College of Computer & Information Science | Fall 2014
Northeastern University | Handout 25
CS7800: Advanced Algorithms | 1 December 2014

Problems of the Week – 17 to 23

17. Maximum satisfiability

In the maximum satisfiability problem, we are given a set $\mathcal{C}$ of m clauses over n variables and the goal is to determine a boolean assignment to the n variables that maximizes the number of satisfied clauses. In class, we saw a randomized 7/8-approximation for the special case where there are 3 literals per clause, and a 1/2-approximation for the general case. Here, we consider an alternative algorithm.

Write the maximum satisfiability problem as an integer linear program by having an LP variable x_i for each boolean variable v_i and a linear constraint for each clause.

Consider the following randomized algorithm. Solve the linear relaxation for the above integer linear program to obtain solution x^*. Set each boolean variable v_i to 1 with probability equal to x_i^*. Show that the expected number of clauses satisfied is at least $\text{OPT} \cdot (1 - (1 - 1/q)^q)$, where q is the maximum number of literals in a clause and OPT is the optimal value.

18. Greedy online learning

In the online learning problem we studied in class, we have n experts, each giving a binary prediction for an event in each step. Suppose we consider a greedy procedure for combining their predictions: in each step, we select the prediction of the expert that has the highest correct predictions until that step; in case of ties, we select the expert with the smallest index.

Show that an adversary can ensure that the ratio of the number of mistakes made by the online learning algorithm to the number of mistakes made by the best expert can be made to be at least n.

19. Maximum cut with an additional constraint

In class, we studied two 1/2-approximation algorithms for the global maximum cut problem, the problem of finding a maximum-size cut in a given graph. One algorithm was a greedy algorithm, and the other a randomized algorithm.

Consider the maximum cut problem with the additional constraint that for two given sets S_1 and S_2 of pairs of vertices, we need to ensure that every pair of vertices in S_1 is on the same side of the cut, while every pair of vertices in S_2 is separated by the cut.

Extend the randomized algorithm to the above problem, and analyze its approximation ratio.

20. Trading currencies

You are embarking on a trip to Europe and would like to purchase the maximum number of Euros possible for D US Dollars. Given the rate at which dollar is sliding with respect to many currencies, you try to perform this exchange through a collection of trades through other currencies.

For instance, you may exchange some of your dollars for British pound, some of which you exchange for Japanese yen, which you convert to Euros.

You log on to your currency trading account on oanda.com with the goal of maximizing the number of Euros you can get for D US Dollars today. Oanda offers you a fixed rate r_{ij} for converting any currency i into any other currency j, but imposes a limit ℓ_{ij} on how much of currency i you can convert to currency j on a given day, for all ordered pairs (i, j). Assume that there are no arbitrage opportunities – that is, there is no directed cycle of currency trades whose product of rates exceeds 1. (Also assume that there are no other individual transaction fees.)

Give a polynomial-time algorithm to solve the above problem. Show that it is possible to achieve your objective without ever borrowing currency, and ending the day with no other currency except Euros and possibly US Dollars.

21. Cheapest short paths

Let $G = (V, E)$ be a directed graph, and $\ell : E \to N$ and $c : E \to N$ be two functions, where N is the set of positive integers. We call $\ell(e)$ to be the length of edge e and $c(e)$ to be the cost of edge e.

Given G, ℓ, c, a source $s \in V$, a destination $t \in V$, and an integer L, design an algorithm for finding a path from s to t with length at most L and whose cost is smallest among all paths from s to t of length at most L. (If no such path exists, then your algorithm should indicate so.) The worst-case running time of your algorithm must be polynomial in $|V|$, $|E|$, L, and the sizes of ℓ and c.

22. Hardness of independent set

Recall that an *independent set* of an undirected graph G is a subset V' of vertices such that no two vertices in V' have an edge in G. The IndependentSet problem is to find a maximum-size independent set in G. We know that IndependentSet is NP-complete. In this problem, we investigate the approximability of IndependentSet.

Define the *product* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ as $G = (V, E)$, where

$$V = \{\langle v_1, v_2\rangle : v_1 \in V_1, v_2 \in V_2\},$$

and there is an edge between vertices $\langle u_1, u_2\rangle$ and $\langle v_1, v_2\rangle$ in G if either $u_1 = v_1$ and $(u_2, v_2) \in E_2$ or $(u_1, v_1) \in E_1$. For a positive integer n, let G^n be defined by the recurrence relation $G^{i+1} = G^i \times G$ and $G^1 = G$.

(a) Prove that G has an independent set of size k if and only if G^t has an independent set of size k^t.

(b) Give a polynomial-time algorithm to construct an independent set of G of size $\lceil k^{1/t}\rceil$ from any independent set of G^t of size k.

(c) Using parts (a) and (b), argue that if there exists a constant c such that there is a polynomial-time c-approximation algorithm for IndependentSet, then one can obtain an arbitrarily good approximation – $(1 + \varepsilon)$-factor for $\varepsilon > 0$ arbitrarily small – for IndependentSet in polynomial time.

23. Scheduling with resumptions

Consider POW-15 in a new setting where jobs can be stopped and resumed (without losing any work). Thus, for example, a job that requires 5 processing units can be processed for 3 units, then stopped in favor of another job, and later resumed and processed for another 2 units leading to its completion. Here is the problem restated.

Given a collection of n jobs, labeled 1 through n, with job i released at time r_i and requiring processing time p_i units, we would like to determine if all of the jobs can be scheduled (with stopping and resumptions, as necessary) in such a way that they all complete within their deadlines.

Is this problem NP-complete? If yes, give a proof. If not, give a polynomial-time algorithm.