CSC 373 H 1 Y — Summer 2007

University of Toronto — St. George Campus

Lecture Summary for Week 12

This summary is not a replacement for the lecture. If you miss a class, please arrange with a friend to take note for you.

6 APPROXIMATION ALGORITHMS [11.1, 11.6, CLRS 35.1]

For many problems finding an exact (or optimal) solution may be expensive while solutions that are "good enough" might be easier to find. An algorithm for a maximization problem is said to have approximation ratio of $\rho(n)$ if on an input of length n, the output of the algorithm has value at least $opt/\rho(n)$, where opt is the optimal value for that input. Similarly, an algorithm for a minimization problem has approximation ratio of $\rho(n)$ if its output has value at most $\rho(n)opt$, where opt is the optimal value.

Note that $\rho(n) \geq 1$, and we will try to design algorithms that achieve $\rho(n)$ as small as possible.

Approximation algorithms can be designed using the techniques that we have learnt so far. Here we will focus on the greedy and linear programming techniques. We start with a simple algorithm for the Vertex Cover problem that you have seen in the test.

6.1 Vertex Cover Problem [CLRS 35.1]

Given a graph $G = (V, E)$. Recall that the Vertex Cover problem is to find minimum-size subset $V' \subseteq V$ that "covers" E, i.e., every edge in E has at least one endpoint in V'.

This problem is NP-hard, so there might be a polytime algorithm for it. Consider the following algorithm that has the approximation ratio of 2. The idea is to choose an edge that has not been covered and take both of its endpoints.

1. $C \leftarrow \varnothing$ % C is the output
2. $E' \leftarrow E$ % E' is the set of edges not covered by C
3. While $E' \neq \varnothing$ do
4. Let $e = (u, v)$ be an edge in E'
5. $C \leftarrow C \cup \{u, v\}$
6. Remove from E' all edges incident on u, v.
7. End While
8. Return C

Clearly the output of the algorithm is a vertex cover for G. In spite of its simplicity, the algorithm has approximation ratio of 2:

Theorem: The above algorithm has approximation ratio of 2.

Let A be the set of edges selected by the algorithm in line 4. Then the edges in A do not share any endpoint, and C is the set of endpoints of edges in A. Therefore to cover the edges in A we need at least one endpoint of each edge. Consequently the size of any cover of G must be at least $|A| = |C|/2$. As a result, $|C| \leq 2|OPT|$ where OPT is any minimum-size vertex cover of G. □

Note that we proved the above theorem without knowing the exact size of a minimum vertex cover.

Another interesting issue is whether the constant 2 can be improved. In other words, can the algorithm have an approximation ratio of c for some constant $c < 2$? The answer is NO, since there is a family of graphs for which the algorithm produces vertex covers of size precisely twice the smallest size. The graphs in this family consist of disjoint edges.

6.2 Load Balancing [11.1]

Suppose that there are m machines $M_1, M_2, \ldots, M_m$ each can process only one job at a time. Given a set of n jobs where job j has processing time t_j. We wish to assign each job to a machine so that the "loads" placed on the machines are as "balanced" as possible.

Here the load placed on a machine M_i is the total processing time of all jobs assigned to it:

$$T_i = \sum_{j \in A(i)} t_j$$

where $A(i)$ is the set of all jobs assigned to machine M_i. The problem is to minimize

$$max\{T_i \ : \ 1 \leq i \leq m\}$$

($max\{T_i \ : \ 1 \leq i \leq m\}$ is also called the *makespan* of the job assignment.)

Example: There are 3 machines M_1, M_2, M_3 and 6 jobs with processing time

$$t_1 = 2, t_2 = 3, t_3 = 4, t_4 = 6, t_5 = 2, t_6 = 2$$

Suppose that we assign jobs $2, 4$ to M_1, 1, 5, 6 to M_2 and 3 to M_3. Then $T_1 = 9, T_2 = 6, T_3 = 3$ and the makespan is 9.

The problem is NP-hard. Here we will consider two slightly different approximation algorithms using the greedy technique.

The first algorithm, Greedy 1, always assign a job to the currently least-loaded machine.

Greedy 1:

1. For $i = 1$ to m do $T_i \leftarrow 0$, $A(i) \leftarrow \varnothing$ End For
2. For $j = 1$ to n do
3. Let M_i be a machine so that $T_i = \min\{T_k \ : \ 1 \leq k \leq m\}$
4. $A(i) \leftarrow A(i) \cup \{j\}$
5. $T_i \leftarrow T_i + t_j$
6. End For

Running Greedy 1 on the example above we obtain the following assignment:

$$A(1) = \{1, 4\}, \quad A(2) = \{2, 5\}, \quad A(3) = \{3, 6\}$$

and the makespan is 8.

Theorem: Greedy 1 has approximation ratio of 2.

Proof: Let OPT be the optimal makespan, and let T_i be the makespan of Greedy 1. We show that $T_i \leq 2OPT$. Let j be the last job to be assigned to M_i. Then at the time when job j is assigned to M_i, M_i must have the least load. At that time the load on M_i is $T_i - t_j$. It follows that

$$T_i - t_j \leq T_k$$

for all $k, 1 \leq k \leq m$. Therefore

$$m(T_i - t_j) \leq \sum_{1 \leq k \leq m} T_k$$

The RHS is the total processing time of all jobs:

$$\sum_{1 \leq k \leq m} T_k = \sum_{1 \leq j \leq n} t_j$$

Now, for an assignment that achieves the optimal makespan OPT, the machine with load OPT has the maximum load, so

$$\sum_{1 \leq j \leq n} t_j \leq mOPT$$

It follows from the last 3 (in)equalities that

$$m(T_i - t_j) \leq mOPT$$

i.e.,

$$T_i \leq t_j + OPT \tag{1}$$

Obviously $t_j \leq OPT$, so from (1) we have $T_i \leq 2OPT$. □

Note that Greedy 1 cannot have approximation ratio c for any constant $c < 2$. Consider the following example: There are $m(m-1)+1$ jobs (i.e., $n = m(m-1)+1$). The last job has processing time m and all other jobs has processing time 1. It is easy to see that the optimal makespan in this case is m.

Running Greedy 1 on this input, the output has makespan $2m - 1$. As m goes to ∞, the ratio $(2m-1)/m$ goes to 2. So Greedy 1 cannot have a constant approximation ratio that is less than 2.

The second algorithm differs from Greedy 1 in that it sorts the jobs in decreasing order of processing time first. It turns out that this algorithm has approximation ratio of 3/2.

Greedy 2:

1. Sort the jobs so that $t_1 \geq t_2 \geq \ldots \geq t_n$
2. For $i = 1$ to m do $T_i \leftarrow 0$, $A(i) \leftarrow \varnothing$ End For
3. For $j = 1$ to n do
4. Let M_i be a machine so that $T_i = \min\{T_k \ : \ 1 \leq k \leq m\}$
5. $A(i) \leftarrow A(i) \cup \{j\}$
6. $T_i \leftarrow T_i + t_j$
7. End For

Theorem: Greedy 2 has approximation ratio 3/2.

Proof: We proceed as in the proof of the last Theorem. Let OPT be the optimal makespan, and let T_i be the makespan of Greedy 1. We show that $T_i \leq 2OPT$. Let j be the last job to be assigned to M_i. Here we consider two cases. First,

First, suppose that M_i is assigned only one job t_i. Then since the makespan is t_i, it must be an optimal makespan.

Now suppose that M_i is assigned at least 2 jobs. As in the proof of the previous theorem, we arrive at (1). Here we will show that $t_j \leq OPT/2$. It will then follow that $T_i \leq 3/2OPT$.

Since M_i is assigned at least two jobs, it must be the case that $j \geq m+1$, i.e., $t_{m+1} \geq t_j$. Now consider an assignment that achieves the optimal makespan OPT. Then there is a machine that is assigned at least two jobs from $\{1, 2, \ldots, m+1\}$. So the load on this machine is at least $2t_{m+1}$. Hence $OPT \geq 2t_{m+1}$, and thus $OPT \geq 2t_j$. □

6.3 Bin Packing Problem

Next week we will discuss an approximation algorithm for the Bin Packing Problem described below.

Suppose that we are given a set of n items where item i has size s_i which satisfies $0 < s_i < 1$. We want to pack all items into bins of unit-size bins (i.e., bins of size 1), and we want to use as few bin as possible.