FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Knowledge-Based
Systems Group

Fixed-parameter tractable reductions to SAT

Ronald de Haan   Stefan Szeider

Vienna University of Technology

Reductions to SAT

- Problems in NP can be encoded into SAT in poly-time.
- Problems at the second level of the PH or higher cannot be encoded into SAT in poly-time (unless the PH collapses).
- **This talk**: fixed-parameter tractable (fpt) reductions as a way to get efficient SAT encodings for problems beyond NP.

Main point of this talk

1) Introduce fpt-reductions to SAT as a notion of tractability.
   - Analyze in what cases problems allow this.

2) Explain why such a strange complexity analysis can be useful.

3) Illustrate with some results (related to Boolean satisfiability).

Preliminaries: fpt-reductions

- Distinguish a parameter k in addition to input size n.
  - Parameter captures structure in input (k smaller $\sim$ more structure).
- Fpt-algorithm: runs in time $f(k) \cdot n^c$, for some computable function f and some constant c (fpt-time).
- Fpt-reduction: maps an instance (x, k) of problem P_1 to the instance (x', k') of problem P_2, such that:
  - $(x, k) \in P_1$ if and only if $(x', k') \in P_2$;
  - (x', k') is computed in fpt-time;
  - $k' \leq g(k)$.

  where g is a fixed computable function.
- Main idea: running time is reasonable for small values of k.

Illustrating example

- Example: QBF-SAT
- PSPACE-complete in general (so much harder than SAT).
- Now take instances with only few universal variables:
  - these are structured instances
  - parameter k: # of universal variables
  - apply quantifier expansion k many times
  - you get a SAT instance with blow-up (at most) 2^k
  - fpt-reduction to SAT

Why fpt-reductions to SAT?

- Best of two worlds: allow algorithms that use both structure in the input and practical performance of SAT solvers.
- Confront problems at second level of PH or higher (e.g., Σ_2^P).
  - Poly-time reductions to SAT not possible.
- Solve them with reasonable running time, for small values of the parameter k.

? Why not just use fixed-parameter tractability?
  - Parameters can be much less restrictive,
  - i.e., larger classes of instances are 'tractable.'

Various notions of fpt-reductions

- Many-to-one reductions (as before).
- Turing reductions:
  - fpt-algorithms that can query a SAT oracle:
    - **$\mathbf{f(k)}$ many times**;
    - $f(k) \cdot \log n$ many times; or
    - $f(k) \cdot n^c$ many times.

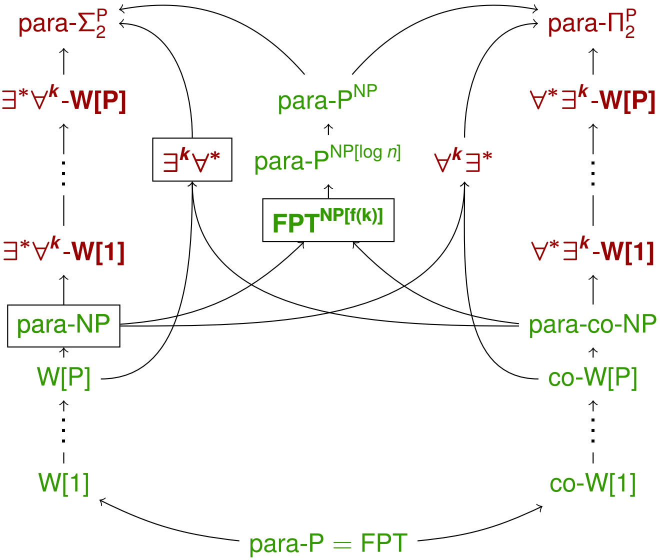
  where f is some fixed computable function.

  - (# SAT calls not the only important factor in practice)

Theoretical tools

- Existing tools:
  - para-NP: all parameterized problems many-to-one fpt-reducible to SAT
  - para-Σ_2^P: even Σ_2^P-hard for constant parameter value
- Recently developed/considered tools:
  - $FPT^{NP[f(k)]}$: all parameterized problems Turing fpt-reducible to SAT
  - $\exists^k\forall^*$: evidence against fpt-reducibility to SAT (but poly-time reducible to SAT for constant parameter value)

Theoretical tools: a picture



Minimizing implicants of DNF formulas

- An implicant of a formula φ is a set L of literals such that $\bigwedge L \models \varphi$.

Small DNF Implicant

Instance:	A DNF formula φ, an implicant L of φ of size n, and a positive integer m.
Question:	Is there an implicant $L' \subseteq L$ of φ of size m?

Theorem

DNF Minimization parameterized by $k = (n - m)$ is $\exists^k\forall^*$-complete.

Theorem

DNF Minimization parameterized by $k = m$ is $\exists^k\forall^*$-complete.

Minimizing DNF formulas

DNF Minimization

Instance:	A DNF formula of size n, and a positive integer m.
Question:	Is there a DNF formula φ' of size m such that $\varphi' \equiv \varphi$, that can be obtained from φ by deleting literals?

Theorem

DNF Minimization parameterized by $k = (n - m)$ is $\exists^k\forall^*$-complete.

Minimizing DNF formulas

Theorem

DNF Minimization parameterized by $k = m$ can be solved in fpt-time using $\lceil \log_2 k \rceil + 1$ many SAT calls.

- Algorithm (idea):
  - Identify "relevant" variables, using binary search ($\lceil \log_2 k \rceil$ many SAT calls).
  - Enumerate all possible DNF formulas of size $\leq k$ over these variables, and check if at least one of them is equivalent to φ (1 SAT call).

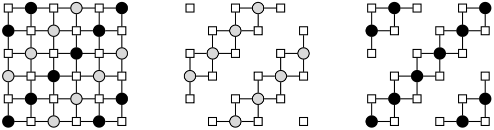
2QBF with bounded existential or universal treewidth

- Consider $\exists X.\forall Y.\psi$, where ψ is in DNF.
  Problem: is this formula true? (Σ_2^P-complete)

- For a DNF formula $\psi = \delta_1 \vee \cdots \vee \delta_m$ and a subset Z of its variables, consider the incidence graph of ψ w.r.t. Z:

$$\begin{aligned} \mathrm{IG}(\psi, Z) &= (V, E); \\ V &= Z \cup \{\delta_1, \ldots, \delta_m\}; \text{ and} \\ \{\delta_i, z\} \in E &\text{ iff } z \text{ occurs in } \delta_i. \end{aligned}$$

- Incidence treewidth w.r.t. to X or Y can be much smaller (than w.r.t. Z):



(wrt Z) (wrt X) (wrt Y)

2QBF with bounded existential treewidth

Theorem

$\exists\forall$-QBF-SAT(DNF) parameterized by the incidence treewidth w.r.t. the existential variables is para-Σ_2^P-complete.

- In other words: this kind of structure does not help at all.
- Idea: replace each existential variable x by a fresh universal variable y, and make sure they get the same value.

2QBF with bounded universal treewidth

Theorem

$\exists\forall$-QBF-SAT(DNF) parameterized by the incidence treewidth w.r.t. the universal variables is para-NP-complete.

- In other words: an fpt-reduction to SAT.
- Idea: encode dynamic programming algorithm to handle the assignment to the universal variables by means of a SAT instance.

Take home message

- Introduced fpt-reductions to SAT as a notion of tractability.
  - Discussed tools for corresponding complexity analysis.
- Explained that this analysis can be useful for developing algorithms for problems higher in the PH.
- Illustrated by analyzing some problems.
  - Minimizing implicants of DNF formulas
  - Minimizing DNF formulas
  - 2QBF with bounded existential or universal treewidth