Komplexitätstheorie 2004
Problemset 2

November 25, 2004

Definitions

Satisfiability Variants

- NaeSat is the set of CNF-expressions where each clause contains *exactly three* literals, such that there is an assignment which does not assign all three literals of any clause the same truth value. I.e., a CNF-expression is not-all-equal satisfiable, if there is an assignment which satisfies at least one literal in each clause but does not satisfy all three literals in any clause simultaneously.
- An instance of Max2Sat consists of a positive integer K and a CNF-expression ϕ where each clause contains at most two literals. The question is whether there is a Boolean assignment which satisfies at least K clauses simultaneously in ϕ.

Graph Problems

- A VertexCover-instance is consists of an undirected graph $\langle V, E\rangle$ and a positive bound K. The problem is to find a subset $V' \subseteq V$ with $|V'| \leq K$ such that $\{u, v\} \cap V' \neq \emptyset$ for all $\langle u, v\rangle \in E$.
- HyperGraphVertexCover is a generalization of VertexCover. Hypergraphs allow *more than two* vertices per edge, i.e., in a hypergraph $\langle V, H\rangle$ an hyperedge $h \in H$ is subset of V. HyperGraphVertexCover asks for a subset $V' \subseteq V$ with $|V'| \leq K$ such that for all $h \in H$ we have a $v \in V'$ with $v \in h$.
- DominatingSet has directed graphs $\langle V, E\rangle$ together with a bound K as instances. The task is to find a subset $D \subseteq V$ with $|D| \leq K$ such that for each $v \in V - D$ there is a $u \in D$ with $\langle u, v\rangle \in E$.
- A LongestPath-instance contains a directed graph G, two vertices s, t and a positive integer K. The question is whether there is a simple path (i.e., a path that passes a vertex at most once) through G of length K which starts at s and ends at t – or longer?
- In FeedbackVertexSet we are asked to find a subset $V' \subseteq V$ with $|V| \leq K$ of the vertices of a directed graph $G = \langle V, E\rangle$ such that any cycle in G contains at least one vertex in V'.

- In FEEDBACKEDGESET we are asked to find a subset $E' \subseteq E$ with $|E| \leq K$ of the edges of a directed graph $G = \langle V, E \rangle$ such that any cycle in G contains at least one edge in E'.

Set Problems

- SETCOVER has instances of the form $\langle C, U, K \rangle$ where $c \in C$ are subsets of the universe U and K is a positive integer. The task is to find a subset $C' \subseteq C$ with $|C'| \leq K$ such that $\bigcup_{c \in C'} c = U$.

Numerical Problems

- The problem: Given a purse with a set of coins, and given an amount of money to pay, can you pay the amount *precisely*?

  Formally: An instance $\langle A, S \rangle$ of SUBSETSUM is a set of positive integers $A = \{a_1, \ldots, a_n\}$ and a positive integer S. $\langle A, S \rangle$ is a positive instance, iff there is a subset $A' \subseteq A$ such that $\sum_{a \in A'} a = S$.[1]

- A special case of SUBSETSUM is PARTITION– given a set of coins, can you divide it into two sets of equal value?

  Formally: A PARTITION-instance $\langle A \rangle$ is a positive one, iff there is a subset A' such that $\sum_{a \in A'} a = \sum_{a \in A - A'} a$.

- The Problem: Given a set of items (each with a value and a weight), can you find a subset of maximum value fitting your knapsack?

  Formally: An instance $\langle A, w, v, W, V \rangle$ of KNAPSACK comes with a set of items $A = \{a_1, \ldots, a_n\}$, a weight function $w : A \to \mathcal{N}$, a value function $v : A \to \mathcal{N}$, a positive capacity W, and the minimal value V to be packed into the knapsack.
  The task is to find a subset $A' \subseteq A$ such that $\sum_{a \in A'} w(a) \leq W$ with $\sum_{a \in A'} v(a) \geq V$.

- The Problem: Given a set of items (each with a certain weight), how many knapsacks do you need to carry them?

  Formally: An instance $\langle A, C, B \rangle$ of BINPACKING is a set of positive integers $A = \{a_1, \ldots, a_n\}$ and two positive integer C and B. Given an instance $\langle A, C, B \rangle$, the task is to find a partition $\langle B_1, \ldots, B_k \rangle$ of A with

  - $k \leq B$
  - $\sum_{a \in B_i} a \leq C$ for $1 \leq i \leq k$
  - $\bigcup_{i=1}^{i=k} B_i = A$.

- In INTEGERPROGRAMMING we have an integer matrix A and two integer vectors b, c and an integer B, determine if there is an integer vector x such that $Ax \leq b$ and $cx \geq B$?

[1]Note that this formulation does not allow to have more than one item with the same weight. Whether multiple items of the same weight are allowed or not is unimportant with respect to the complexity of the problem.

Problem 1 – 3Sat $\leq$ Max2Sat

This reduction will be a "local replacement" reduction – you need to replace any clause with three literals with a specific set of 2-clauses.

Problem 2 – 3Sat $\leq$ NaeSat

Again, this reduction will use a "local replacement" where each clause is replaced with a new set of clauses.

Problem 3 – HamiltonianCycle $\leq$ HamiltonianPath $\leq$ LongestPath

Consider the directed case. Both reductions are require only minimal changes on the given instance!

Problem 4 – VertexCover = HyperGraphVertexCover

You have to show two reductions: First you need to show that VertexCover $\leq$ HyperGraphVertexCover. This reduction is a reduction by restriction, i.e., you need to show that HyperGraphVertexCover is a generalization of VertexCover. Then you must show HyperGraphVertexCover $\leq$ VertexCover. To do so, you will have to design a gadget to replace any hyper-edge with an ordinary graph.
This example is a bit harder than the preceding ones.

Problem 5 – HyperGraphVertexCover = SetCover

These two problems are really the same – there reformulations of each other. Show how to transform a HyperGraphVertexCover-instance into a SetCover-instance and vice versa.

Problem 6 – VertexCover $\leq$ DominatingSet

This reduction is a local replacement reduction. You will have to replace each edge with a gadget.

Problem 7 – Partition $\leq$ Knapsack and Partition $\leq$ BinPacking

In both cases, you can show that Partition is a special case – i.e., both are reductions by restriction.

Problem 8 – IntegerProgramming is NP-hard

Choose two problems and reduce them to IntegerProgramming.

Problem 9 – VertexCover $\leq$ FeedbackVertexSet

You will have to replace each undirected edge in VertexCover by some construct to obtain a proper FeedbackVertexSet instance.

Problem 10 – FeedbackVertexSet $\leq$ FeedbackEdgeSet

Again, a the reduction will be based on a local replacement. This time, you will replace each vertex of the FeedbackVertexSet with some small replacement.

Challenge 1 – Closure Properties of P and NP

Let $\mathcal{C}$ be a class of decision problems. Then $\mathcal{C}$ is closed under

- union if $L \cup K = \{x | x \in L \vee x \in K\}$ is in $\mathcal{C}$ for all $L, K \in \mathcal{C}$.
- intersection if $L \cap K = \{x | x \in L \wedge x \in K\}$ is in $\mathcal{C}$ all $L, K \in \mathcal{C}$.
- the Kleene-star if $L = \bigcup_{i>0} L^i$ is in $\mathcal{C}$ for all $L \in \mathcal{C}$, where $L^k = \{x_1 \bullet x_2 \bullet \ldots \bullet x_k | x_1, \ldots x_k \in L\}$ and $\bullet$ denotes the string-concatenation.

We do not know whether $\mathbf{P} = \mathbf{NP}$. It is not too hard to prove that **NP** is closed under all three operators defined above. So if **P** would not be closed under one of these operators, then $\mathbf{P} \neq \mathbf{NP}$. Intersection and union are easy to prove for **P** – however, the Kleene-star is more interesting.

1. Prove that **NP** is closed under union, intersection and the Kleene-star.
2. Prove that **P** is closed union, intersection and the Kleene-star.
   Hint: To show that **P** is closed under the Kleene-star, construct a polynomial time algorithm explicitly. Ensure that this algorithm is only evaluating a polynomial number of sub-strings of a given input x. One possibility to do so, is to maintain a list of valid prefix end-points (i.e., points where a series $x_1 \bullet \ldots \bullet x_i$ ends) and to design the algorithm in a way that guarantees that only a polynomial number of updates occur to this list.

Challenge 2 – IntegerProgramming $\in$ NP

Prove that IntegerProgramming is in **NP**. At a first glance, it is straightforward to a write a nondeterministic program which solves any given IntegerProgramming-instance. The problem is the *size* of the given instance – you have to give an appropriate upper bound for the size of the integers which appear in the solution vector.