CSE 347 Analysis of Algorithms April 10, 2019

Lecture 21 Summary

Instructor: Brendan Juba

1. Surprisingly often, good approximation algorithms can be obtained by choosing a solution at random. We will see this on the following optimization variant of 3SAT, Max-Exact-3sat: given a 3CNF in which all clauses have *exactly* three literals, find an assignment that maximizes the number of satisfied clauses.

   (a) Note that it's easy to convert any 3CNF to one in the required form by adding three dummy variables that must be set to false in any satisfying assignment, and using these to pad out clauses containing less than three literals. Thus, Max-Exact-3sat is also NP-complete.

   (b) When we choose an assignment at random, it's immediate that $\frac{7}{8}$ of the clauses are satisfied. Since we can at best satisfy all of the clauses, if we can find an assignment that satisfies $\frac{7}{8}$ of them, this is a $\frac{7}{8}$-approximation.

   (c) Indeed, we can show that if there are m clauses, with probability at least $\frac{1}{8m}$ we must get such an assignment.

   (d) Thus, if we repeat $8m \ln \frac{1}{\delta}$ times, we will obtain such an assignment with probability $1 - \delta$. Alternatively, it's then also immediate if we simply keep picking assignments at random until we find one that satisfies $\frac{7}{8}$ of the clauses, this succeeds in $8m$ trials in expectation.

2. A standard, powerful approach to designing approximation algorithms uses reductions to optimization problems such as Linear Programming.

   (a) A linear program is the following optimization problem

      i. The candidate solutions are real (actually rational) vectors $(x_1, \ldots, x_n)$.

      ii. The constraints are linear inequalities, $\langle \mathbf{a}_1, \mathbf{x} \rangle \leq b_1, \ldots, \langle \mathbf{a}_m, \mathbf{x} \rangle \leq b_m$, where recall $\langle \mathbf{a}_j, \mathbf{x} \rangle = \sum_{i=1}^{n} a_{j,i} x_i$. (Each $\mathbf{a}_j$ is a rational vector.) We will frequently write this using matrix notation as $A\mathbf{x} \leq \mathbf{b}$.

      iii. The objective is $\langle \mathbf{c}, \mathbf{x} \rangle$ for some rational vector $\mathbf{c}$.

      iv. We can easily express $\geq$ constraints using $\langle -\mathbf{a}, \mathbf{x} \rangle \leq -b$. We can easily express equality constraints using a pair of inequalities.

   (b) Any flow network can be written as a linear program. This shows that linear programs are very expressive. But, what we give up is that we *cannot* promise to find integer solutions to our linear programs, unlike maximum flow problems. So, linear programming is much less useful for finding exact optimal solutions to discrete problems.

   (c) Linear programs always obtain an optimal solution at a vertex of the feasible region. A vertex is defined by n of the inequalities that are *"tight,"* i.e., satisfied with equality. Cramer's rule allows us to bound the size of such solutions, both in magnitude and in the number of bits needed.

(d) The *Simplex algorithm* starts at a vertex of the feasible region and chooses a neighboring vertex with better objective value until it finds a locally optimal solution. This algorithm is reasonably fast in practice, but has examples where it takes exponential time.

(e) The *Ellipsoid algorithm* solves linear programs in polynomial time, but it is very slow in practice. *Interior point methods* are reasonable to run in practice, but not especially fast.