EXACT TSP

Problem **EXACT TSP**

**INSTANCE:** \( n \) cities \( 1, \ldots, n \), a non-negative integer distance \( d_{ij} \) between any two cities \( i \) and \( j \) (such that \( d_{ij} = d_{ji} \)), and an integer \( B \).

**QUESTION:** Is the length of the shortest tour equal to \( B \)?

**Complexity of EXACT TSP**

**EXACT TSP** can be considered as the intersection of two problems – one in NP and one in co-NP:

- in NP: \( \text{TSP(D)} \) (= asking if the shortest tour has length \( \leq B \)).
- in co-NP: \( \text{TSP COMPLEMENT} \) (= asking if the shortest tour has length \( \geq B \)).
**SAT-UNSAT**

**Problem SAT-UNSAT**

INSTANCE: two Boolean expressions \((\varphi, \varphi')\) (possibly both in 3-CNF).

QUESTION: Is it true that \(\varphi\) is satisfiable and \(\varphi'\) is unsatisfiable?

**Proposition**

SAT-UNSAT is \(DP\)-complete.

**Proof of Membership**

Let \(L_1 = \{ (\varphi, \psi) \mid \varphi\text{ satisfiable and } \psi\text{ arbitrary propositional formula} \}\)

Let \(L_2 = \{ (\varphi, \psi) \mid \varphi\text{ arbitrary propositional formula and } \psi\text{ unsatisfiable} \}\)

Clearly \(L_1 \in NP\), \(L_2 \in co-NP\), and SAT-UNSAT = \(L_1 \cap L_2\).

**Proof of Hardness**

Let \(L\) be an arbitrary language in \(DP\), i.e., there exists a language \(L_1 \in NP\) and a language \(L_2 \in co-NP\) with \(L = L_1 \cap L_2\).

Let \(x\) be an arbitrary instance of \(L\). We reduce \(x\) to the following instance \(R(x)\) of SAT-UNSAT:

\[ R(x) := (R_1(x), R_2(x)) \]

Clearly, \(R(x)\) is a positive instance of SAT-UNSAT \(\iff\) \(R_1(x)\) is satisfiable and \(R_2(x)\) is unsatisfiable \(\iff\) (by the correctness of \(R_1\) and \(R_2\)) \(x \in L_1\) and \(x \in L_2 \iff x \in L\).

**Further DP-complete problems**

“Critical Problems”

**CRITICAL SAT**

INSTANCE: Propositional formula \(\varphi\) in CNF

QUESTION: Is it true that \(\varphi\) is unsatisfiable but deleting any clause makes \(\varphi\) satisfiable?

**CRITICAL HAMILTON PATH**

INSTANCE: (Directed or undirected) graph \(G = (V, E)\)

QUESTION: Is it true that \(G\) has no Hamilton path but addition of any edge creates a Hamilton path?

**CRITICAL 3-COLORABILITY**

INSTANCE: Undirected graph \(G = (V, E)\)

QUESTION: Is it true that \(G\) has no 3-coloring but deletion of any node makes it 3-colorable?

**Proposition**

CRITICAL SAT, CRITICAL HAMILTON PATH, and CRITICAL 3-COLORABILITY are \(DP\)-complete.

**Remark**

The above problems are called “critical” because the input \(x\) is “critical” with respect to some property, i.e., \(x\) has some property but the slightest modification of \(x\) does not.
Oracle Machines

Motivation

Intuitively, an oracle is a subroutine with 0 cost (we count the cost of the oracle as 1 for the call – but we neglect the cost of the computation carried out by the oracle). ⇒ We can study complexity in a setting where a part of the computation comes “for free”. Oracles allow us to isolate orthogonal (independent) sources of complexity, i.e. we can answer questions like: Suppose that we know the complexity of some sub-task $A$ for solving problem $B$. What is the remaining complexity of problem $B$?

Oracle Machines

Definition

An oracle Turing machine $M'$ has the following additional features:

- an additional tape (= query tape)
- three additional states: query state $q_?$, answer states $q_{YES}$, $q_{NO}$

Suppose that $M'$ has an oracle for the problem $A$. Then the call of the oracle works as follows: If $M'$ is in state $q_?$, then $M'$ decides if the string $z$ on the query tape is a positive instance of $A$ or not. ⇒ $M'$ either enters state $q_{YES}$ or $q_{NO}$ in one step.

Notation. For any time complexity class $C$ and oracle $A$ (where $A$ is either a problem or a class of problems) we write $C^A$ for the problems which can be decided by a TM within the time bound of $C$, where the TM is allowed to use an oracle for (any problem in the class) $A$.

Examples. $P^{SAT}$, $NP^{SAT}$, $P^{NP}$, $NP^{NP}$, . . . ,

The Polynomial Hierarchy

Definition

The polynomial hierarchy is a sequence of classes:

- $\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$
- $i \geq 0$, $\Delta_{i+1}^P = P^{\Sigma_i^P}$
  $\Sigma_{i+1}^P = NP^{\Sigma_i^P}$
  $\Pi_{i+1}^P = co-NP^{\Sigma_i^P}$

Cumulative polynomial hierarchy: $PH = \bigcup_{i \geq 0} \Sigma_i^P$

In the literature also the following notation is used: $\Delta_i^P$, $\Sigma_i^P$, $\Pi_i^P$

Properties of the Polynomial Hierarchy

- special case $i = 1$:
  $\Delta_1^P = P^{\Sigma_0^P} = P^P = P$
  $\Sigma_1^P = NP^{\Sigma_0^P} = NP^P = NP$
  $\Pi_1^P = co-NP^{\Sigma_0^P} = co-NP$

- special case $i = 2$:
  $\Delta_2^P = P^{\Sigma_1^P} = P^{NP}$
  $\Sigma_2^P = NP^{\Sigma_1^P} = NP^{NP}$
  $\Pi_2^P = co-NP^{\Sigma_1^P} = co-NP^{NP}$

$\Delta_i^P \subseteq \Sigma_i^P \subseteq \Delta_{i+1}^P \subseteq \Sigma_{i+1}^P \subseteq \Pi_{i+1}^P \subseteq \Delta_{i+2}^P$
Characterization via Certificates

**Theorem**

- Let $L$ be a language and $i \geq 1$. Then $L \in \Sigma_i P$ iff there exists a polynomially balanced relation $R$ (i.e., there exists $k$, s.t. $(x, y) \in R$ implies $|y| \leq |x|^k$), such that the language $\{x | x \# y (x, y) \in R\}$ is in $\Pi_{i-1} P$ and
  \[ L = \{x | \text{there exists a } y \text{ with } |y| \leq |x|^k \text{ s.t. } (x, y) \in R\} \]

- Let $L$ be a language and $i \geq 1$. Then $L \in \Pi_i P$ iff there is a polynomially balanced relation $R$ such that the language $\{x \# y | (x, y) \in R\}$ is in $\Sigma_{i-1} P$ and
  \[ L = \{x | \text{for all } y \text{ with } |y| \leq |x|^k, (x, y) \in R\} \]

**Remark.** Of course, in the definition of $\Sigma_i P$, we could omit the condition $|y| \leq |x|^k$, since we talk about a polynomially balanced relation $R$.

**Proof (continued)**

"$\Rightarrow$" Suppose that $L \in \Sigma_i P$, i.e., $L$ is decided by a non-deterministic, polynomial-time TM $M$ with an oracle for some language $K \in \Sigma_{i-1} P$. We must show that an appropriate relation $R$ exists.

By the induction hypothesis, there exists a binary relation $S$, s.t. the language $\{u \# v | (u, v) \in S\}$ is in $\Pi_{i-2} P$ and $K = \{u | \text{there exists a } v \text{ with } |v| \leq |u|^k \text{ s.t. } (u, v) \in S\}$.

We construct a relation $R$ as follows: We know that $x \in L$ iff there exists an accepting computation of $M$ (with oracle for $K$) on $x$. We define $R$, s.t. $(x, y) \in R$, iff $y$ is a "certificate" of $x$ in the following sense:

1. $y$ encodes the non-deterministic choices of a successful computation of the TM $M$ as in the construction of succinct certificates for NP.
2. In addition, $y$ contains a certificate $v_j$ for every successful call $u_j$ to the oracle $K \in \Sigma_{i-1} P$.

"$\Leftarrow$" Suppose that such a relation $R$ exists. We must show that $L \in \Sigma_i P$. Indeed, $L$ is decided by the following non-deterministic, polynomial-time Turing machine with $\Sigma_{i-1} P$-oracle:

1. On input $x$, guess an appropriate $y$.
2. Check by means of a $\Pi_{i-1} P$ oracle if $(x, y) \in R$ (or, equivalently, check by a $\Sigma_{i-1} P$ oracle if $(x, y) \not\in R$).

**Proof (continued)**

It remains to show that $(x, y) \in R$ can indeed be decided in $\Pi_{i-1} P$:

1. We must check that the first part of $y$ indeed encodes a successful computation of the TM $M$. This can be done in polynomial time (as in the construction of succinct certificates for NP).
2. We must check for polynomially many pairs $(u_j, v_j)$ that $(u_j, v_j) \in S$ holds. Each such test is in $\Pi_{i-2} P$. Hence, in total, these tests are in $\Delta_{i-1} P \subseteq \Pi_{i-1} P$ (actually, they are even in $\Sigma_{i-1} P$).
3. For all "no"-queries $u_j$ to the $K$-oracle, we must check that indeed $u_j \not\in K$. Since $K \in \Sigma_{i-1} P$, each test $u_j \not\in K$ is in $\Pi_{i-1} P$.

All these tests together can be done by a single $\Pi_{i-1} P$ computation since the co-query "there exists a $j$ with $u_j \in K$" is in $\Sigma_{i-1} P$: Guess $j$ and $v_j$ and check in $\Pi_{i-2} P$ that $(u_j, v_j) \in S$. 
**Characterization via Certificates and Alternation**

**Definition**
A relation $R \subseteq (\Sigma^*)^{i+1}$ is said to be polynomially balanced if whenever $(x, y_1, \ldots, y_i) \in R$, it holds that $|y_1|, \ldots, |y_i| \leq |x|^k$ for some $k$.

**Corollary, part 1**
Let $L$ be a language and $i \geq 1$. Then $L \in \Sigma_i \cap P$ iff there is a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation $R$ such that

$$L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Qy_i \text{ such that } (x, y_1, \ldots, y_i) \in R \}$$

where $Q$ is $\forall$ if $i$ is even and $\exists$ if $i$ is odd.

**Proof idea**
Use the above theorem and proceed by induction on $i$. Repeatedly replace languages in $\Pi_i \cap P$ and $\Sigma_i \cap P$ by their certificate forms as in the theorem.

**Properties of PH**

**Definition**
We say that the polynomial hierarchy collapses to the $i$-th level if $\Delta_i \cap P = \Sigma_i \cap P = \Pi_i \cap P$ holds for every $j > i$.

**Remark**
It is unknown whether PH is indeed an infinite hierarchy, i.e.: $\Delta_0 \cap P \subset \Sigma_1 \cap P \subset \Sigma_2 \cap P \subset \ldots$ is generally believed but not known.

**Proposition**
- If for some $i \geq 1$, $\Sigma_i \cap P = \Pi_i \cap P$, then the polynomial hierarchy collapses to the $i$-th level. In particular, if $\Pi = \coNP$, then the polynomial hierarchy collapses to the first level.
- $P = \NP$ iff $P = \PH$.
- Notice that it can be the case that $P \neq \NP$ and $\NP \neq \coNP$ but $\PH$ collapses to the second level (not expected to happen, though).

**QBFs: Quantified Boolean Formulae**

**QSAT**
“quantified satisfiability with $i$ alternating blocks of quantifiers”:

INSTANCE: Boolean expression $\varphi$ with the Boolean variables partitioned into $i$ sets $X_1, \ldots, X_i$.

QUESTION: Is it true that there exists a partial truth assignment for the variables $X_1$ such that for all partial truth assignments for $X_2$ there exists a partial truth assignment for $X_3$ \ldots $\varphi$ is satisfied by the overall truth assignment?

**Notation**
A QSAT$_i$-formula is given in the form $\exists X_1 \forall X_2 \exists X_3 \cdots QX_i \varphi$, where $Q$ is $\forall$ if $i$ is even and $\exists$ if $i$ is odd.
### Σ_i P-Completeness

**Theorem**

For all $i \geq 1$, QSAT$_i$ is Σ$_i$P-complete.

**Proof of Σ$_i$P-membership**

Recall the characterization of Σ$_i$P via certificates: A language $L$ is in Σ$_i$P iff there is a polynomially balanced, polynomial-time decidable $(i+1)$-ary relation $R$, s.t. $L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \cdots Qy_i \text{ such that } (x, y_1, \ldots, y_i) \in R \}$.

For a QSAT$_i$-formula $\psi = \exists X_1 \forall X_2 \exists X_3 \cdots QX_i \varphi$, we take as certificates $(y_1, \ldots, y_i)$ combinations of variable assignments to the alternating blocks $X_1, X_2, \ldots, X_i$ of variables (where each $y_j$ is an assignment on the variables in $X_j$), s.t. the formula $\varphi$ is true in the overall assignment.

**Proof of Σ$_i$P-hardness (continued)**

Now let $x$ be an arbitrary instance of the decision problem corresponding to the language $L$. Moreover, let $\hat{X}$ denote the values of the variables in $X$ corresponding to the string $x$. By $\varphi(\hat{X})$ we denote the result of substituting in $\varphi$ the corresponding Boolean values $\hat{X}$ for $X$. We define the desired instance of QSAT$_i$ as $\psi = \exists Y_1 \forall Y_2 \cdots \exists Y_i \exists Z \varphi(\hat{X})$.

For the correctness proof, we observe: Let $y_1, \ldots, y_i$ be arbitrary strings with Boolean “encoding” $\hat{Y}_1, \ldots, \hat{Y}_i$ and suppose that we substitute these values $\hat{Y}_1, \ldots, \hat{Y}_i$ for the variables $Y_1, \ldots, Y_i$ in $\varphi$. Then the resulting expression $\varphi(\hat{X}, \hat{Y}_1, \ldots, \hat{Y}_i)$ is satisfiable (i.e., there exist an appropriate assignment to the variables in $Z$) iff $(x, y_1, \ldots, y_i) \in R$.

It remains to show that $x \in L \iff \psi$ is a positive instance of QSAT$_i$.

Indeed, $x \in L$ iff there is a $y_1$ s.t. for all $y_2, \ldots$ there is a $y_i$ s.t. $(x, y_1, \ldots, y_i) \in R$. In terms of $\psi$, this means that for these values of $\hat{X}$ there are values $\hat{Y}_1$ for $Y_1$ s.t. for all values $\hat{Y}_2$ for $Y_2, \ldots$ there are values $\hat{Y}_i$ for $Y_i$ and there are values $\hat{Z}$ for $Z$ s.t. the resulting formula $\varphi(\hat{X}, \hat{Y}_1, \ldots, \hat{Y}_i, \hat{Z})$ is true, i.e., $\psi$ is a positive instance of QSAT$_i$.

### Further Complete Problems

**Theorem**

For all $i \geq 1$ even, the QSAT$_i$ problem remains Σ$_i$P-complete even if the instances $\exists X_1 \forall X_2 \exists X_3 \cdots \exists X_i \varphi$ are restricted s.t. $\varphi$ is in 3-DNF.

For all $i \geq 1$ odd, the QSAT$_i$ problem remains Σ$_i$P-complete even if the instances $\exists X_1 \forall X_2 \exists X_3 \cdots \exists X_i \varphi$ are restricted s.t. $\varphi$ is in 3-CNF.

**Theorem**

**MINIMAL MODEL SAT**: Given a propositional formula $\varphi$ in CNF and an atom $x$, is $x$ true in some (subset) minimal model of $\varphi$?

**MINIMAL MODEL SAT is Σ$_2$P-complete.**
MINIMAL MODEL SAT

Proof of the $\Sigma_2^P$-membership

We have to show that MINIMAL MODEL SAT can be decided by an NP-algorithm using an NP-oracle (or, equivalently, a co-NP-oracle). Let $\psi$ be an arbitrary CNF-formula with variables in $X$.

1. Guess a truth assignment $I$, s.t. $x$ is true in $I$. Let $Y \subseteq X$ denote the variables which are true in $I$.
2. Check that $\psi$ is true in $I$.
3. Check (with an oracle) that there does not exist a “smaller” satisfying truth assignment $J$ of $\psi$, i.e., let $Z$ denote the variables true in $J$, then $Z \subseteq Y$ for any satisfying truth assignment $J$ of $\psi$.

The check in step 3 can be done by a co-NP-oracle, i.e.: checking that there does exist a “smaller” satisfying truth assignment $J$ of $\psi$ can be clearly done in NP.

MINIMAL MODEL SAT

Proof of the $\Sigma_2^P$-hardness (continued)

$\Rightarrow$ Suppose that $\psi = (\exists x_1, \ldots, x_k)(\forall y_1, \ldots, y_l)\varphi$ is true. Then there exists a partial assignment $I$ on $\{x_1, \ldots, x_k\}$, s.t. for any values assigned to $\{y_1, \ldots, y_l\}$, the formula $\varphi$ is true (or, equivalently, $\neg\varphi$ is false).

We define the truth assignment $J$ appropriate to $\chi$ as follows:

$J(x_i) = I(x_i)$ and $J(x'_i) = I(\neg x_i)$ for every $i$,

$J(y_j) = \text{true}$ for every $j$, and $J(z) = \text{true}$.

We claim that $J$ is a minimal model of $\chi$ where $z$ is true.

Clearly, $J$ is a model (i.e., satisfying truth assignment) of $\chi$ since all conjuncts ($\neg x_i \leftrightarrow x'_i$) and the disjunct ($y_1 \land \cdots \land y_l \land z$) are true in $J$.

Moreover, $J(z) = \text{true}$ by definition. It remains to show that there does not exist a strictly “smaller” model of $\chi$.

Suppose to the contrary that there exists a model $J'$ of $\chi$, s.t. $J'$ is strictly smaller than $J$. Then there exists a variable $x_i, x'_i, y_j$ or $z$, s.t. this variable is true in $J$ and false in $J'$. We distinguish 3 cases:

MINIMAL MODEL SAT

Proof of the $\Sigma_2^P$-hardness (continued)

Case 1. Suppose that there exists a variable $x_i$ with $J(x_i) = \text{true}$ and $J'(x_i) = \text{false}$. Since both $J$ and $J'$ satisfy $\chi$, we thus have $J(x'_i) = \text{false}$ and $J'(x'_i) = \text{true}$. Hence, $J'$ cannot be smaller than $J$.

Case 2. Suppose that there exists a variable $x'_i$ with $J(x'_i) = \text{true}$ and $J'(x'_i) = \text{false}$. Analogously to case 1, this leads to a contradiction since $J(x_i) = \text{false}$ and $J'(x'_i) = \text{true}$.

Case 3. Suppose that either $J'(y_j) = \text{false}$ for some $j$ or $J'(z) = \text{false}$. Then clearly the disjunct ($y_1 \land \cdots \land y_l \land z$) is true in $J'$. For $J'$ to be a model of $\chi$, it must be a model of $\neg\varphi$, i.e., $J'$ is an extension of $I$ to $\{y_1, \ldots, y_l, z\}$, s.t. $\neg\varphi$ is true (or, equivalently, $\varphi$ is false) in $J'$.

This contradicts the assumption that $I$ is a partial assignment on $\{x_1, \ldots, x_k\}$, s.t. for any values of $\{y_1, \ldots, y_l\}$, the formula $\varphi$ is true.

Exercise. Prove also the other direction of the equivalence between QSAT$_2$ and MINIMAL MODEL SAT.
Further Properties of PH

Theorem

If there is a PH-complete problem, then the polynomial hierarchy collapses to some finite level.

Proof

Assume L is PH-complete. Then L ∈ Σ_i P for some i. But then any L’ ∈ Σ_{i+1} P reduces to L. This means that Σ_i P = Σ_{i+1} P since each level is closed under reductions. Thus PH collapses to the i-th level.

By the above theorem, PH probably has no complete problems. But of course each level of PH does (namely QSAT_i).

Restrictions on the Oracle Calls

Motivation

We consider two kinds of restrictions:

1. Number of oracle calls.
   - In DP, only 2 calls to an oracle are allowed.
   - Many natural problems require only O(\log n) oracle calls, since they come down to finding the optimal value via binary search, e.g.: max size of a clique, max/min. cardinality of a model, etc.

2. Adaptive vs. non-adaptive calls:
   - adaptive: The i-th question to the oracle may depend on the result of the previous (i − 1) calls to the oracle.
   - non-adaptive: otherwise.

Examples in P^{NP[log n]}

CARD-MINIMAL MODEL SAT

INSTANCE: Boolean formula ϕ and an atom z.
QUESTION: Is z true in a cardinality-minimal model of ϕ?

Proof of P^{NP[log n]}-membership

1. Compute the size K of a cardinality-minimal model of ϕ. This can be done by a binary search asking questions like “Does ϕ have a model of size ≤ k?” For this task, we need log n calls to an NP-oracle, where n = number of variables in ϕ.
2. Finally, ask an NP-oracle: “Is z true in some model I of ϕ, s.t. I sets exactly K variables to true?”

Analogously: CARD-MAXIMAL MODEL SAT

Complexity Classes

Definition

- Δ_2^P = P^{NP}: no restrictions on the oracle calls
- Δ_2^P[log n] = P^{NP[log n]}: only O(\log n) oracle calls allowed
- Δ_2^P[log n] is also referred to as Θ_2^P.

- P^{||}: polynomially many, non-adaptive oracle calls

- analogous classes for function problems: FP^{NP[log n]}, FP^{NP}

Remark

Recognizing a language L ∈ P^{||}: Machine M computes on input x in polynomial time a polynomial number of SAT-instances (or any other problem in NP) and then calls the oracle for all these instances at once. Based on the answers, M decides in polynomial time whether x ∈ L.
Examples of Optimization Problems in $\text{FP}^{\text{NP}[\log n]}$

**Some graph problems**
- **MIN-VERTEX COVER, MAX-CLIQUE, MAX-INDEP.-SET**: Given a graph $G = (V, E)$, what is the size of the smallest vertex cover (resp. the biggest clique or the biggest independent set)?
- **CHROMATIC NUMBER**: Given a graph $G = (V, E)$, what is the smallest number $k$, s.t. $G$ has a $k$-coloring?

**Some SAT-related problems**
- **CARD-MINIMAL-MODEL, CARD-MAXIMAL-MODEL**: Given a Boolean formula $\varphi$, what is the size of a minimal (resp. maximal) model of $\varphi$?
- **MAX-SAT**: Given a Boolean formula $\varphi$ in CNF, what is the maximal number of clauses that can be satisfied by a truth assignment?

Examples in $\text{P}^{\text{NP}}$ (Continued)

**LEX-MINIMAL MODEL SAT**

INSTANCE: Boolean formula $\varphi$, order $(x_1, \ldots, x_n)$ of the variables in $\varphi$.

QUESTION: Is $x_n$ true in the lexicographically smallest model of $\varphi$?

**Proof of $\text{P}^{\text{NP}}$-membership**

**LEX-MINIMAL MODEL SAT** can be decided by the following program with $n$ calls to an NP-oracle.

for $i := 1$ to $n$ do {
  check if $\varphi$ has a model $I$, s.t. (for all $j < i$: $I(x_j) = v_j$) and $I(x_i) = 0$;
  if yes then set $v_i := 0$, otherwise set $v_i := 1$;
}
if $v_n = 1$ then return true else return false.

Analogously: **LEX-MAXIMAL MODEL SAT**

**WEIGHT-MINIMAL MODEL SAT**

INSTANCE: Boolean formula $\varphi$, vector $(w_1, \ldots, w_n)$ of weights (i.e., positive integers) of the variables $(x_1, \ldots, x_n)$ in $\varphi$ and an atom $x_i$.

QUESTION: Is $x_i$ true in a weight-minimal model of $\varphi$?

**Proof of $\text{P}^{\text{NP}}$-membership**

1. Compute the minimal weight $W$ of all models of $\varphi$. This can be done by a binary search asking questions like “Does $\varphi$ have a model of weight $\leq w$?”. For this task, we need logarithmically many calls (w.r.t. the total weight) to an NP-oracle. These are polynomially many calls w.r.t. the representation of the weights $w_1, \ldots, w_n$.
2. Finally, ask an NP-oracle: “Is $x_i$ true in some model $I$ of $\varphi$, s.t. the total weight of the variables true in $I$ is $W$?”

Analogously: **WEIGHT-MAXIMAL MODEL SAT**

**Some graph problems**
- **MIN-WEIGHT-VERTEX COVER, MAX-WEIGHT-CLIQUE, MAX-WEIGHT-INDEP.-SET**: Given a graph $G = (V, E)$ and weights $w_i$ of the vertices, what is the size of the minimal total weight of a vertex cover, etc.?
- **TSP**: What is the length of the shortest tour through the $n$ cities?

**Some SAT-related problems**
- **WEIGHT-MINIMAL-MODEL, WEIGHT-MAXIMAL-MODEL**
- **MAX-WEIGHT-SAT**: Given a Boolean formula $\varphi$ in CNF and vector $(w_1, \ldots, w_m)$ of weights of the clauses $(c_1, \ldots, c_m)$ in $\varphi$, what is the maximal total weight of clauses that can be simultaneously satisfied by a truth assignment?
Theorem

\[ P^{\text{NP}[\log n]} = P^{\text{NP} \parallel} \]

Proof (continued)

“\(\subseteq\)”: Suppose that a machine \(M\) makes \(k \log n\) adaptive queries. For each of these queries, there are 2 possible outcomes. Hence, in total there are at most \(2^k \log n = n^k\) queries in the whole computation. Hence, the computation of \(M\) can be simulated by first computing the \(n^k\) possible queries and asking all of them at once to the oracle.

“\(\supseteq\)”: Suppose that a language \(L\) is decided by a TM \(M\) with polynomially many non-adaptive SAT queries. Then \(L\) can be decided with logarithmically many adaptive NP queries as follows:

- In \(O(\log n)\) queries determine the precise number \(K\) of “yes” answers to the non-adaptive queries. This can be done by binary search using the oracle: “Given a set of Boolean expressions, does it have satisfying truth assignments for at least \(k\) of them?"
- Ask the NP query: “Do there exist \(K\) satisfiable Boolean expressions such that if all other expressions were unsatisfiable (at this point, we know that they must be), then \(M\) would end up accepting?”

Remark. A succinct certificate for the last query consists of indices \(i_1, \ldots, i_K\) of Boolean expressions and models \(I_1, \ldots, I_K\) of them.

\[ \Delta_2^P \text{ and } \Theta_2^P\text{-Hardness Proofs} \]

Motivation

- Most problems in \(\Delta_2^P\) and \(\Theta_2^P\) are about optimization (asking if a minimum/maximum “solution” has certain properties).
- The complexity classes \(\Delta_2^P\) and \(\Theta_2^P\) are defined via oracle calls.
- We need to draw a connection between the result of a sequence of oracle calls and optimization.
- To this end, we define auxiliary problems \(\text{NP-MAX}\) and \(\text{LogNP-MAX}\), which will allow us to draw this connection.

Remark. In principle, algorithms in \(\Theta_2^P\) are allowed to make \(O(\log n)\) oracle calls (where \(n\) is the size of the input). For the sake of simplicity, we assume below that there are only \(\log n\) oracle calls.
Problems \textbf{NP-MAX} and \textbf{LogNP-MAX}

Theorem

- The problem \textbf{NP-MAX} is $\Delta_2P$-complete.
- The problem \textbf{LogNP-MAX} is $\Delta_2P[\log n]$-complete.

Proof of Membership

Maintain a bit vector $(v_1, v_2, \ldots)$ of the lexicographically maximal prefix of possible outputs of TM $M$ on input $x$:

- initialize $i$ to 0 and ask the following kind of questions to an NP-oracle:
- Does there exist a computation path of TM $M$ on input $x$, such that the first $i$ output bits are $(v_1, \ldots, v_i)$ and $M$ outputs yet another bit?
- If the answer to this oracle call is “no” then
  - the algorithm stops with acceptance if $(i \geq 1$ and $v_i = 1)$ holds
  - otherwise (i.e., $i = 0$ or $v_i = 0$) it stops with rejection.

\Delta_2P\text{-Hardness Proof of \textbf{NP-MAX}}

Consider an arbitrary problem $P$ in $\Delta_2P$, i.e., $P$ is decided in deterministic polynomial time by a Turing machine $N$ with access to an oracle $N_{\text{SAT}}$ for the SAT-problem.

Let $x$ be an arbitrary instance of problem $P$. We construct instance $M; x$ of \textbf{NP-MAX}, where we leave $x$ unchanged and we define $M$ as follows:

1. In principle, $M$ simulates the execution of $N$ on input $x$. However, whenever $N$ reaches a call to the SAT-oracle, with some input $\varphi_i$ say, then $M$ non-deterministically executes $N_{\text{SAT}}$ on $\varphi_i$.

Intuition. In the computation tree of $M$, the subtree corresponding to this non-deterministic execution of $N_{\text{SAT}}$ on $\varphi_i$ is precisely the computation tree of $N_{\text{SAT}}$ on $\varphi_i$.

2. On every computation path ending in acceptance (resp. rejection) of $N_{\text{SAT}}$, the TM $M$ writes 1 (resp. 0) to the output. After that, $M$ continues with the execution of $N$ as if it had received a “yes” (resp. a “no”) answer from the oracle call.

3. After the last oracle call, $M$ executes $N$ to the end. If $N$ ends in acceptance, then $M$ outputs 1; otherwise it outputs 0 as the final bit.

Conclusion of the Hardness Proof

It remains to show the following properties:

1. Correctness. Let $w$ denote the lexicographically maximal output string over all computation paths of $M$ on input $x$. Then the last bit of $w$ has value 1 if and only if $x$ is a positive instance of $P$.

2. Polynomial time. The total length of each computation path of $M$ on input $x$ is polynomially bounded in $|x|$.

3. Logarithmically bounded output. If the number of oracle calls of TM $N$ is logarithmically bounded in its input (i.e., problem $P$ is in $\Theta_2P$), then the size of the output produced by $M$ on input $x$ is also logarithmically bounded.
Correctness of the problem reduction

Claim. For every \(i \geq 1\) and for every bit vector \(w_i\) of length \(i\): \(w_i\) is a prefix of the lexicographically maximal output string over all computation paths of \(M\) on input \(x\) if and only if \(w_i\) encodes the correct answers of the first \(i\) oracle calls of TM \(N\) on input \(x\), i.e., for every \(j \in \{1, \ldots, i\}\):
- \(w_i[j] = 1\) if the \(j\)-th call of the \(N_{\text{SAT}}\) oracle yields a “yes” answer and
- \(w_i[j] = 0\) if the \(j\)-th call of the \(N_{\text{SAT}}\) oracle yields a “no” answer.

Proof of the Claim. By an easy induction on \(i\).

Conclusion. Let \(m\) denote the number of oracle calls carried out by TM \(N\) on input \(x\) and let \(w\) denote the lexicographically maximal output of TM \(M\) over all its computation paths.

- Then \(w\) has length \(m + 1\) such that the first \(m\) bits encode the correct answers of the oracle calls of TM \(N\) on input \(x\).
- By the construction of \(M\), we indeed have that the last bit of \(w\) is 1 (resp. 0), if and only if \(x\) is a positive (resp. negative) instance of \(P\).

Logarithmically bounded output

Suppose that \(P\) is an arbitrary problem in \(\Theta_2P\), i.e., TM \(N\) only has logarithmically many calls of the \(N_{\text{SAT}}\) oracle.

- Problem. Similarly to the polynomial-time bound of the computation of \(M\) on \(x\) (in the simulation of \(N\) with oracle \(N_{\text{SAT}}\) by a non-deterministic computation of \(M\)), we possibly produce computation paths which \(N\) on input \(x\) can never reach.
- We have to make sure that the size of the output on every computation path of \(M\) on input \(x\) is logarithmically bounded by adding a counter for the number of output bits.
- If the counter exceeds \(\log |x|\), then \(M\) outputs 0 and halts.

Polynomial time

Suppose that \(N\) on input \(x\) with \(|x| = n\) holds after \(\leq p(n)\) steps and that \(N_{\text{SAT}}\) on input \(\varphi\) with \(|\varphi| = m\) holds after \(\leq q(m)\) steps for polynomials \(p()\) and \(q()\).

Then the total length of the computation of \(N\) counting also the computation steps of the oracle machine \(N_{\text{SAT}}\) is bounded by a polynomial \(r(n)\) with \(r(n) = O(p(n) \cdot q(p(n)))\).

Observation. In our simulation of \(N\) with oracle \(N_{\text{SAT}}\) by a non-deterministic computation of \(M\) on input \(x\), we possibly produce computation paths which \(N\) on input \(x\) can never reach, e.g.: if the correct answer of \(N_{\text{SAT}}\) on oracle input \(\varphi_1\) is “yes”, then the continuation of the simulation of \(N\) on input \(x\) on all computation paths where answer “no” on oracle input \(\varphi_1\) is assumed, can never be reached by a computation of \(N\) (with oracle \(N_{\text{SAT}}\)) on input \(x\).

Problem. How can we be sure that the upper bound \(r(n)\) applies to every branch of the computation tree of \(M\) on input \(x\)?

Simple solution. Extend TM \(M\) by a counter such that \(M\) outputs 0 and halts if more than \(r(n)\) steps have been executed.

SAT-Variants in \(\Delta_2P\)

**LEX-MAXIMAL MODEL SAT**

INSTANCE: Boolean formula \(\varphi\), order \((x_1, \ldots, x_n)\) of the variables in \(\varphi\).

QUESTION: Is \(x_n\) true in the lexicographically biggest model of \(\varphi\)?

**LEX-MINIMAL MODEL SAT**

INSTANCE: Boolean formula \(\varphi\), order \((x_1, \ldots, x_n)\) of the variables in \(\varphi\).

QUESTION: Is \(x_n\) true in the lexicographically smallest model of \(\varphi\)?

**WEIGHT-MAXIMAL MODEL SAT**

INSTANCE: Boolean formula \(\varphi\), vector \((w_1, \ldots, w_n)\) of weights (i.e., positive integers) of the variables \((x_1, \ldots, x_n)\) in \(\varphi\) and an atom \(x_i\).

QUESTION: Is \(x_n\) true in a weight-maximal model of \(\varphi\)?

analogously: **WEIGHT-MINIMAL MODEL SAT**
Complexity Theory 6. The Polynomial Hierarchy 6.6. $\Delta_2 P$ and $\Theta_2 P$-Hardness Proofs

SAT-Variants in $\Theta_2 P$

LogLEX-MAXIMAL MODEL SAT

INSTANCE: Boolean formula $\varphi$, order $(x_1, \ldots, x_n)$ on some of the variables in $\varphi$ with $n \leq \log |\varphi|$.

QUESTION: Is $x_n$ true in the lexicographically biggest bit vector $(b_1, \ldots, b_n)$ that can be extended to a model of $\varphi$?

LogLEX-MINIMAL MODEL SAT

INSTANCE: Boolean formula $\varphi$, order $(x_1, \ldots, x_n)$ on some of the variables in $\varphi$ with $n \leq \log |\varphi|$.

QUESTION: Is $x_n$ true in the lexicographically smallest bit vector $(b_1, \ldots, b_n)$ that can be extended to a model of $\varphi$?

CARD-MAXIMAL MODEL SAT

INSTANCE: Boolean formula $\varphi$ and an atom $z$.

QUESTION: Is $z$ true in a cardinality-maximal model of $\varphi$?

analogously: CARD-MINIMAL MODEL SAT

Proof of Hardness

Reduction from NP-MAX resp. LogNP-MAX

Let $M; x$ be an arbitrary instance of NP-MAX (resp. LogNP-MAX), i.e.:
- $M$ is a Turing machine running in non-deterministic polynomial time and producing a binary string as output;
- $x$ is an input string to $M$.

Assumptions on the Turing machine $M$.
- every computation path of $M$ on input $x$ has length $N$ with $N = \rho(|x|)$ for some polynomial $\rho(\cdot)$.
- $M$ has two tapes, where tape 2 is the dedicated output tape.

LEX-MAXIMAL MODEL SAT and LogLEX-MAXIMAL MODEL SAT

Theorem

LEX-MAXIMAL MODEL SAT is $\Delta_2 P$-complete;
LogLEX-MAXIMAL MODEL SAT is $\Theta_2 P$-complete.
Hardness holds even if formulas are restricted to 3-CNF.

Proof of Membership

Idea of a polynomial-time algorithm with NP-oracle.
- maintain a bit vector $(v_1, \ldots, v_n)$ of the lexicographically maximal (prefix of a possible) model of $\varphi$
- for $i$ from 1 to $n$ call an NP-oracle asking:
  Does there exist a model of $\varphi$, s.t. the truth values of the first $i - 1$ variables $(x_1, \ldots, x_{i-1})$ are $(v_1, \ldots, v_{i-1})$ and variable $x_i$ is set to 1?
  If the answer is “yes”, then set $v_i = 1$; otherwise set $v_i = 0$.
- if $v_n = 1$, then stop with acceptance; otherwise stop with rejection.

Reduction from NP-MAX resp. LogNP-MAX

Analogously to the proof of the Cook-Levin Theorem, we construct a propositional formula $\varphi$ over the following variables:
- symbol$^{(i)}[\tau, \pi]$ for $1 \leq i \leq 2$, $0 \leq \tau \leq N$, $0 \leq \pi \leq N$ and $\sigma \in \Sigma$.
  Intuitive meaning: at instant $\tau$ of the computation, cell number $\pi$ on tape $i$ contains symbol $\sigma$.
- cursor$^{(i)}[\tau, \pi]$ for $1 \leq i \leq 2$, $0 \leq \tau \leq N$ and $0 \leq \pi \leq N$.
  Intuitive meaning: at instant $\tau$, the cursor of tape $i$ points to cell number $\pi$.
- state$[\tau]$ for $0 \leq \tau \leq N$ and $s \in K$.
  Intuitive meaning: at instant $\tau$, the NTM $T$ is in state $s$.

$x, z$ additional variables $\bar{x} = (x_1, x_1', x_2, x_2', \ldots, x_m, x_m')$ and $z$ to encode the output string $w$ and the last bit.
Form of the propositional formula $\varphi$

As in the proof of the Cook-Levin Theorem, formula $\varphi$ consists of a big conjunction containing the following groups of conjuncts:

- initialization facts
- transition rules
- uniqueness constraints
- inertia rules
- Since this is a Turing machine producing output (and not a decision procedure), we have to drop the acceptance condition.

In addition, we now have a further conjunct to encode the output string $w$ of length $\leq m$ and the value of the last bit of $w$.

Transformation into 3-CNF and ordering on the variables

- Finally, transform $\varphi$ into $\varphi^*$ in 3-CNF by a standard transformation (e.g., the Tseytin transformation).
- Let $\vec{y}$ denote the vector of variables $\text{symbol}_\alpha^{(i)}[\tau, \pi]$, $\text{cursor}_\alpha^{(i)}[\tau, \pi]$, and $\text{state}_\alpha[i]$ plus the additional variables introduced by the transformation into 3-CNF.
- Let the variables in $\vec{y}$ be arranged in arbitrary order with $\vec{y} = \{y_1, \ldots, y_k\}$.
- In the reduction from $\text{NP-MAX}$, define the following order on all variables in $\varphi^*$: $x_1 > x_1' > \cdots > x_m > x_m' > y_1 > \cdots > y_k > z$.
- In the reduction from $\text{LogNP-MAX}$, define an order only on part of the variables in $\varphi^*$, namely: $x_1 > x_1' > \cdots > x_m > x_m' > z$, i.e., we ignore the variables in $\vec{y}$.

Encoding of the output string and the last bit

$$\chi = \bigwedge_{\pi=1}^{m} x_\pi \leftrightarrow \text{symbol}_\alpha^{(2)}[N, \pi] \land$$
$$\bigwedge_{\pi=1}^{m} x_\pi' \leftrightarrow \neg\text{symbol}_\alpha^{(2)}[N, \pi] \land$$
$$z \leftrightarrow \bigvee_{\pi=1}^{N} \left(x_\pi \land \bigwedge_{\pi' = \pi + 1}^{N} \neg x_{\pi'}\right)$$

- For every $i$, variable $x_i$ is true in a model $J$ of $\varphi$ if the $i$-th bit of $w$ (along the computation path of $M$ corresponding to $J$) is 1.
- Consequently, truth value false of $x_i$ can mean that, on termination of $M$, the symbol in the $i$-th cell of tape 2 is either 0 or $\bot$.
- Variable $x_i'$ is used to distinguish these two cases: $x_i'$ is false if and only if the symbol in the $i$-th cell of tape 2 is $\bot$.
- Variable $z$ is true in model $J$ of $\varphi$ if and only if the last bit of $w$ is 1.

Correctness of the reduction from $\text{NP-MAX} / \text{LogNP-MAX}$

- Let $w$ denote the lexicographically maximal output string produced by $M$ on input $x$. We have to show: the last bit of $w$ is 1 if and only if $z$ is true in the lexicographically maximal model of $\varphi^*$.
- As in the proof of the Cook-Levin Theorem, every computation path of $M$ corresponds to one or more models of $\varphi^*$.
- We construct a truth assignment $J$ on the variables $\vec{x}$, $\vec{y}$, and $z$ in $\varphi^*$:
  - $J$ restricted to $\vec{y}$ is chosen such that it is lexicographically maximal among all truth assignments on $\vec{y}$ corresponding to a computation path of $M$ on input $x$ and with output $w$.
  - We claim that $J$ is the lexicographically maximal model of $\varphi^*$.
  - From this, the correctness of the reduction follows immediately.
- For the reduction from $\text{LogNP-MAX}$, we thus make use of the following observation: the truth assignments of $J$ on $\vec{x}$ and $z$ are uniquely defined by the output $w$ of $M$ (independently of $\vec{y}$ in $J$).
Complexity Theory 6. The Polynomial Hierarchy 6.6. ∆ 2 P and Θ 2 P-Hardness Proofs

Conclusion of the correctness proof

It remains to show that \( J \) is the lexicographically maximal model of \( \varphi^* \).
Suppose to the contrary that there exists a bigger model \( J' \) of \( \varphi^* \). We distinguish 3 cases according to the group of variables where \( J' \) is bigger:

1. If \( J' \) is bigger than \( J \) on \( \vec{x} \), then (since the truth values of \( \vec{x} \) encode the output string of some computation of \( M \) on \( x \)) there exists a bigger output than \( w \). This contradicts our assumption that \( w \) is maximal.
2. If \( J' \) coincides with \( J \) on \( \vec{x} \) and \( J' \) is bigger than \( J \) on \( \vec{y} \), then \( J' \) restricted to \( \vec{y} \) corresponds to a computation path producing the same output string \( w \) as the computation path encoded by \( J \) on \( \vec{x} \). This contradicts our choice of truth assignment \( J \) on \( \vec{y} \).
3. The truth value of \( z \) in any model of \( \varphi^* \) is uniquely determined by the truth value of \( \vec{x} \). Hence, it cannot happen that \( J' \) and \( J \) coincide on \( \vec{x} \) but differ on \( z \).

WEIGHT-MAXIMAL MODEL SAT is \( \Theta_2^P \)-complete

Idea of \( \Theta_2^P \)-hardness proof

By reduction from LEX-MAXIMAL MODEL SAT:
Consider an arbitrary instance \( \varphi; (x_1, \ldots, x_n) \), where \( \varphi \) is a Boolean formula over variables \( X \) and \( (x_1, \ldots, x_n) \) is an ordering of the variables in \( X \).

We define the instance \( \varphi; (x_1, \ldots, x_n); (w(x_1), \ldots, w(x_n)); z \) of WEIGHT-MAXIMAL MODEL SAT as follows:

- Formula \( \varphi \) (and, hence, also variable set \( X \)) is left unchanged.
- For every \( i \in \{1, \ldots, n\} \), we define the weight \( w(x_i) = 2^{n-i} \).
- We set \( z = x_0 \).

Idea. Simulate lexicographical ordering by weights of variables \( x_i \):
\( w(x_i) \) is greater than the sum of weights \( w(x_{i+1}) + \cdots + w(x_n) \).

CARD-MAXIMAL MODEL SAT is \( \Theta_2^P \)-complete

Idea of \( \Theta_2^P \)-hardness proof

By reduction from LogLEX-MAXIMAL MODEL SAT:
consider an arbitrary instance \( \varphi; (x_1, \ldots, x_n) \) with a Boolean formula \( \varphi \) and an ordering \( (x_1, \ldots, x_n) \) of logarithmically many variables in \( \varphi \);
let \( Y = \{y_1, \ldots, y_m\} \) denote the remaining variables in \( \varphi \).

To define an instance \( \psi; z \) of CARD-MAXIMAL MODEL SAT, we add the following fresh variables:
- “copies” of each variable \( x_i \), i.e., for every \( i \in \{1, \ldots, n\} \), we introduce \( 2^n - i \) new variables \( x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)} \) with \( n = 2^{n-i} - 1 \).
- a primed copy of each variable in \( Y \): \( Y' = \{y'_1, \ldots, y'_m\} \).

Instance of CARD-MAXIMAL MODEL SAT

Formula \( \psi \) and distinguished variable \( z \).

\[
\psi = \varphi \land \\
\land \bigwedge_{i=1}^{n} (x_i \leftrightarrow x_i^{(1)}) \land \cdots \land (x_i \leftrightarrow x_i^{(n)}) \land \\
\land \bigwedge_{i=1}^{m} y_i \leftrightarrow \neg y'_i \\
z = x_n
\]

Idea.

- Simulate weight of \( x_i \) by “copies” \( x_i^{(j)} \), which are forced to have identical truth values as \( x_i \) in every model of \( \psi \).
- Use the primed variables \( y'_i \) to encode the dual of \( y_i \). They thus make the cardinality of models of \( \psi \) indistinguishable on \( Y \cup Y' \).
CARD-MINIMAL MODEL SAT is $\Theta_2^P$-complete

Idea of $\Theta_2^P$-hardness proof

- By reduction from CARD-MAXIMAL MODEL SAT:
  consider an arbitrary instance $\varphi; x_i$,
  where $\varphi$ is a Boolean formula over the variables $X = \{x_1, \ldots, x_n\}$.
- Add primed and double-primed copies of the variables, i.e., $X' = \{x'_1, \ldots, x'_n\}$ and $X'' = \{x''_1, \ldots, x''_n\}$.
- Define instance $\psi; x_i$ of CARD-MINIMAL MODEL SAT with
  $$\psi = \varphi \land \bigwedge_{i=1}^n ((x_i \leftrightarrow \neg x'_i) \land (x_i \leftrightarrow \neg x''_i))$$
- Idea. The cardinality-minimal models of $\psi$ restricted to the variables $X$ are precisely the cardinality-maximal models of $\varphi$.

Learning Objectives

- Oracle machines
- Complexity classes: DP, $\Delta_i^P$, $\Sigma_i^P$, $\Pi_i^P$, PH
- The intuition of these classes and complete problems
- Restrictions on the oracle calls:
  $\Delta_2^P[\log n] = P^{NP[\log n]} = P^{NP[\parallel]}$
- oracle calls vs. optimization
- Problem reductions in $\Sigma_2^P$
- Properties of PH (sufficient conditions for PH to collapse)
- Characterization of $\Sigma_i^P$ and $\Pi_i^P$ via certificates
- The power of alternation: limited alternation in QSAT;
  The classes $\Delta_2^P$ and $\Delta_2^P[\log n]$