15-859FF: Coping with Intractability **CMU, Fall 2019**
HW #5 (out Friday Nov 22, 2019) **Due:** Friday, Dec 6, 2019

You may discuss all problems on this HW with others, but groups of size at most 3, please. Please write down the names of your collaborators. Also, you should write your solutions yourself.

Exercises

Exercises are for fun and edification, please do not submit. **But do try to solve them, they may have ideas that are useful elsewhere, probably even in this HW!**

1. Consider the following reduction from the Nearest Codeword Problem (NCP) to Minimum Distance Problem (MDP). Recall that an instance of NCP (for binary linear codes) is given by (C, t, k) where $C \in \mathbb{F}_2^{n\times\ell}$, $t \in \mathbb{F}_2^n$, and $k \in \mathbb{N}$, and the goal is to determine if there exists $x \in \mathbb{F}_2^\ell$ such that the codeword Cx differs from t in at most k locations, i.e., $\mathrm{wt}(Cx - t) \le k$. Now form a matrix $C' \in \mathbb{F}_2^{n\times(\ell+1)}$ as $C' = (C \mid t)$, i.e., we add t as a last column to C.
   (a) Prove that the distance of C' is at most k if $\mathrm{wt}(Cx - t) \le k$ for some x.
   (b) Is the above a correct reduction from NCP to MDP? If so, prove its correctness. If not, what condition on C would make this a correct reduction?

2. Show that given a generator matrix of a subspace C of $\mathbb{F}_2^n$, one can a parity check matrix for C in $O(n^3)$ time, and vice versa.

3. Consider the code $C \subseteq \mathbb{F}_2^{2^t-1}$ defined as $C = \{c \mid Hc = 0\}$ where H is a $t \times (2^t - 1)$ matrix whose columns are all the nonzero vectors in $\mathbb{F}_2^t$.
   (a) What is the distance of the code C?
   (b) Give a generator matrix for the code, i.e., a matrix G whose columns span C. (Hint: This will be a $(2^t - 1) \times (2^t - 1 - t)$ matrix.)

4. Give a deterministic polynomial time reduction from Clique to Biclique, thus proving the NP-hardness of Biclique. (As mentioned in class, the obvious reduction based on taking the double cover does *not* work.)

Problems

Solve any five of the following problems.

1. Suppose a graph G has a vertex cover of size k. Prove that the treewidth of G is at most k.

2. In the *exact version* of NCP, given a matrix $H \in \mathbb{F}_2^{m\times n}$, $s \in \mathbb{F}_2^m$, and $k \in \mathbb{N}$, the goal is to determine if there are *exactly* k columns of H that add up to s.
   (a) Prove that the exact version of NCP is W[1]-hard, when parameterized by k.
   (b) The exact version of MDP is defined analogously: given a matrix $H \in \mathbb{F}_2^{m\times n}$ and $p \in \mathbb{N}$, the goal is to determine if there are *exactly* p columns of H that add up to 0. The quantity p is taken as the parameter in the parameterized version.

Give a parameterized reduction from the exact version of NCP to the exact version of MDP. Conclude that the exact version of MDP is W[1]-hard.

(This is Exercise 13.19 in the Cygan et al book which asks for a reduction from Exact Odd Set to Exact Even Set. But please try to solve the problem without seeing their hint. It is not a hard reduction!)

3. In this exercise, your task is to prove that MDP Turing-reduces to NCP in an approximation preserving way.

   Formally, suppose there is an (approximate) NCP-oracle that given a (binary linear) code $C' \subseteq \mathbb{F}_2^{m \times n}$ and a target $t' \in \mathbb{F}_2^n$ outputs a vector $w \in C'$ whose distance from t' is at most $f(n)$ times the distance of t' and its closest codeword in C', for some function $f : \mathbb{N} \to \mathbb{N}$.

   Describe an algorithm, that given a binary linear code $C \subseteq \mathbb{F}_2^n$, uses the NCP oracle $O(n)$ times, and outputs a non-zero vector in C of Hamming weight at most $f(n)$ times the minimum distance of C.

   Hint: Suppose $c_1, \ldots, c_\ell$ form a basis of C. What are useful target vectors t' and codes C' to use that will help find nonzero codewords of small Hamming weight in C?

4. Recall the Exact Hitting Set problem parameterized by size of hitting set: Given a family $\mathcal{F} \subseteq 2^U$ of subsets from a universe of size $|U| = n$, are there at most k elements that hit each subset in $\mathcal{F}$ exactly once. Prove that Exact Hitting Set can be solved in $f(k)n^{\omega k/3+O(1)}|\mathcal{F}|^{O(1)}$ time, where ω is the matrix multiplication exponent (so $N \times N$ matrices can be multiplied in N^ω time).

   Hint: Show that the following problem can be solved in $f(k)n^{\omega k/3+O(1)}W^{O(1)}$ time: given an n-vertex graph whose vertices have integer weights in $\{0, 1, 2, \ldots, W\}$ (for some integer W), and two parameters k and $t \in \{0, 1, 2, \ldots, W\}$, find a k-clique in the graph whose vertices have total weight exactly t. To solve this hint, think back to HW#0. (For partial (half) credit, you may solve the problem assuming the hint without a solution to the hint.)

5. In the Monotone Circuit Satisfiability problem, we are given a monotone Boolean circuit (a circuit without any NOT gates) C and a positive integer k, does C have a satisfying assignment with k 1's. We get a parameterized version using the solution size k as the parameter.

   (a) Prove that Monotone Circuit Satisfiability is W[2]-hard.

   (b) Prove that Monotone Circuit Satisfiability is W[1]-hard for weft-1 circuits of depth $O(\log k)$. Hint: Reduction from Multicolored Clique.

6. In this exercise, your goal is to give a reduction that will show that solving the Nearest Codeword Problem (NCP) requires $2^{(1-o(1))n}$ time assuming SETH.

   (a) Let $k \geq 1$ be a positive integer.

   Construct a matrix $G \in \mathbb{F}_2^{2^k \times k}$ with the following property.

   For every $z \in \mathbb{F}_2^k$, there is a z-isolating target $t^{(z)} \in \mathbb{F}_2^{2^k}$ such that: (a) the Hamming distance between Gz and $t^{(z)}$ is 2^k, and (b) the Hamming distance between Gz' and $t^{(z)}$ is 2^{k-1} for every $z' \neq z$.

   (b) Using the above construction, give a polynomial time reduction from Max k-SAT to NCP with the following properties:

i. The reduction maps instances ϕ of Max k-SAT with n variables and m clauses to a code C of dimension at most n (that is, C is spanned by n vectors) and a target t whose distance to C is precisely $2^k(m - \mathrm{OPT}/2)$ where OPT is the optimum value of the Max k-SAT instance.

ii. Conclude that NCP on dimension n codes does not admit a $2^{(1-\epsilon)n}$ time algorithm for any fixed $\epsilon > 0$, assuming SETH.

7. The Ajtai-Kumar-Sivakumar algorithm we saw in lecture can be modified to output the shortest vector in an n-dimensional lattice $L = L(\{b_1, \ldots, b_n\}) \subseteq \mathbb{R}^n$ that is *outside* a given subspace $X \subseteq \mathbb{R}^n$ (of dimension $< n$). In this exercise, we will see the core idea of how to accomplish this. (The SVP of course corresponds to the case when $X = \{0\}$.)

   Let λ denote the length of the shortest vector in $L \setminus X$ (in Euclidean norm). We will use the notation from the AKS algorithm as presented in lecture. Suppose we sample the points $x_1, \ldots, x_N$ (for large enough N) from $B(0, r)$ for a radius r satisfying

$$\frac{1}{2}(1+\epsilon)\lambda \leq r \leq \frac{1}{2}(1+\epsilon)^2\lambda \tag{1}$$

   for some $\epsilon \in (0, 1/2)$.[1]

   Assume also that in each iteration of Step 2 we run a variant of seiving where we reduce R to R/a for $a = 1 + 2/\epsilon$, and we run the loop of Step 2 till the radius of the ball in which the y_i's are contained is at most $(1+\epsilon)r$. (You might want to check the choice of parameter in the sieving enables this.)

   Also in the final step (Step 3), suppose we return the shortest lattice vector amongst the vectors $(y_i - z_i)$ for $i \in Z$ that is outside X (if one exists, otherwise we output an arbitrary basis vector $b_i \notin X$).

   Prove that when run with large enough $N \leq (1/\epsilon)^{O(n)} \log\big(\max_i\{\|b_i\|_2\}\big)$, the above-sketched algorithm will output a vector $v \in L \setminus X$ with $\|v\|_2 \leq (1+\epsilon)^3\lambda$ with high probability (over the choice of the random x_i's in $B(0, r)$).

   (You need not analyze the time complexity of the algorithm, which can be shown to be $(1/\varepsilon)^{O(n)}$ times a polynomial in the input length.)

[1] We of course don't know λ, but by trying r values increasing geometrically with ratio $(1+\epsilon)$ we will hit such an r. In any case, you can assume that the algorithm is run with a radius r satisfying (1).