CSC2411 - Linear Programming and Combinatorial Optimization*
Lecture 10: Integrality Gap, Rounding

Notes taken by Gerald Quon

April 9, 2007

Summary: In this lecture, we review the properties of Totally Unimodular Matrices (TUM), and prove that the incidence matrix of directed and bipartite graphs is TUM. We then discuss the Maximum Matching problem as a dual problem of the Vertex Cover problem. We then define Integrality-Gap, and prove that for vertex cover, we attain an integrality gap of 2. We then discuss alternative rounding strategies besides naive rounding, and apply this to the Set Cover problem. We end with an introduction to the maximum satisfiability (MAX-SAT) problem.

1 Totally Unimodular Matrices

In the previous lecture, we were introduced to Totally Unimodular Matrices (TUMs), that is, matrices A for which every square submatrix M has $det(M) \in \{+1, -1, 0\}$, which also leads to the fact that each such matrix A can only have +1, -1, or 0 as its elements. Furthermore, we pointed out that if A is TUM, then the vertices of the polytope of $Ax = b, x \geq 0$ are integral.

We now prove several theorems that relate TUM incidence matrices to various graphs.

Claim 1.1. *If H is a directed graph, then its incidence matrix is TUM.*

Proof. Consider any submatrix M, and its first column - there are three possibilities. If there exists a 1 in a column, then the determinant of M is the the determinant of the reminaing matrix. If there is all zeros, then the matrix is singular and its determinant is 0. Else, if a column has a 1 and a -1, the matrix is singular because the sum of rows is zero in that column.

We now use Claim 1.1 to show the following.

Claim 1.2. *If G is a bipartite graph, its incidence matrix is TUM.*

* Lecture Notes for a course given by Avner Magen, Dept. of Computer Sciecne, University of Toronto.

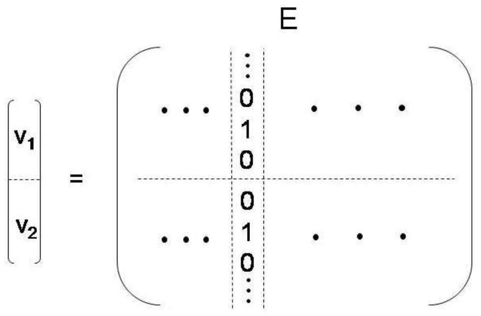


Figure 1: The incidence matrix of a directed, bipartite graph $H = (V, E)$, where V_1 is one side, and V_2 is the other side, and the dges are directed from V_1 to V_2. Rows represent vertices in H, and columns represent its edges. $uv = e \in E \Rightarrow A_{ue} = +1, A_{ve} = -1$.

Proof. Take G, and construct a new directed graph H with an incidence matrix as shown in Figure 1. Then, apply Claim 1.1 on H. To see that the result applies to the original matrix G, note that the incidence matrix of H is constructed by flipping the sign of some entries of G (see Figure 2). Hence, the determinent will only scale by a multiplication factor of 1 or -1, but wil still be either zero, +1, or -1. In other words, we can view a bipartite graph as a directed graph G, and use Claim 1.1 to get that $det(E_G) = \pm det(E_H)$.

Remark 1.3. Note that if G is not bipartite, then E_G is not TUM. Consider a matrix G that contains an odd cycle formed by vertices $x_{C_1}, x_{C_2}, ..., x_{C_m}$, and consider the submatrix of G, M, formed by the column and rows $C_1, C_2, ..., C_m$. $det(M) = 2$, therefore M is not TUM.

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ & & \ddots & & & \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

2 Maximum Matching Problem

The Maximum Matching Problem (MM), intuitively, is to find a set of non-adjacent edges whose total weight is maximal, where non-adjacent edges implies that no two edges share a common vertex. Here we will show that the LP relaxed version of MM

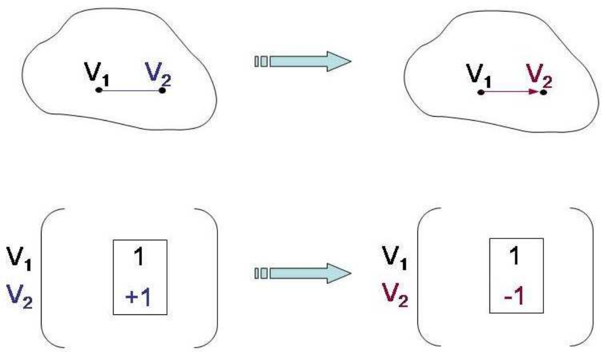


Figure 2: The incidence matrix of an original bipartite graph G is modified to obtain an incidence matrix of a directed graph H.

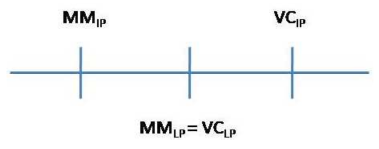


Figure 3: The relationship between the optimal values of the relaxed (LP) and exact (IP) integer programming problems of VC and MM. Note $MM_{IP} = MM_{LP} = VC_{LP} = VC_{IP}$ for bipartite graphs, according to our Claim that for bipartite graphs, the LP relaxation is exact.

is like a relaxation of the dual of another Integer Programming (IP) problem. In this case, Vertex Cover (VC) is the dual of MM.

Problem VC-Relaxed

$$\min \sum_{v \in V} y_v \text{ s.t.}$$
$$y_u + y_v \geq 1, uv \in E$$
$$y_u \geq 0$$

Problem MM-Relaxed

$$\max \sum_{e \in E} x_e \text{ s.t.}$$
$$\sum_{\text{e touches v}} x_e \leq 1, \forall v, x_e \in \{0, 1\}$$
$$x_e \geq 0$$

In general, the relationship between the optimal values of the relaxed and exact solutions of MM and VC are related by the Figure 3, where all of the optimal values are equal for a bipartite graph.

3 Integrality Gap

In this section we will discuss the notion of Integrality Gap, as well as rounding procedures for turning LP relaxation solutions into integer solutions.

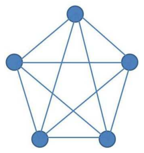


Figure 4: K_n for $n = 5$.

Rounding is a method of taking a fractional solution and mapping it to an integral solution. For VC, it is easy to see that in general, the LP relaxation is not exact. For example, take $G = K_n$ (complete graph on n vertices - see Figure 4). The best integral solution is $n - 1$ since if 2 nodes are left out, there is one edge between them that is missed. One possible fractional solution, when all $x_i = \frac{1}{2}$, is $\frac{1}{2}n = \frac{n}{2}$. Hence, the integrality gap is at least $\frac{n-1}{\frac{n}{2}} = 2 - O(1)$.

Claim 3.1. *The integrality gap of LP relaxation for VC is* ≤ 2.

Proof. Note in general the easiest way to solve these types of proofs is to take a feasible solution, 'round' to an integer solution, and see the change in the objective function. Do not try to start with an optimal integer solution. So let $y = (y_1, y_2, ..., y_n)$ be a feasible solution to the LP relaxation of VC. Now define $\forall y_i$,

$$z_i = 1 \text{ if } y_i \geq 0.5$$
$$z_i = 0 \text{ if } y_i < 0.5$$

So $y \overset{rounding}{\longrightarrow} z \in \{0,1\}^n$. We now must check that z is a valid solution.

$$ij \in E \Rightarrow y_i + y_j \geq 1$$
$$\Rightarrow y_i \geq \tfrac{1}{2} \text{ or } y_j \geq \tfrac{1}{2}$$
$$\Rightarrow z_i = 1 \text{ or } z_j = 1$$
$$\text{So}$$
$$\text{Integrality-Gap}=sup\frac{OPT(IP)}{OPT(LP)} = \frac{\sum z_i}{\sum y_i}.$$
$$\text{But } \sum z_i = \sum_{\{i|y_i \geq \frac{1}{2}\}} 1 \leq 2 \sum_{\{i|y_i \geq \frac{1}{2}\}} y_i \leq 2 \sum y_i$$
$$\Rightarrow \frac{\sum z_i}{\sum y_i} \leq 2$$
$$\Rightarrow \text{Integrality-Gap} \leq 2.$$

Figure 5 shows the relationship between the optimal solutions (of both the integer programming, OPT, and the LP relaxed problem, OPT^*), as well as the integrality gap. Note that for a particular instance, the rounding factor is the (Approximation Factor) x (Integrality Gap). So, bounding the rounding factor bounds the integrality gap.

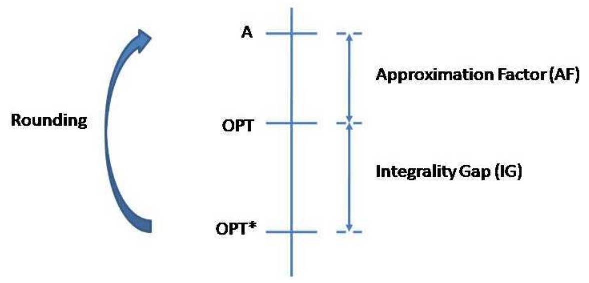
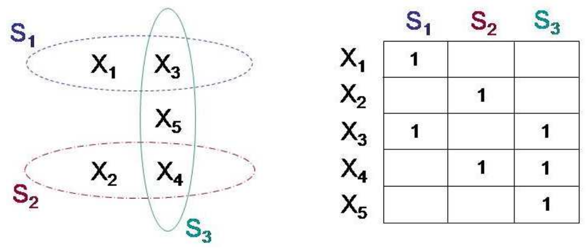


Figure 5: The relationship between the optimal solution to the LP relaxed minimization problem, OPT^*, the optimal solution of the integer program, OPT, and the integer solution achieved by rounding OPT^*, A.



	S_1	S_2	S_3
X_1	1		
X_2		1	
X_3	1		1
X_4		1	1
X_5			1

Figure 6: The matrix A in $Ax = b$, for the SCP LP-relaxed problem. $A_{ij} = 1$ iff element $i \in S_j$. Hence, the rows of A correspond to elements, and columns j correspond to sets S_j.

4 Set-Cover Problem

In this section, we show how to relax the Set-Cover Problem (SCP) and achieve an approximation factor of $4(\log(n) + 2)$, where SCP is defined below.

Input: U is a universe of elements, and $S_1, S_2, ..., S_m \subseteq U$, where $|U| = n$ (U has n elements).

Output: A minimal subfamily of sets so that every element is covered in at least one of the sets in the subfamily.

A practical example of this problem might be in hardware circuit design - we have expensive tests, which each test different registers and variables, and our goal is to test every component at least once, while minimizing the cost of the tests.

Note feasibility is easy to check - simply check if $S_1 \cup S_2 \cup ... \cup S_n$ covers all variables.

We can formulate the LP relaxation of SCP as follows. Let $A_{ij} = 1$ iff element $i \in S_j$ (See Figure 6). Define the corresponding $x_j \in \{0, 1\}$, where $x_j = 1$ iff x_j is in set S_j. We can formally define SCP as:

$$\min \sum x_j$$
$$\text{s.t. } Ax \geq \vec{1}$$
$$x_j \in \{0,1\}$$

We can relax it to an LP formulation as follows:

$$\min \sum x_j$$
$$\text{s.t. } Ax \geq \vec{1}$$
$$x_j \geq 0$$

Let's now consider possible rounding procedures to take a solution to *SCP-relaxed* and translate it into an integer solution for *SCP*. Consider a randomized procedure that picks a set S_j with probability x_j. The expected number of sets is $\sum x_i$, which is the cost of the linear program, which should be at most the cost of the optimal original integer program! However, the solution may not be feasible!

So instead, we pick a set S_j with probability x_j, and repeat this S independent times, and take all sets that were picked at least once in all t iterations. How big a t do we need to ensure that all elements are covered? Recall that OPT is the optimal solution of the original integer programming problem, and OPT^* is the optimal solution of the LP relaxed problem (see Figure 5).

For element i, $\sum_{i \in S_j} x_j = (Ax)_i \geq 1$. Therefore,

Pr[i is not chosen in one iteration] = $\prod_{i \in S_j} (1 - x_j) \leq \prod_{i \in S_j} e^{-x_j} = e^{-\sum_{i \in S_j} x_j} \leq e^{-1}$.

Hence, Pr[i not picked in any of the t iterations] $\leq (e^{-1})^t = e^{-t}$. This implies Pr[any element not picked] $\leq ne^{-t}$ [since at most, events are independent and probabilities add]

So set $t = \log(n) + 2$, then $ne^{-t} \leq \frac{1}{4}$.

Let $y_i \in \{0,1\}$ be the indicator variable for the algorithm's choice ($y_i = 1$ if picked). Then

$E[\sum y_i] \leq t \sum x_j \leq t * OPT^* = (\log(n) + 2) * OPT^*$

Therefore Pr[$\sum y_i \geq 4(\log(n) + 2) \cdot OPT^*$] $\leq \frac{1}{4}$ (by Markov inequality)

To summarize, with probability $\geq 1 - \frac{1}{4} - \frac{1}{4} = \frac{1}{2}$ we get a valid solution, with objective value $\leq 4(\log(n) + 2) \cdot OPT^* \leq 4(\log(n) + 2) \cdot OPT$.

So we get an algorithm with approximation factor $4(\log(n) + 2)$.

5 Maximum Satisfiability

Maximum satisfiability problems are of the form, given n variables $x_i \in \{0,1\}$, $i = 1, ..., n$, and clauses C_j, then MAX-SAT asks to determine a binary configuration of all x_i such that we maximize the sum of the satisfied clauses.

An example is the following:

Let

$$C_1 = (x_1 \vee x_2 \vee \bar{x_5} \vee x_6)$$

$$C_2 = (x_2 \vee \bar{x_1})$$
$$C_3 = (x_2 \vee \bar{x_1})$$

Then the problem is $C_1 \wedge C_2 \wedge C_3$.

Note that $C_1 \Rightarrow x_1 + x_2 + (1 - x_5) + x_6 \geq 1$.

MAX-SAT can be reformulated as an integer programming problem:

$$\max \sum z_{c_j} \text{ s.t.}$$
$$\sum_{i \in c_j^+} x_i + \sum_{i \in c_j^-} (1 - x_i) \geq z_{c_j}, \forall j$$
$$x_i, z_{c_j} \in \{0, 1\}$$

We can then relax to LP:

$$\max \sum z_{c_j} \sum_{i \in c_j^+} x_i + \sum_{i \in c_j^-} (1 - x_i) \geq z_{c_j}, \forall j$$
$$z_{c_j} \leq 1$$
$$x_i \geq 0$$