Final exam, course on BWCA 2021

Uriel Feige

July 19, 2021

Answer all questions (that you know how to answer), and explain your answers clearly. Do the exam on your own. If any of the questions are not clear, do not hesitate to ask me for clarifications.

You are free to consult lecture notes and recordings for the course, as well as any outside sources (books, papers, web sites). However, please cite the sources that you use.

Hand in by Friday, July 23, 14:15, either physically at my mail box in the Zyskind building, or electronically by e-mail to uriel.feige@weizmann.ac.il.

1. Recall the 4-SAT problem (lecture notes from May 24). Consider the following problem. The input is a 4-CNF formula ϕ over n variables $x_1, \ldots, x_n$, an initial assignment A of truth values to the variables, and a parameter k. The desired output is an assignment A' that satisfies ϕ and is of Hamming distance at most k from A. (That is, the truth values that A and A' assign differ for at most k variables.) If no such assignment A' exists, then the output is *no*.

   (a) (Warm-up question.) Show that the problem is NP-hard.

   (b) Show that the problem is fixed parameter tractable, with k as the parameter.

2. Recall the knapsack problem (lecture notes from July 5) and the notion of perturbation resilience (Chapter 5 in BWCA book).

   For $\gamma > 1$, instance I' is a γ-perturbation of an instance I of the knapsack problem if the two instances differ only in profits of the items, and the profit of every item in I' is at least as much and not more than γ times larger than the profit of the same item in I (namely, $p_i \leq p'_i \leq \gamma p_i$). An instance I is γ-perturbation resilient if the optimal solution of I is unique, and every γ-perturbation of I has the same optimal solution as that of I (the items in the two optimal solutions are the same, but their profit may differ).

   For a given input instance I for the knapsack problem, let P denote the ratio between the profit of item with highest profit and the profit of item with smallest profit (namely, $P = \max_{i,j} \frac{p_i}{p_j}$).

   Design an algorithm for solving γ-perturbation resilient instances of the knapsack problem. The running time of your algorithm needs to be polynomial in n (the number of items), but may depend on γ and on P.

3. Recall the $G_{n,\frac{1}{2},k}$ planted clique model (lecture notes from June 7 and June 28). We saw two algorithms based on the ϑ function. One (based on $\lambda_1(M)$, see June 7) correctly returns the size of the planted clique (w.h.p., for sufficiently large k), whereas the solution of the other (based on $Tr(BJ)$, see June 28) is such that additionally, one easily sees which vertices belong to the planted clique.

   Here is a proposal of a constraint that can be added to the convex program for $\lambda_1(M)$, so that from its solution one can easily see which vertices belong to the planted clique. The new constraint is the last constraint in the following program.

   Minimize z subject to:

   - Matrix M is a symmetric matrix of order n.
   - $M_{ii} = 1$ for every i,
   - $M_{ij} = 1$ for every edge $(i,j) \in E$.
   - $\lambda_1(M) \leq z$.
   - $\lambda_1(M) + \lambda_2(M) \leq 2z - 1$.

   (a) Show that the optimum to the above program can be computed (up to arbitrary precision) by the ellipsoid algorithm in polynomial time.

   (b) Suppose that the size k of the planted clique satisfies $k \geq c\sqrt{n}$, where c is a sufficiently large constant (of your choice). Show that with high probability (over choice of input graph G), the optimal value of the above program satisfies $z = k$.

   (c) For an input instance G for which $z = k$, suppose for simplicity that one receives an exact solution to the above program (though if there is more than one exact solution, an adversary can choose which solution to give). Show how (no matter which exact solution the adversary chooses) one can determine from the solution matrix M which vertices are in the planted clique. The procedure for extracting the planted clique from M should run in polynomial time, and should use only M (without access to the input graph G).