Computational Complexity: Homework 2

Arnau Abella
Universitat Politècnica de Catalunya

April 2, 2020

1 Self-reducibility

Exercise: Self-reducibility Show that if the decision version k-COL of the k-coloring problem (k-COL) can be solved in polynomial time, then its search version SEARCH-k-COL can also be solved in polynomial time.

We are going to describe an algorithm that can solve SEARCH-k-COL in polynomial time.

Algorithm 1: Naïve SEARCH-k-COL Algorithm

Input : Given the graph $G = \langle V, E \rangle$ and k colors.
Output: A solution $a := a_1, \ldots, a_{|V|}$ of SEARCH-k-COL, $a_i = \{1..k\}$ or $UNSAT$.

for $i \leftarrow 1$ **to** $|V|$ **do** // $|V|$ times
  for $j \leftarrow 1$ **to** $|k|$ **do** // $|K|$ times
    // Each assignment is a copy of the vector a
    // At most $|v|^{|k|}$ copies
    $a := (a_1, \ldots, a_{i-1}, a_i = j)$;
  end
  if $i = |V|$ **then**
    Test if $x := (x_1 = a_1, \ldots, x_n = a_{|V|})$ is a valid assignment using k-COL; // p(n)-time
    If it is a valid assignment, then *halt* and return a;
  end
end
return $UNSAT$;

The total cost of the algorithm 1 is $\mathcal{O}(|k|^{|V|} \cdot n^c)$ where $|k| \leq n$ and $c > 0$.

We found a polynomial time algorithm that solves SEARCH-k-COL in polynomial time.

2 Zero-error

Exercise: Zero-error Show that if k-COL is in BPP, then it is also in ZPP. In other words, show that if there is a probabilistic polynomial time algorithm that decides whether a graph has a valid k-COL, and does so with bounded error, then there is also a probabilistic polynomial time algorithm that does that with zero error (in the sense of the definition of ZPP).

Since k-COL is NP-complete, we can generalize the problem to

$$NP \subseteq BPP \implies NP = ZPP$$

The basic inclusions are the following:

$$\begin{aligned} P \subseteq ZPP \subseteq RP &\subseteq NP \\ P \subseteq ZPP \subseteq coRP &\subseteq coNP \\ RP &\subseteq BPP \\ coRP &\subseteq BPP \end{aligned}$$

P is contained in $ZPP = RP \cap coRP$, which is contained in BPP. Moreover, RP is contained in NP and $coRP$ is contained in $coNP$. **But we don't know how NP and $coNP$ relate to BPP.**

It follows from $NP \subseteq BPP$, that $NP = RP$. That means that $coRP = coNP$ and $ZPP = NP \cap coNP$. So basically in this case, we have P, which contains NP and $coNP$, and those are both contained in BPP.

Assuming $\boldsymbol{NP = coNP}$, which is an *open problem*, we can show that k-COL is in ZPP.

3 Logarithmic space

Exercise: Logarithmic space Show that 2-COL is in P. Show that it is even in NL. [Since it is known that 3-COL is NP-complete, make sure that you are using something special about 2-COL in your proof, and state what.]

First, let's review what we know so far.

Theorem 3.1. *$E2$-$SAT \in NL$*

Proof. Given a 2-COL F, we construct the implication graph $G(F)$ of F:

- One vertex for every literal
- Two edge $\neg l \to l'$, $\neg l' \to l$ for every clause

Claim: If $G(F)$ contains a cycle that contains a variable and it's negation, then F is $UNSAT$.

Claim: (contrapositive) If $G(F)$ does not contains a cycle that contains a variable and it's negation, then F is SAT. □

Therefore, if we can give an algorithm to *reduce* k-COL to E2-SAT in polynomial time, we can use the theorem 3.1 to solve it in polynomial polynomial time.

The algorithm is the following:

- There will be one clause for each edge, and one literal for each vertex.
- Each clause will contain exactly two literals that represent the edge $\vec{uv}$, and each literal will codify the color of the vertex i as there are only two possible colors.