236521 Approximation Algorithms	**1st Semester, 1995–6**
Lecture 3: 13.11.95	
Lecturer: Yuval Rabani	*Scribe: David Carmel*

1 Relaxation

In this lecture we will show some examples of how to use relaxation methods in order to develop and analyze some approximation algorithms. The main idea is to describe the 'hard' problems as an Integer Linear Programs (ILP), and to solve a an 'easy' version the program by relaxing the constraints of the original problem.

Lets look at a standard optimization problem:

$$OPT = \min_{y \in S(x)} f(y) \tag{1}$$

where $S(x)$ is the set of the feasible solutions for a given problem.

Definition 1 *Lets look at another problem:*

$$LB = \min_{y \in R(x)} g(y) \tag{2}$$

LB will be called a relaxation of OPT iff

1. $\forall x, S(x) \subseteq R(x)$
2. $\forall y \in S(x), g(y) \leq f(y)$

It follows immediately from the definition that a solution to the *relaxation* problem provides a lower bound for the original problem, $LB \leq OPT$. The next sections show some examples of how to approximate NP-Hard problems by such methods.

2 The Vertex Cover Problem (VCP)

The input for *VCP* is a graph $G = (V, E)$ and a weight function on the vertexes $w : V \rightarrow \Re^+ \cup \{0\}$. A set of vertexes $U \subseteq V$ will be called a *vertex cover* for G iff any edge in E is incident to at least one vertex in U.

$w(U) = \sum\limits_{v \in U} w(v)$ will be called the weight of the *vertex cover* U.

The optimization problem is defined as follow: find a minimal *vertex cover* for G, i.e. a *vertex cover* with a minimum weight.

This problem is known as NP-Complete. We shall describe it by Integer Linear Program (ILP):

$$\begin{array}{ll} ILP = \min \sum_{v \in V} w(v)\chi_v & \text{s.t.} \\ (1) \forall (u,v) \in E & \chi_u + \chi_v \geq 1 \\ (2) \forall v \in V & \chi_v \in \{0,1\} \end{array} \tag{3}$$

A solution to VCP can be realized from a solution for the ILP: v belongs to a minimal *vertex cover* of the graph iff $\chi_v = 1$.

A natural *relaxation* to the problem is to relax the constraints $\chi_v \in \{0,1\}$ to $0 \leq \chi_v \leq 1$. The relaxation problem is a standard Linear Program (SLP) that can be solved efficiently. A *vertex cover* for the graph can be realized from the solution of the SLP:

$$U = \{v \in V | \chi_v \geq \frac{1}{2}\} \tag{4}$$

Following constraint 1, U is a *vertex cover* for G. Any edge has at least one incident vertex with a weight of at least $\frac{1}{2}$, hence, any edge has at least one incident vertex in U. Furthermore, the weight of U is bounded by at most twice of the optimal solution:

$$\sum_{v \in U} w(v) \leq 2 \sum_{v \in V} \chi_v w(v) = 2 \cdot SLP \leq 2 \cdot ILP = 2 \cdot OPT \tag{5}$$

How 'good' the approximation solution is?
Lets look at the complete graph K_n with a weight function that associate a weight of 1 for all vertexes of the graph. It is easy to show that any optimal solution for any graph with such a weight function must have a weight of at most of $\frac{n}{2}$. The weight of the minimal *vertex cover* for K_n is at most $n-1$ because any *vertex cover* must include at most $n-1$ vertexes, (otherwise, if two nodes, u, v are not in U, the edge (u,v) is not covered by U). Therefore, the best approximation which can be achieved is $\frac{n-1}{n/2} = 2 - \frac{2}{n}$.

2.1 A linear approximation for VCP

In the previous method we must solve a general Linear Program, a solution that is not practical for many applications. In this section we will show a linear approximation algorithm for VCP.

Lets look on the dual program:

$$\begin{array}{ll} D = \max \sum_{(u,v) \in E} y_{u,v} & \text{s.t.} \\ 1) \forall u \in V & \sum_{v | (u,v) \in E} y_{u,v} \leq w(u) \\ 2) \forall u, v \in V & y_{u,v} \geq 0 \end{array} \tag{6}$$

The main idea is to find in parallel feasible solutions for the primal and the dual LP problems. According to the *weak duality* property of Linear Programming, if the 'distance' between the two solutions is not too high, both solutions are 'close' to the optimal one. The next algorithm returns a 2-approximation for VCP.

```
Algorithm: Linear approximation for VCP
  Init:
    U ← ∅
    ȳ ← 0̄
    F ← E
  Loop:
    while F ≠ ∅ do
      1. choose (u, v) ∈ F
      2. increase y_{u,v} until dual constraint 1 becomes tight for either u or v
      3. w.l.o.g. assume that w(u) becomes tight,
         add u to U and remove all edges incident to u from F.
  Return U
```

Theorem 1 *The algorithm returns a vertex cover U for G, and the solution is a 2-approximation for VCP.*

Proof. The first claim is based on the property that only covered edges are taken out from F. The second claim is followed immediately from weak duality of linear programming:

$$w(U) = \sum_{u \in U} w(u) = \sum_{u \in U} \sum_{v|(u,v) \in E} y_{u,v} = \sum_{(u,v) \in E} |U \cap \{u,v\}| y_{u,v} \leq 2 \sum_{(u,v) \in E} y_{u,v} \leq 2 \cdot LP \leq 2 \cdot OPT \tag{7}$$

□

3 The Set Cover Problem (SCP)

The problem:

Given a set of elements $S = \{1, 2, \ldots, n\}$, and a collection of subsets of S, $S_1, S_2, \ldots, S_m$. Find a minimal collection of subsets such that their union covers S.

We shall represent the problem by a hyper-graph $H = (V, E)$ where $V = \{S_1, S_2, \ldots, S_m\}$ and $E = S$. In a hyper-graph each edge is incident to a set of vertexes, possibly more then two. For the given hyper-graph, H, each edge $i \in E$ is incident to all the sets which it belongs to. We shall mark d_v as the number of edges that are incident to vertex v, (the number of elements that belongs to the set v). The maximal degree of the hyper-graph, $d = \max_v d_v$.

The problem can be represented as an *Integer Linear Program*:

$$\begin{array}{lll} \tau(H) = \min_v \chi_v & \text{s.t.} & \\ \forall e \in E & \sum_{v|e \in v} \chi_v \geq 1 & \\ \forall v \in V & \chi_v \in \{0,1\} & \end{array} \tag{8}$$

We shall show an $O(\log d)$-approximation for SCP. The following greedy algorithm returns a *set cover* for SCP represented by a hyper-graph $H = (V, E)$.

```
Greedy Algorithm for SCP
  C ← ∅
  While E ≠ ∅
    pick v ∈ V with maximal degree in the current hyper-graph
    C ← C ∪ {v}
    E ← E − {e|e ∈ v}
  Rerurn C
```

Theorem 2 *The greedy algorithm is a $O(\log d)$-approximation for SCP.*

First, we shall look on a relaxation of the problem by changing the constraint $\chi_v \in \{0,1\}$ to $\chi_v \geq 0$. We shall mark the solution for the relaxed problem $\tau^*(H)$. $\tau^*(H)$ is a relaxation of $\tau(H)$. Therefore, $\tau^*(H) \leq \tau(H)$.

Definition 2 *A matching in a hyper-graph is a set of independent edges, (a set of edges in which the intersection of any pair of edges in the set is empty).*
k-matching in a hyper-graph is a set of edges such that each vertex of the graph is incident to at most k edges.
$m_k(H)$ is the maximal k-matching of H.

For proving the theorem we have to prove first the following lemmas.

Lemma 3 $\forall k, m_k(H) \leq k\tau^*(H)$

Proof. The solution for $\tau^*(H)$ satisfies the constraints $\forall e, \sum_{v|e\in v} \chi_v \geq 1$. Therefore, if M is a maximal k-matching for H,

$$\sum_{e\in M} \sum_{v|e\in v} \chi_v \geq m_k(H) \tag{9}$$

Any vertex is incident to at most k different edges hence χ_v is counted at most k times. Therefore:

$$\tau^*(H) = \sum_v \chi_v \geq \frac{1}{k} \sum_{e\in M} \sum_{v|e\in v} \chi_v \geq \frac{1}{k} m_k(H) \tag{10}$$

□

Assume that the *set cover* returned by the algorithm is $C = \{v_1, v_2, \ldots, v_t\}$.

Lemma 4

$$t \leq \sum_{i=1}^{d-1} \frac{m_i(H)}{i(i+1)} + \frac{m_d(H)}{d} \tag{11}$$

Proof. At each stage the algorithm picks a vertex in the current graph with the maximal degree, hence v_1 has a degree of d and $|E| = m_d$. We shall define the *cover degree* of a vertex as the number of edges incident to that vertex when it is picked by the algorithm. t_r is the number of vertexes with a *cover degree* of r. Following these definitions:

$$\begin{aligned} t &= t_d + t_{d-1} + \ldots, +t_1 \\ |E| &= dt_d + (d-1)t_{d-1} + \ldots, +t_1 \end{aligned} \tag{12}$$

The second equation is followed from the property that any vertex picked by the algorithm with a *cover degree* of t_i, covers i new edges.

We shall define a hyper-graph $H_i = (V, E_i)$ as the modified hyper-graph accepted after taking out $\sum_{j=i+1}^{d} t_j$ vertexes, during the first stages of the algorithm. The maximal degree of H_i is less than $i+1$ because all the sets that cover at least $i+1$ edges were picked already. Therefore, E_i is a i-matching for the original hyper-graph H.

$$m_i(H) \geq |E_i| = it_i + (i-1)t_{i-1} + \ldots, +t_1 \tag{13}$$

Using these inequalities, we get

$$\begin{aligned} &\tfrac{m_1(H)}{1\cdot 2} + \tfrac{m_2(H)}{2\cdot 3} +, \ldots, + \tfrac{m_{d-1}(H)}{(d-1)d} + \tfrac{m_d(H)}{d} \geq \\ &\tfrac{t_1}{1\cdot 2} + \tfrac{2t_2+t_1}{2\cdot 3}, \ldots, + \tfrac{(d-1)t_{d-1}+\ldots,t_1}{(d-1)d} + \tfrac{dt_d+\ldots,t_1}{d} \end{aligned} \tag{14}$$

All coefficients of t_i in the right side of Equation 14 are summed to 1.

$$\forall t_i : \sum_{j=i}^{d-1} \frac{i}{j(j+1)} + \frac{i}{d} = 1 \tag{15}$$

Therefore:

$$\frac{m_1(H)}{1\cdot 2} + \frac{m_2(H)}{2\cdot 3} +, \ldots, + \frac{m_{d-1}(H)}{(d-1)d} + \frac{m_d(H)}{d} \geq \sum_{i=1}^{d} t_i = t \tag{16}$$

□

Now we can show that the greedy algorithm is a $O(\log d)$-approximation for the SCP.

Proof of Theorem 2. Substituting the results of Lemma 3 into the result of Lemma 4 we get

$$t \leq \sum_{i=1}^{d-1} \frac{m_i(H)}{i(i+1)} + \frac{m_d(H)}{d} \leq \sum_{i=1}^{d-1} \frac{i\tau^*(H)}{i(i+1)} + \frac{d\tau^*(H)}{d} = \tau^*(H) \sum_{i=1}^{d} \frac{1}{i} = \tau^*(H) H_d \approx \tau^*(H)(1+\log d) \tag{17}$$

$\tau^*(H)$ is a relaxation of SCP therefore $t \leq (1 + \log d)\tau(H)$ and this complete the proof. □