semidefinite proofs in $C^{\omega}_{\infty\omega}$

Fact. Existence of semidefinite proofs reduces to feasibility of SDPs.

Is *polytop* $\cap$ *cone of positive semidefinite matrices* non-empty?

Theorem [Anderson, Dawar, Holm'15] [Dawar, Wang'17].
Approximate feasibility of bounded SDPs is expressible in FPC.

Finding semidefinite proofs in $C^{\omega}_{\infty\omega}$

Fact. Existence of semidefinite proofs reduces to feasibility of SDPs.

Is *polytop* $\cap$ *cone of positive semidefinite matrices* non-empty?

Theorem [Anderson, Dawar, Holm'15] [Dawar, Wang'17].
Approximate feasibility of bounded SDPs is expressible in FPC.

Key: The ellipsoid method for SDPs is expressible in FPC.

Finding semidefinite proofs in $C^{\omega}_{\infty\omega}$

Fact. Existence of semidefinite proofs reduces to feasibility of SDPs.

Is *polytop* $\cap$ *cone of positive semidefinite matrices* non-empty?

Theorem [Anderson, Dawar, Holm'15] [Dawar, Wang'17].
Approximate feasibility of bounded SDPs is expressible in FPC.

Key: The ellipsoid method for SDPs is expressible in FPC.

Theorem. The ellipsoid method for arbitrary class of bounded convex sets is expressible in FPC.

Finding semidefinite proofs in $C^{\omega}_{\infty\omega}$

Fact. Existence of semidefinite proofs reduces to feasibility of SDPs.

Is *polytop* $\cap$ *cone of positive semidefinite matrices* non-empty?

Theorem [Anderson, Dawar, Holm'15] [Dawar, Wang'17].
Approximate feasibility of bounded SDPs is expressible in FPC.

Key: The ellipsoid method for SDPs is expressible in FPC.

Theorem. The ellipsoid method for arbitrary class of bounded convex sets is expressible in FPC.

Theorem. Feasibility of SDPs is expressible in $C^{\omega}_{\infty\omega}$.