CSci570 Spring 2016

HW#12

Assigned: 4.22.2016
Solution posted also on 4.22.2016.

Problem 1. (Problem 5 of Exam III, Spring 2008)
A carpenter makes tables and bookshelves, and he wants to determine how many tables and bookshelves he should make each week for a maximum profit. The carpenter knows that a net profit for each table is \$25 and a net profit for each bookshelf is \$30. The wooden material available each week is 690 units; and his/her working hours are 120 hours per week. The estimated wood and working hours for making a table are 20 units and 5 hours respectively, while for making a bookshelf are 30 units and 4 hours. Formulate the problem using any technique we have covered in class. You do not need to solve it numerically.
We formulate the problem using linear programming
Suppose that the carpenter decides to make *x* tables and *y* bookshelves.

Based on the description in the problem, we want to maximize $25x + 30y$ subject to
$20x + 30y \leq 690$, and
$5x + 4y \leq 120$ where x, y are nonnegative integers.

Problem 2. (Problem 5 of Exam III, Spring 2010)
A furniture manufacturer makes two types of furniture: chairs and sofas. The production of the sofas and chairs requires three operations: carpentry, finishing, and upholstery. Manufacturing a chair requires 3 hours of carpentry, 9 hours of finishing, and 2 hours of upholstery. Manufacturing a sofa requires 2 hours of carpentry, 4 hours of finishing, and 10 hours of upholstery. The factory has allocated at most 66 labor hours for carpentry, 180 labor hours for finishing, and 200 labor hours for upholstery. The profit per chair is \$90 and the profit per sofa is \$75. The goal is to determine how many chairs and how many sofas should be produced each day to maximize the profit. Formulate the problem as a linear programming problem and then solve the problem numerically.
If *x* is the number of chairs produced and *y* is the number of sofas produced we will want to maximize $P=90x+75y$ subject to the following constraints:
$3x + 2y \leq 66$,
$9x + 4y \leq 180$, and
$2x + 10y \leq 200$ where x, y are nonnegative integers.

The corner points are (0, 0), (0,20), (10, 18), (16, 9), and (20, 0) where (10, 18) maximizes the profit (the maximized profit is 2250).

Problem 3. (Problem 4 of Exam III, Spring 2010)
If there is a polynomial-time 2-approximation algorithm for the weighted vertex cover problem then there is a polynomial-time 1/2-approximation algorithm for the Independent Set problem. Prove or disprove the statement.
The statement is false.

Let $G = (V, E)$ be the graph on which the independent set and the vertex cover problem is to be solved. A set is independent if and only if its complement is a vertex cover. That is, there exists an independent set of size k in G if and only if there exists a vertex cover of size $|V| - k$. For the weighted version, that is, there exists an independent set of weight k in G if and only if there exists a vertex cover of weight $\Sigma_v weight(v) - k$.

Consider an ε-approximation algorithm for weighted vertex cover where ε is a constant. If there actually exists an independent set of weight k in G, then the approximation algorithm only proves the existence of a vertex cover of weight $\varepsilon(\Sigma_v weight(v) - k)$. This only implies the existence of an independent set of weight $\Sigma_v weight(v) - \varepsilon(\Sigma_v weight(v) - k) = (1 - \varepsilon) \Sigma_v weight(v) + \varepsilon k$, which is not a constant approximation. (Let ε be 2. The statement is true only if $3k$ is not smaller than $2\Sigma_v weight(v)$.)

Problem 4. (Problem 6 of Exam III, Spring 2008)
A variation of the satisfiability problem is the MIN-2-SAT problem. The goal in the MIN-2-SAT problem is to find a truth assignment that minimizes the number of satisfied clauses. Give the best approximation algorithm that you can find for the problem.
We assume that no clause contains a variable as well the complement of that variable. (Such clauses are satisfied regardless of the truth assignment used.)

We give a 2-approximation algorithm as follows:

Let I be an instance of MINSAT consisting of the clause set C_I and variable set X_I. Construct an auxiliary graph $G_I(V_I, E_I)$ corresponding to I as follows. The node set V_I is in one-to-one correspondence with the clause set C_I. For any two nodes v_i and v_j in V_I, the edge (v_i, v_j) is in E_I if and only if the corresponding clauses c_i and c_j are such that there is a variable x belongs to X_I that appears in uncomplemented form in c_i and complemented form in c_j, or vice versa.

To construct a truth assignment, we construct an approximate vertex cover V' for G_I such that $|V'|$ is at most twice that of a minimum vertex cover for G_I. Then, construct a truth assignment that causes all clauses in $V_I - V'$ to be false. (For a method to find an approximate vertex cover, please refer to section 11.4 in the textbook.)

Problem 5.
Given as input a 3-CNF formula with n variables and m clauses, the MAX-SAT problem (optimization version) is to find a truth assignment to the variables that maximizes the number of clauses satisfied. Design a simple 2-approximation to the MAX-SAT problem (that is, your truth assignment should satisfy at least half as many clauses as an optimal assignment).
Start with an arbitrary truth assignment and compute the number clauses it satisfies. If at least half the clauses are satisfied, we are done. Otherwise, invert the truth assignment (all the clauses not satisfied in the original assignment will be satisfied by the inverted assignment).

Problem 6.
A Max-Cut of an undirected graph $G = (V, E)$ is defined as a cut C_{max} such that the number of edges crossing C_{max} is the maximum possible among all cuts. Consider the following algorithm. (i) Start with an arbitrary cut C. (ii) While there exists a vertex v such that moving v from one side of C to the other increases the number of edges crossing C, move v and update C.
(a) Does the algorithm terminate in time polynomial in $|V|$?
(b) Prove that the algorithm is a 2-approximation; that is, the number of edges crossing C_{max} is at most twice the number crossing C.
(a) Yes, step (ii) takes at most $O(|V|)$ time and is repeated at most $|E|$ times since at each iteration the number of edges crossing increases by at least 1.

(b) When the algorithm terminates, for every vertex v, at least half of its neighbors is on the other side (Otherwise we can move v). This implies that the number of edges crossing C is at least $|E|/2$. Since there are at most $|E|$ edges, an optimal solution can have at most $|E|$ edges crossing the cut.

Problem 7. Kleinberg & Tardos, Chapter 11, Exercise 1.

a) An example would be to have K = 10, and have items of weights $w_1 = 6$, $w_2 = 6$, $w_3 = 4$, and $w_4 = 4$. Then, we could pack items 1 and 3 on one truck, and 2 and 4 on another, for a total of two trucks. Instead, the greedy algorithm packs item 1 on truck by itself, then puts 2 and 3 on another truck, and finally puts item 4 on a truck by itself. Thus, it uses 3 trucks.
b) We let l_j denote the load of the *jth* truck, i.e., the total weight of all items that were put on truck *j*. Then, we always have that $l_j + l_{j+1} > K$. For otherwise, the first item that ended up on truck $j + 1$ would have been put on truck j by the algorithm, as it would still have fir there. Let m be the number of trucks used by the algorithm, and $L = \Sigma_i w_i = \Sigma_j l_j$ be the total load. We can now rewrite
$$L = \sum_j l_j \geq \sum_{j=1}^{\lfloor m/2 \rfloor} (l_{2j-1} + l_{2j}) > \sum_{j=1}^{\lfloor m/2 \rfloor} K > \lfloor m/2 \rfloor \cdot K$$
As the total load is thus greater than $\lfloor m/2 \cdot K \rfloor$, even the optimum solution will have to use at least $\lfloor m/2 \rfloor + 1$ trucks of volume K, i.e., at least half as many as our solution.

Problem 8. Kleinberg & Tardos, Chapter 11, Exercise 9.

Algorithm

```
Start with M=Φ and R=T
while R≠Φ do
    Let $t_i = (x_i, y_i, z_i)$ be an arbitrary triple in R.
    Add $t_i$ to M.
    Remove from R all triples $t_j$ intersecint $t_i$.
end while
```

The algorithm obviously runs in polynomial time $O(|T|^2)$. We will show that any triple chosen by our algorithm can intersect at most three triples in the optimum solution, so the optimum solution contains at most three times as many triples.

If $t_i = (x_i, y_i, z_i)$ is a triple chosen by our algorithm, and four triples, t_1, t_2, t_3, t_4 in the optimum solution intersected it, then each of those four has to include either x_i, y_i, or z_i. But then, two of them must include the same element, which is impossible, as they do not intersect. If we let S^* denote the optimum solution, and S the solution for our algorithm, we can map each triple $i \in S^*$ to an $h(i) \in S$ such that $t_{h(i)}$ intersects t_i. (If there are multiple triples it intersect, we let h(i) be any one of them. There must always be at least one, as otherwise, our algorithm would not have terminated, but rather included t_i as well.) Then, the above argument shows that

$|S^*| \leq 3|\{h(i)| \ i \in S^*\}| \leq 3|S|$,

i.e., our algorithm is a 1/3-approximation.