CS 577: Introduction to Algorithms Fall 2019

Structuring NP-Hardness Submissions

Instructor: Dieter van Melkebeek TA: Andrew Morgan

This note presents a suggested outline for submissions that prove NP-hardness. Following the outline is an example write-up for HW10 #2, which asks to prove that a newly-posed problem, "multiple interval scheduling", is NP-hard.

Be aware that proving NP-*completeness* requires a little more than what is documented below: one must also prove that the problem is in NP.

1. Describe your reduction

   (a) State the problem you reduce from, henceforth A.

   In the example, A is Maximum Independent Set (MIS). (It's not necessary to restate a specification of the problem, as is done in the example, but doing so is a helpful aide when doing the rest.)

   (b) Give clear instructions that, given an instance x of A, declare how to construct an instance x' of the problem reduced to.

   In the example, it is described how to make an instance of multiple interval scheduling from an instance of MIS.

   (c) Say how to answer x given the answer for x'.

   In the example, MIS and multiple interval scheduling are viewed in their decision variants, so this part amounts to copying the "yes" or "no" answer.

   In these steps, prefer prose for describing the reduction.

2. Prove correctness

   - Typically this means proving a one-to-one and onto correspondence between solutions for x and solutions for x'.
   - For decision and search problems, it means proving there exists a solution for x iff there exists a solution for x'.

   In the example, we establish two claims that collectively prove that the assumed instance of MIS has a solution if and only if the instance of multiple interval scheduling constructed from it has a solution.

3. Analyze running time

   - This is the time to do the construction (step 1b in this outline) plus the time to translate solutions of x' to solutions of x (step 1c).
   - Do *not* consider the running time for solving x'. Remember the point of NP-hardness is to justify that there is (assuming P$\neq$NP) *no* efficient algorithm to do this.

Example Solution for HW 10 #2 (NP-hardness reduction)

Reduce from Max. Independent Set (MIS) to Mult. Int. Sched.

MIS: INPUT: graph G, parameter $K \in \mathbb{N}$
OUTPUT: Y/N, whether G has an indep. set of size $\geq K$.

Reduction: Given $G = (V, E)$, K, make an instance of Mult. Int. Sched. as follows.
Number the edges in E $1, 2, \ldots, m$. Let $t : E \to \{1, 2, \ldots, m\}$ be the function sending an edge to its number.
For each vertex $v \in V$, make a job J_v, that uses times $(t(e) - 1/3, t(e) + 1/3)$ for each edge e incident to v.
Pass this set of jobs and the same K to Mult. Int. Sched.
If it says "Y" (there is a set of $\geq K$ non-interfering jobs), say "Y" (there is an ind. set in G w/ $\geq K$ vtxs)
Otherwise say "N".

Running Time:
- Jobs can be constructed in time $O(n+m)$
- $O(1)$ time to decide whether to say "Y" or "N".

Total time: $O(n+m)$ = polynomial in $|G|, K$.

Correctness:

Claim: If there is an independent set in G of size $\geq K$, then there is a set of $\geq K$ compatible jobs.

pf Let $S \subseteq V$ be an independent set in G of size $\geq K$. Consider the set of jobs $\{J_v : v \in S\}$. If J_v and J_w $(v, w \in S)$ overlap, there is $e \in E$ s.t. they overlap at time $t(e)$. But such e would have to be an edge connecting v and w, while S is an independent set. So the jobs J_v, $v \in S$ are all compatible. □

Claim: If there is a set of $\geq K$ compatible jobs, then there is an independent set in G of size $\geq K$.

pf Let S be a set of K compatible jobs. Consider the set $\{v \in V \mid J_v \in S\}$. If there were an edge, e, between two such vertices, say v and w, then J_v and J_w overlap at time $t(e)$. Since all jobs in S are compatible, it follows that $\{v \in V \mid J_v \in S\}$ is an independent set in G. □