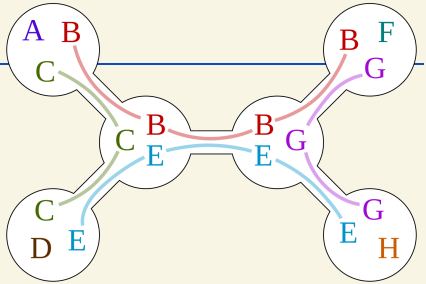
Parameterized Complexity Theory I

DM898: Parameterized Algorithms
Lars Rohwedder



Today's lecture

- Parameterized reductions
- Exponential Time Hypothesis (ETH)
- No FPT algorithm for Clique problem under ETH

Classical complexity theory

Impossibility results

For some problems we have good algorithms, for others we do not (yet). We would like to prove (conditional) impossibility results that show certain algorithmic problems cannot be solved efficiently. This would be an important insight in itself and also can help guide the research effort towards achievable goals.

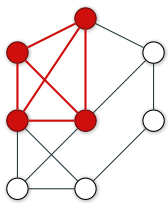
The backbone of classical impossibility results within NP is two-fold:

- We can construct **reductions** from a problem A to a problem B, which show that if B can be solved in polynomial time then also A can
- We can prove for some problem (3-SAT) that under a believable **hypothesis** (P$\neq$NP) it cannot be solved in polynomial time

This gives us a tool for conditional proofs that all kinds of problems do not have efficient (polynomial time) algorithms.

Parameterized complexity

There are also fundamental problems for which we have not discovered FPT algorithms, for example, the **Clique** problem (parameterized by clique size)



This problem (and many other parameterized problems) has been studied exhaustively. Can we find also formal evidence that they cannot be solved efficiently (here: in **FPT time**)?

Parameterized reductions

The concept of reductions is easy to transfer to FPT:

Parameterized reduction

Let A, B be two parameterized problems. A parameterized reduction is an algorithm that takes an instance (I, k) of problem A and outputs an instance (I', k') of problem B such that:

- (I, k) is a YES-instance if and only if (I', k') is a YES-instance
- $k' \leq g(k)$ for some computable function g
- The running time is $f(k)n^{O(1)}$ for some computable function f

Assume there is a reduction from A to B, then: if B has an FPT algorithm, so does A. Conversely, if we have evidence that A has no FPT algorithm, then also for B.

Parameterized reductions

The concept of reductions is easy to transfer to FPT:

Parameterized reduction

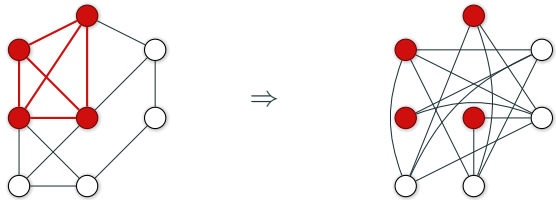
Let A, B be two parameterized problems. A parameterized reduction is an algorithm that takes an instance (I, k) of problem A and outputs an instance (I', k') of problem B such that:

- (I, k) is a YES-instance if and only if (I', k') is a YES-instance
- $k' \leq g(k)$ for some computable function g
- The running time is $f(k)n^{O(1)}$ for some computable function f

Assume there is a reduction from A to B, then: if B has an FPT algorithm, so does A. Conversely, if we have evidence that A has no FPT algorithm, then also for B.

Example

The standard reduction from Clique to Independent Set, by complementing the graph, preserves the parameter (clique size / independent set size), hence is a parameterized reduction



Parameterized reductions

The concept of reductions is easy to transfer to FPT:

Parameterized reduction

Let A, B be two parameterized problems. A parameterized reduction is an algorithm that takes an instance (I, k) of problem A and outputs an instance (I', k') of problem B such that:

- (I, k) is a YES-instance if and only if (I', k') is a YES-instance
- $k' \leq g(k)$ for some computable function g
- The running time is $f(k)n^{O(1)}$ for some computable function f

Assume there is a reduction from A to B, then: if B has an FPT algorithm, so does A. Conversely, if we have evidence that A has no FPT algorithm, then also for B.

Example 2

The standard reduction from Independent Set to Vertex Cover by complementing the solution ($V \setminus U$ is independent set if and only if U is vertex cover) is **not** a parameterized reduction since it increases parameter k to $n - k$

Hypothesis for parameterized complexity

Clique is NP-hard. Thus, assuming P≠NP there is no polynomial time algorithm for it.

Hypothesis P≠NP

Equivalent: There is no $n^{O(1)}$ time algorithm for 3-SAT with n variables.[1]
(P≠NP is famously unproven, but widely believed to be true)

P≠NP seems insufficient to show that Clique has no FPT algorithm.

[1] the number of clauses m in a 3-SAT is at most $O(n^3)$. Therefore we can omit it in this running time.

Hypothesis for parameterized complexity

Clique is NP-hard. Thus, assuming P≠NP there is no polynomial time algorithm for it.

Hypothesis P≠NP

Equivalent: There is no $n^{O(1)}$ time algorithm for 3-SAT with n variables.[1]
(P≠NP is famously unproven, but widely believed to be true)

P≠NP seems insufficient to show that Clique has no FPT algorithm.

The best algorithm for 3-SAT is not even close to $n^{O(1)}$. We have seen an $O(2^{0.8n})$ time algorithm, slightly improving over the trivial $O(2^n)$. Further research has led to $O(2^{0.387n})$.

It seems these ideas are also limited and perhaps we cannot achieve a sublinear exponent, e.g. $O(2^{\sqrt{n}})$ or even $O(2^{n/\log(n)})$.

[1] the number of clauses m in a 3-SAT is at most $O(n^3)$. Therefore we can omit it in this running time.

Hypothesis for parameterized complexity

Clique is NP-hard. Thus, assuming P≠NP there is no polynomial time algorithm for it.

Hypothesis P≠NP

Equivalent: There is no $n^{O(1)}$ time algorithm for 3-SAT with n variables.[1]
(P≠NP is famously unproven, but widely believed to be true)

P≠NP seems insufficient to show that Clique has no FPT algorithm.

The best algorithm for 3-SAT is not even close to $n^{O(1)}$. We have seen an $O(2^{0.8n})$ time algorithm, slightly improving over the trivial $O(2^n)$. Further research has led to $O(2^{0.387n})$.

It seems these ideas are also limited and perhaps we cannot achieve a sublinear exponent, e.g. $O(2^{\sqrt{n}})$ or even $O(2^{n/\log(n)})$.

Exponential Time Hypothesis (ETH)

There exists some constant $\epsilon > 0$ such that there is no $O(2^{\epsilon n})$ time algorithm for 3-SAT.

The ETH is not proven either. In fact, would imply P≠NP and is even more controversal.

We will show that assuming ETH, there is **no FPT algorithm for Clique**.

---

[1] the number of clauses m in a 3-SAT is at most $O(n^3)$. Therefore we can omit it in this running time.

Sparsification Lemma

Theorem (Sparsification Lemma)

If the ETH is true, then there is some $\epsilon' > 0$ such that there is no $O(2^{\epsilon'(n+m)})$ time algorithm for 3-SAT.

- This is called Sparsification Lemma, because it means we can reduce a 3-SAT formular to (multiple) "sparse" 3-SAT formulars, i.e., with only linearly many clauses.
- The proof the Sparsification Lemma is difficult and we look at it it after first seeing consequences
- A more general form of the sparsification lemma for k-SAT (or k-CSP) with constant k can be proven as well. We omit this here.

No FPT for Clique

Theorem

If there was an FPT algorithm for Clique parameterized by clique size, then for every $\epsilon > 0$ there would be an $O(2^{\epsilon m})$ time algorithm for 3-SAT, which would refute the ETH via the Sparsification Lemma.

Proof: blackboard

Completeness

The classical P vs. NP framework has the nice feature of having **complete** problems: An NP-complete problem has no polynomial time algorithm **if and only if** P$\neq$NP.

Clique is **not** necessarily complete with respect to ETH: If ETH holds, then Clique has no FPT algorithm. But if Clique has no FPT algorithm, it does not necessarily imply that ETH holds.

The classical P vs. NP framework has the nice feature of having **complete** problems: An NP-complete problem has no polynomial time algorithm **if and only if** P≠NP.

Clique is **not** necessarily complete with respect to ETH: If ETH holds, then Clique has no FPT algorithm. But if Clique has no FPT algorithm, it does not necessarily imply that ETH holds.

To a complexity theoreticial, this may feel slightly unsatisfying...

W-hierarchy

Looking closer at reductions, parameterized problems fall into equivalence classes (for which they are complete), called W[1],W[2],... that form a hierarchy (problems in W[t] are reducible to problems in W[$t+1$])

We omit details on this hierarchy.