

Descriptive Complexity of Optimisation Problems

James Gate

Algorithms and Complexity Research Group (ACiD)
Department of Computer Science
Durham University

Bristol Algorithms Day 2009

Outline of Talk

- 1 Preliminaries
 - Vocabulary, Structures and Orderings
 - Representing Graphs
 - Logics and Satisfying Structures

Outline of Talk

- 1 Preliminaries
 - Vocabulary, Structures and Orderings
 - Representing Graphs
 - Logics and Satisfying Structures
- 2 Decision Problems
 - Capturing Problems and Complexity Classes
 - Key Results
 - An Example: Reachability in FO(IFP)

Outline of Talk

- 1 Preliminaries
 - Vocabulary, Structures and Orderings
 - Representing Graphs
 - Logics and Satisfying Structures
- 2 Decision Problems
 - Capturing Problems and Complexity Classes
 - Key Results
 - An Example: Reachability in FO(IFP)
- 3 Optimisation Problems
 - Formal Definitions
 - Classes
 - Logical Characterisation

Vocabulary

In finite model theory, every structure \mathcal{A} is of a particular vocabulary (or signature) σ . This vocabulary defines all the relation and constant symbols that are in \mathcal{A} .

Definition

A *vocabulary* σ is a sequence of relation and constant symbols

$$\sigma = \langle R_1^{a_1}, \dots, R_p^{a_p}, c_1, \dots, c_q \rangle$$

where each of the p relation symbols R_i are of arity a_i and there are q constant symbols.

Universes and Orderings

Universe

Every structure \mathcal{A} has a universe A .

- The cardinality of the universe $|A|$ corresponds to the 'size' of the structure.

Ordering

A structure is *ordered* if the elements of its universe are ordered. This ordering is represented as:

- A successor relation *succ*, such that $\forall x \forall y [succ(x, y) \Rightarrow (x < y \wedge \neg \exists z (x < z \wedge z < y))]$.
- Two special constant symbols *min* and *max*, which respectively refer to the first and last members of the universe.

Structures

- A structure \mathcal{A} is an assignment of symbols in a vocabulary σ to elements of a universe A .
- This assignment is denoted by the superscript on each symbol:
 - Each relation symbol $R_i^{\mathcal{A}} \subseteq A^{a_i}$, where a_i is the arity of R_i .
 - Each constant symbol $c_j^{\mathcal{A}} \in A$.
- Where the particular structure is implied, the superscript is omitted.

Graphs

Definition

The vocabulary of graphs $\sigma_G = \langle E^2 \rangle$.

In a structure representing a graph:

- The vocabulary is σ_G .
- The set of vertices is equal to the universe.
- The binary relation $E \in \sigma_G$ is the edge relation.

This can be extended to graphs with special vertices. $\langle E^2, s, t \rangle$ is the vocabulary of graphs with two special vertices s and t .

Logic

- A formula in some logic is *appropriate* to a structure if it shares the same vocabulary.
- A *sentence* is a formula of a particular vocabulary that only has free variables amongst the symbols from the vocabulary.
- To evaluate if a formula is *satisfied* by an appropriate structure:
 - Replace all occurrences of constant symbols in the formula with the appropriate values from the structure.
 - When the formula queries a relation, evaluate it in the structure, e.g. $E(x, y) \Leftrightarrow (x, y) \in E^A$.
- If a structure \mathcal{A} satisfies a sentence φ we write $\mathcal{A} \models \varphi$.

Representing Problems using Logic

Definition

A sentence φ represents the question posed in a decision problem Q iff:

- $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{A}$ is a “yes” instance to Q .
- $\mathcal{A} \not\models \varphi \Leftrightarrow \mathcal{A}$ is a “no” instance to Q .

where \mathcal{A} is any valid input structure of the problem Q

Capturing Complexity Classes

In Descriptive Complexity, we are interested in the relationships between logics and complexity classes.

Definition

For some logic \mathcal{L} and some complexity class K , the statement $\mathcal{L} = K$ holds if, every problem $Q \in K$ can be represented by some sentence φ in \mathcal{L} and if, any sentence in \mathcal{L} corresponds to a problem in K .

Key Results for Decision Problems

Key Results for Decision Problems

- $NP = SO\exists$ (second-order existential logic).

Key Results for Decision Problems

- $NP = SO\exists$ (second-order existential logic).
- $co-NP = SO\forall$ (second-order universal logic).

Key Results for Decision Problems

- Polynomial time hierarchy = SO (second-order logic).
- NP = $SO\exists$ (second-order existential logic).
- co-NP = $SO\forall$ (second-order universal logic).

Key Results for Decision Problems

- Polynomial time hierarchy = SO (second-order logic).
- NP = $SO\exists$ (second-order existential logic).
- co-NP = $SO\forall$ (second-order universal logic).
- P = FO(LFP) (first-order logic + least fixed-point operators).

Key Results for Decision Problems

- Polynomial time hierarchy = SO (second-order logic).
- $NP = SO\exists$ (second-order existential logic).
- $co-NP = SO\forall$ (second-order universal logic).
- $P = FO(LFP)$ (first-order logic + least fixed-point operators).
- $P = FO(IFP)$ (first-order logic + inflationary fixed-point operators).

Key Results for Decision Problems

- Polynomial time hierarchy = SO (second-order logic).
- $NP = SO\exists$ (second-order existential logic).
- $co-NP = SO\forall$ (second-order universal logic).
- $P = FO(LFP)$ (first-order logic + least fixed-point operators).
- $P = FO(IFP)$ (first-order logic + inflationary fixed-point operators).
- $P = SO\text{-Horn}$ (second-order logic with the first-order component restricted to Horn Clauses).

Key Results for Decision Problems

- Polynomial time hierarchy = SO (second-order logic).
- NP = $SO\exists$ (second-order existential logic).
- co-NP = $SO\forall$ (second-order universal logic).
- P = FO(LFP) (first-order logic + least fixed-point operators).
- P = FO(IFP) (first-order logic + inflationary fixed-point operators).
- P = SO-Horn (second-order logic with the first-order component restricted to Horn Clauses).

Note

All logics that (so far) capture P require the structure's universe to be ordered.

Defining and Applying an Inflationary Fixed Point Operator

Defining and Applying an Inflationary Fixed Point Operator

Definition

Given a first-order formula $\varphi(R, \bar{x})$, where R is a k -ary relation symbol and \bar{x} is a k -tuple, an IFP operator that constructs the relation R over some structure \mathcal{A} is defined as follows:

- $R^0 = \emptyset$.
- $R^{i+1} = R^i \cup \{\bar{x} \mid (\mathcal{A}, \bar{x}) \models \varphi(R^i, \bar{x})\}$.
- If $R^{i+1} = R^i$ then stop and return R^i (this is the *fixed-point*).

Reachability in FO(IFP)

Reachability

- Input instance: A (directed) graph G and two named vertices, s and t .
- Question: Is there a path from s to t in G ?

Reachability in FO(IFP)

Reachability

- Input instance: A (directed) graph G and two named vertices, s and t .
- Question: Is there a path from s to t in G ?

Example (Reachability in FO(IFP))

- $\varphi(R, x) := (x = s) \vee \exists y(R(y) \wedge E(y, x))$
- $\mathcal{A} \models [IFP_{R,x}\varphi](t)$

Optimisation Problems

- Decision problems ask *if* a structure has a particular property. A witness, or solution is a sub-structure that exhibits this property.
- Optimisation problems ask *what* the cost is of the best solution that exhibits a particular property.

Optimisation Problems

- Decision problems ask *if* a structure has a particular property. A witness, or solution is a sub-structure that exhibits this property.
- Optimisation problems ask *what* the cost is of the best solution that exhibits a particular property.

Decision Variants of Optimisation Problems

- Reachability asks "is there a path from s to t ?"
- k -Shortest Path asks "is there a path of length $\leq k$ between s and t ?"
- Shortest Path asks "what is the length of the shortest path between s and t ?"

Definition of an Optimisation Problem I

Definition

An optimisation problem Q is given by the following:

- \mathcal{I}_Q is the class of valid input structures.
- $\mathcal{F}_Q(\mathcal{A})$ is a function that maps an input structure $\mathcal{A} \in \mathcal{I}_Q$ to a set of feasible solutions.
- $cost_Q(\mathcal{A}, S)$ is a function that maps an input structure and a feasible solution S , to a numeric value.
- $\mu \in \{max, min\}$.
- The optimal value of a solution to Q is given by:

$$opt_Q(\mathcal{A}) = \mu\{cost_Q(\mathcal{A}, S) \mid S \in \mathcal{F}_Q(\mathcal{A})\}$$

Definition of an Optimisation Problem II

There are restrictions on some of the items that make up an optimisation problem.

Computational Restrictions

- \mathcal{I}_Q must contain only structures that can be recognised in polynomial time.
- $cost(\mathcal{A}, S)$ must be computable in polynomial time.

Absence of Feasible Solutions

- If $S \notin \mathcal{F}_Q(\mathcal{A})$ then $cost_Q(\mathcal{A}, S) = \perp$.
- $max\{\} = \perp$ and $min\{\} = \perp$.

Decision Variants of Optimisation Problems

Every optimisation problem \mathcal{Q} has a decision variant \mathcal{Q}_D . This is constructed by:

- Adding the constant c , which represents the *target cost*, to the input structure \mathcal{A} .
- Asking (for a maximisation problem) “is there a feasible solution S such that $cost(\mathcal{A}, S) \geq c$?” (or \leq for minimisation problems).

Decision Variants of Optimisation Problems

Every optimisation problem \mathcal{Q} has a decision variant \mathcal{Q}_D . This is constructed by:

- Adding the constant c , which represents the *target cost*, to the input structure \mathcal{A} .
- Asking (for a maximisation problem) “is there a feasible solution S such that $cost(\mathcal{A}, S) \geq c$?” (or \leq for minimisation problems).

Classification of Optimisation Problems

Given an optimisation problem \mathcal{Q} and its decision variant \mathcal{Q}_D :

- $\mathcal{Q} \in \text{NP}_{opt} \Leftrightarrow \mathcal{Q}_D \in \text{NP}$.
- $\mathcal{Q} \in \text{P}_{opt} \Leftrightarrow \mathcal{Q}_D \in \text{P}$.

Polynomially Bounded Optimisation Problems

Definition

An optimisation problem \mathcal{Q} is polynomially bounded, if and only if there exists some polynomial p such that for all input structures $\mathcal{A} \in \mathcal{I}_{\mathcal{Q}}$ and for all feasible solutions $S \in \mathcal{F}_{\mathcal{Q}}(\mathcal{A})$:

$$\text{cost}_{\mathcal{Q}}(\mathcal{A}, S) \leq p(|\mathcal{A}|)$$

Polynomially Bounded Optimisation Problems

Definition

An optimisation problem Q is polynomially bounded, if and only if there exists some polynomial p such that for all input structures $\mathcal{A} \in \mathcal{I}_Q$ and for all feasible solutions $S \in \mathcal{F}_Q(\mathcal{A})$:

$$\text{cost}_Q(\mathcal{A}, S) \leq p(|\mathcal{A}|)$$

Classes of Polynomially Bounded Optimisation Problems

- $\text{NP}_{opt}^{PB} \subset \text{NP}_{opt}$.
- $\text{P}_{opt}^{PB} \subset \text{P}_{opt}$.

Previous Results for NP_{opt}

- Kolaitis and Thakur developed a framework that captures problems in NP_{max}^{PB} [KT94].
- They refined it to also capture minimisation problems and hence all of NP_{opt}^{PB} [KT95].
- Zimand extended their framework to arbitrary objective functions, capturing NP_{opt} [Zim98].

Previous Results for P_{opt}

- Manyem and Bueno modified Kolaitis and Thakur's NP-optimisation framework to capture P_{opt}^{PB} by restricting it to Horn clauses [BM08, Man08].

Previous Results for P_{opt}

- Manyem and Bueno modified Kolaitis and Thakur's NP-optimisation framework to capture P_{opt}^{PB} by restricting it to Horn clauses [BM08, Man08].
- There are some problems with this approach:
 - Solving the framework for maximisation problems is NP-Hard (via an NP-completeness proof).
 - Tuple counting frameworks (of which this is one) are inappropriate for minimisation problems [KT95].
 - The difference between a class and its complement is the same as the difference between maximisation and minimisation problems [KT95]. Since $P = \text{co-P}$ it makes sense to treat maximisation and minimisation problems the same.

Previous Results for P_{opt}

- Manyem and Bueno modified Kolaitis and Thakur's NP-optimisation framework to capture P_{opt}^{PB} by restricting it to Horn clauses [BM08, Man08].
- There are some problems with this approach:
 - Solving the framework for maximisation problems is NP-Hard (via an NP-completeness proof).
 - Tuple counting frameworks (of which this is one) are inappropriate for minimisation problems [KT95].
 - The difference between a class and its complement is the same as the difference between maximisation and minimisation problems [KT95]. Since $P = \text{co-P}$ it makes sense to treat maximisation and minimisation problems the same.
- SO-Horn has several problems: FO(IFP) is an alternative.

Shortest Path

Start from the example of Reachability in FO(IFP) and add a termination clause.

Example (Shortest Path)

$$\psi(R, x) := [(x = s) \vee \exists y(R(y) \wedge E(y, x))] \wedge \neg R(t)$$

Shortest Path

Start from the example of Reachability in FO(IFP) and add a termination clause.

Example (Shortest Path)

$$\psi(R, x) := [(x = s) \vee \exists y(R(y) \wedge E(y, x))] \wedge \neg R(t)$$

Observations

- The optimal value equals the *inductive depth* of the FP operator generated by ψ , *minus 1*.
- We need to take into account optimal values of zero (e.g. $s = t$).
- Undefined values are different from an optimal value of zero.





A Fixed Point Framework for P_{opt}^{PB}

Theorem

An problem \mathcal{Q} is a polynomially bounded optimisation problem (i.e. $\mathcal{Q} \in P_{opt}^{PB}$) iff there exists some formula φ over σ , the vocabulary of \mathcal{Q} , such that for any $\mathcal{A} \in \mathcal{I}_{\mathcal{Q}}$, the optimal value $opt_{\mathcal{Q}}(\mathcal{A})$ is given by:

$$opt_{\mathcal{Q}}(\mathcal{A}) = depth([IFP_{R, \bar{x}}\varphi](\bar{t}))$$

where φ is a formula in the logic $FO(IFP)$ with free variables consisting only of the k -ary relation symbol $R \notin \sigma$ and the k -tuple \bar{x} . The k -tuple \bar{t} consists of constants from σ .

-  Orestes Bueno and Prabhu Manyem, *Polynomial-time maximisation classes: Syntactic hierarchy*, *Fundamenta Informaticae* **84** (2008), no. 1, 111–133.
-  Phokion G. Kolaitis and Madhukar N. Thakur, *Logical definability of NP optimization problems*, *Information and Computation* **115** (1994), no. 2, 321–353.
-  _____, *Approximation properties of NP minimization classes*, *Journal of Computer and System Sciences* **50** (1995), no. 3, 391–411.
-  Prabhu Manyem, *Syntactic characterizations of polynomial time optimization classes*, *Chicago Journal of Theoretical Computer Science* **2008** (2008), no. 3, 1–23.



Marius Zimand, *Weighted NP optimization problems: Logical definability and approximation properties*, SIAM Journal on Computing **28** (1998), no. 1, 36–56.