CS602 Combinatorial Optimization via Linear and Convex Programming Jan-Apr 2022

Homework: LP and SDP rounding

Lecture 16 (Mar 7)

- Consider the following algorithm for the maximum weight satisfiability problem. Pick the highest weight clause. Set some variable in this clause so as to satisfy it. Then move to the second highest weight clause. If not already satisfied, then set some variable in this clause so as to satisfy it (if possible). And continue so on.

  Show that this algorithm can give a very bad (arbitrarily low) approximation factor.

- Consider a different algorithm for the maximum weight satisfiability problem. Find to total weight of clauses with x_1 and total weight of clauses with $\overline{x}_1$. Set x_1 depending on which weight is higher. Remove the already satisfied clauses. Then continue the process with x_2 and so on.

  Show that this algorithm always gives at least 1/2-approximation.

- Consider a randomized algorithm for the maximum weight satisfiability problem, where variables are set to True or False with some randomized scheme. Show that the expected total weight of satisfied clauses is

$$\sum_{C \in \text{Clauses}} w_j \times \Pr(\text{clause } C \text{ is satisfied}).$$

- Recall the LP we wrote for the maximum weight satisfiability problem with m clauses and n variables. Suppose w_j is the weight of the j-th clause. Let P_j and N_j be the sets of indices of positive and negative literals appearing in clause C_j.

$$\begin{aligned} \max & \sum_{j=1}^{m} w_j z_j \\ & \text{subject to} \\ & t_i, f_i \geq 0 \text{ for } 1 \leq i \leq n \\ & t_i + f_i = 1 \text{ for } 1 \leq i \leq n \\ & 0 \leq z_j \leq 1 \text{ for } 1 \leq j \leq m \\ z_j \leq & \sum_{i \in P_j} t_i + \sum_{i \in N_j} f_i \text{ for } 1 \leq j \leq m. \end{aligned}$$

  Suppose (t^*, f^*, z^*) is an optimal solution for this LP. We set our Boolean variables as follows. For each i, set $x_i = True$ if and only if $t^* \geq 1/2$.

  Show via the following example that this rounding scheme cannot give better than 1/2-approximation.

  $(x_1 \vee x_2), (x_2 \vee x_3), \ldots, (x_{2n} \vee x_{2n+1}), (x_{2n+1} \vee x_1), (\overline{x}_1 \vee \overline{x}_2), (\overline{x}_2 \vee \overline{x}_3), \ldots, (\overline{x}_{2n} \vee \overline{x}_{2n+1}), (\overline{x}_{2n+1} \vee \overline{x}_1)$

- Consider any randomized algorithm for the maximum weight satisfiability problem, where variables are set to True or False with some randomized scheme. Suppose we deterministically implement this algorithm using the method of conditional expectations (as described in the class). Show that the total weight of the satisfied clauses obtained from the deterministic implementation is at least as large as the expectation of the total weight of the satisfied clauses in the randomized algorithm.

Lecture 17 (Mar 10)

- Prove that $g(z) = 1 - (1 - z/\ell)^\ell$ is a concave function in the region $[0, 1]$, for $\ell = 1, 2, 3, \ldots$.
- Prove that $(1 - 1/2^\ell)/2 + (1 - (1 - 1/\ell)^\ell)/2 \geq 3/4$ for every $\ell = 1, 2, 3, 4, \ldots$.
- Prove that a combination of the two algorithms – simple randomized and LP rounding based – gives an approximation factor of 3/4.
- Write an LP for the Max-Cut problem and design a LP rounding based approximation algorithm with approximation factor 1/2.

Lecture 18 (Mar 14)

- Recall all the properties of PSD matrices from the class that were said to be equivalent to each other. Prove the equivalence (we only proved one direction in the class).
- Suppose P is a symmetric matrix that is not PSD. Design an algorithm to find a separating hyperplane between P and the PSD cone. That is, an inequality that is satisfied by all the PSD matrices M but not for P. More precisely, $\{a_{i,j}\}_{i,j}$ and b such that

$$\sum_{i,j} a_{i,j} M_{i,j} \geq b, \text{for all PSD matrices } M, \text{ but}$$

$$\sum_{i,j} a_{i,j} P_{i,j} < b.$$

Lecture 18 (Mar 14)

- Show that

$$\min_\theta \frac{\theta}{1 - \cos\theta} \approx 1.38.$$

- Design a SDP based approximation algorithm for MAX-2-SAT (each clause is OR of two literals).
- Write an SDP to find the maximum eigenvalue of a given symmetric matrix.
- Find an example where the gap between maxcut value and optimal value of the SDP (discussed in class) is roughly 0.878.