Algorithmic Methods 20.12.2010

Lecture Notes 10: Local Ratio

Professor: Yossi Azar *Scribe: Inna Kalp*

1 Introduction

In this Lecture we will introduce the Local Ratio Technique. We will show applications of this method to the following problems:

- Vertex cover
- Interval scheduling
- Interval scheduling with deadlines
- Interval scheduling with intervals of various widths.

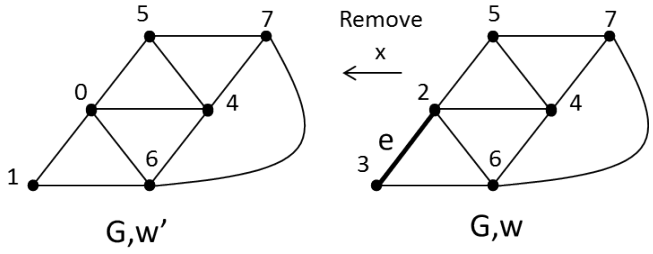
2 Vertex Cover

Introducing the problem:
Input: Graph G = (V,E), weight function $w : V \to R^+$
Feasible solution: $S \subseteq V$ s.t $\forall e = (v, u) \in E.\ u \in S$ or $v \in S$
Goal Function: Minimize $\sum_{v \in S} w(v)$.
Claim 1: The value of the solution is 0 $\Longleftrightarrow$the group of all the vertexes with weight 0 is a valid cover $\Longleftrightarrow \forall e = (u, v) \in E\ u \in S$ or $v \in S$
Proof: follows by definition.

2.1 First Solution:

Given input: G = (V,E), w(v)$\to$ R^+, choose an edge e=(u,v)$\in$ E and subtract $X = min\{w(u), w(v)\}$ from w(v) and w(u). The result is graph G with a new weight function w'.

Figure 1: An example of G with weights w & w':



Notice that:

(1) OPT(G,w')≤OPT(G,w)-X
(2) A(G,w)≤A(G,w') + 2X

Explanations:
(1) if S is a feasible solution to (G,w) with value V , it will also be a feasible solution to (G,w') with value at most V-X.
(2) if S is a feasible solution to (G,w') with value V , it will also be a feasible solution to (G,w) with value at most V +2X (if both v and u are in S, we add 2X).

Claim 2: We can achieve a 2-approximation algorithm by recursion on (G,w'):
The Algorithm: choose an edge e$\in E$ and generate (G,w'). Continue until the weight of at least one of the vertexes of each edge is 0.

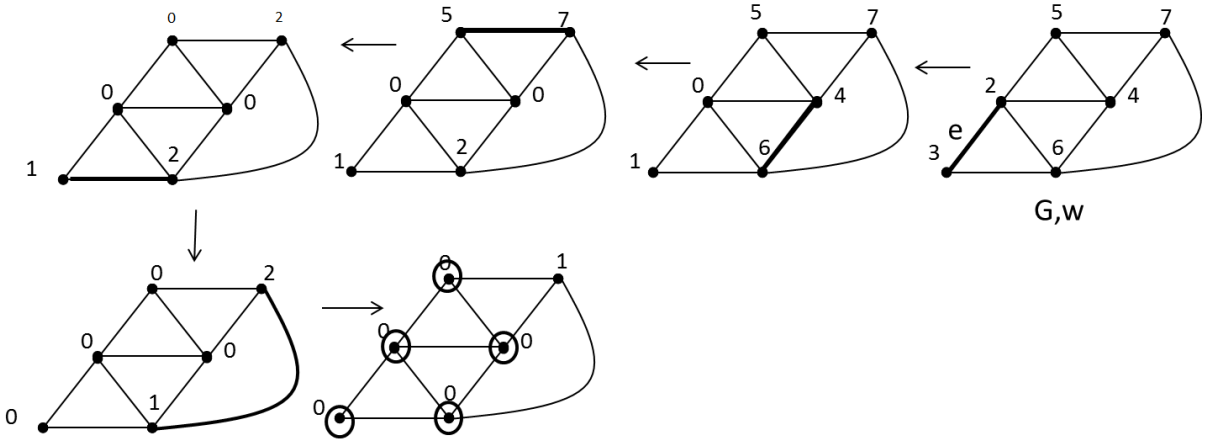
Proof: By induction: (on the number of edges (u,v) s.t w(u) >0 and w(v) > 0)

Base case: At least one of the vertexes of each edge is 0 $\rightarrow$ the value of the solution is 0 (we will take all the vertexes with weight 0). The solution is optimal, and therefore 2-approximate.

Induction Step:

$$A(G,w) \leq_{(2)} A(G,w')+2x \leq_{induction} 2OPT(G,w')+2x = 2(OPT(G,w')+x) \leq_{(1)} 2OPT(G,w)$$

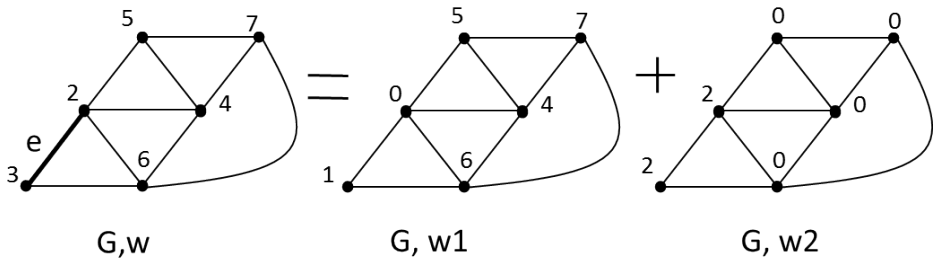
Figure 2: An Example of the recursive algorithm:



2.2 An alternative approach:

We divided the weight function w into 2 functions: w_1:V$\rightarrow$ $R^+$and w_2:V$\rightarrow$ R^+ s.t w = w_1+w_2.

Figure 3: The division of W in our example:



The Local Ratio Theorem: If solution S is α-approximate for (G, w_1) and (G, w_2), S is also $\alpha - approximate$ for (G, w)
Proof: $w(S) = w_1(S) + w_2(S) \leq \alpha OPT(G, w_1) + \alpha OPT(G, w_2) = \alpha(OPT(G, w_1) + OPT(G, w_2)) \leq \alpha OPT(G, w_1 + w_2) = \alpha OPT(G, w)$

The local Ratio Theorem applied to the Vertex Cover:

The Algorithm:

- If a zero-cost solution can be found, return one. Otherwise:
- decompose w into w_1&w_2 (as described above)
- solve the problem recursively, using w_1 as the weight function in the recursive call.

Claim 3: The algorithm is 2-approximate.
Proof: By Induction: (on the number of edges (u,v) s.t w(u) >0 and w(v) > 0)
Base case: The algorithm returns a VC of zero cost, which is optimal.
Inductive step: consider the solution returned by the recursive call. In $(G, w1)$ there is one edge (u, v) less s.t $w(u) > 0$ and $w(v) > 0$, by the inductive hypothesis it is 2-approximate with respect to w_1. We claim that it is also 2-approximate with respect to w_2 . In fact, every feasible solution is 2-approximate with respect to w_2: OPT(G,w_2)$\geq x$ and

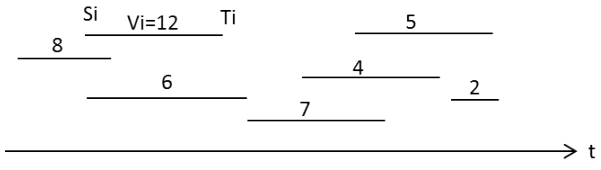
$$Alg(G, w_2) = \sum_{v \in S} w_2(v) \leq \sum_{v \in V} w_2(v) = 2x.$$

Hence there is a single solution which 2-approximate both (G,w_1) and (G,w_2). Hence this single solution also 2-approximate (G, w).

3 Interval Scheduling

Introducing the problem:
Input: A set of intervals: $I_i = (s_i, t_i, v_i)$ where $s_i \leq t_i$ are the endpoints and v_i is its the value if accepted.
Feasible solution: subset S of non-conflicting intervals.
Goal Function: maximize the sum of values of S.

Figure 4: Example of Interval scheduling problem:



3.1 Simple Case: $\forall i.v_i = 1$:

The Goal is to maximize the number of intervals in S.
The Algorithm: Choose the interval that ends first, and remove all of its conflicting intervals. Repeat until there are no more intervals.
Claim 4: The Algorithm gives an optimal solution.
Proof: The chosen interval is better than any other interval, because it can be replaced by any of the other intervals we removed.

3.2 The General Case $\forall i.v_i \geq 0$:

3.2.1 Solution using dynamic programming:

The Algorithm:
Sort the intervals by the ending time(t_i):$I_1 \leq I_2 ... \leq I_n$
Define: $F(I)$ = the maximum gain from intervals ending until the end of the i-th interval. In particular: $F(0) = 0$.
Denote by J_i= the interval of maximum index that ends before the beginning of interval i. Then we have
$F(I) = \max\{F(i-1), v(i) + F(j_i)\}$

3.2.2 Solution using Local Ratio technique:

The Algorithm:

1. If there are interval s.t $v_i \leq 0$ - remove them. If the input is $\phi - return\ \phi$.
2. The weight function $w_2(v)$: Select the interval that that ends first - I_1 with value v and remove v from all of $I_1's$ conflicting intervals. (That means that the weight function $w_1(v) = w(v) - w_2(v)$). Remove I_1.
3. Solve recursively.
4. Take the recursive solution and add I_1 to it if possible (= if interval I_1 does not conflict with the other intervals in the solution).

Claim 5: The algorithm returns an optimal solution.
Proof: By induction (on the number of intervals):
Base case: If the input is $\phi - return\ \phi$- optimal solution.
Inductive step: First, notice that the all the intervals with value > 0 are conflicting with each other, and OPT = v. Consider the solution S returned by the recursive call. In w_1 there is at least one interval less than in w. By the inductive hypothesis S is optimal with respect to w1.
If S contains an interval conflicting with I_1: There is an interval conflicting with I_1in the solution $\rightarrow$ the value of the solution in Alg(w_2) = $v \rightarrow optimal$.
If S doesn't contain an interval conflicting with I_1 we add it to the solution: $Alg(w_2) = v \rightarrow optimal$. S is optimal with respect to w_1 and $w_2 \rightarrow S$ is optimal with respect to w = $w_1 + w_2$.

4 Interval Scheduling with deadlines

Introducing the problem:
Input: A set of intervals: $I_i = (s_i.t_i, d_i, v_i)$: interval i can be placed in the range $[s_i, t_i]$, the interval's length is d_i, and its value - v_i.
Feasible solution: Subset S of non-conflicting intervals, and the starting location for each selected interval in its range.
Goal Function: Maximize the sum of values of S.
Remark: The problem is NP-Complete (can be proved by reduction from Bin Packing).
Assumption: s_i, t_i, d_i are Integer.
The technique: We would actually solve a more general problem where each interval has multiple discrete options, where each option might have a different value. We can choose at most one occurrence of the interval.
Transformation from deadlines to multiple discrete options: Open each interval to a set of all of it's possible (integer) locations within the range $[s_i, t_i]$.
The Algorithm (multiple discrete options):

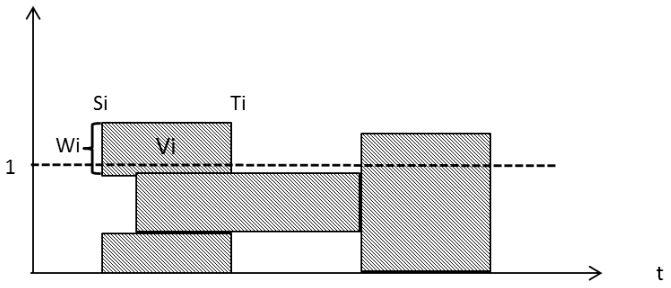
1. If there are interval s.t $v_{ij} \leq 0$ remove them. If the input is ϕ return .
2. The weight function $w_2(v)$: Select the interval that that ends first - I_1 with value v and remove value v from: (1) all of $I_1's$ conflicting intervals. (2) all the copies of I_1. (the weight function $w_1(v) = w(v) - w_2(v)$). Remove I_1.
3. Solve recursively.
4. Take the recursive solution and add I_1 to it if possible (= if interval I_1 does not conflict with the other intervals in the solution, and there is not any copy of I_1 in the solution).

Claim 6: The algorithm is 2-approximate.
Claim 7: The optimal solution with respect to the weight function w_2 is at most $2v$.
Proof (of Claim 7): We can have a value of at most v from all the intervals conflicting with I_1, and value of at most v from a copy of I_1. We get at most $2v$.
Claim 8: The value of every maximal solution with respect to I_1 is at least v (for weight function w_2).
Proof (of Claim 8): By definition: If we take a copy of I_1 then we get a value of v. If we take a copy of an interval conflicting with I_1 then we get a value of v. Otherwise, we can add I_1 to the solution and again we get a value of v.
Corollary 1: From claims 7 and 8 we get that any feasible solution to w_2 is 2-approximate.
Proof (of Claim 6): By induction (on the number of intervals):
Base case: If the input is ϕ -return ϕ which is an optimal solution (and in particular 2-approximate).
Inductive step: Consider the solution S returned by the recursive call (and possibly adding I_1). In w_1 there is at least one interval less than in w. By the inductive hypothesis S is 2-approximate with respect to w_1 (which clearly remains true if we also add I_1). We have also shown (in corollary 1) that any maximal solution to w_2 is 2-approximate. Since S is 2-approximation with respect to w_1 and $w_2 \rightarrow S$ it is also 2-approximation with respect to w = $w_1 + w_2$.

5 Interval Scheduling with intervals of various widths:

Introducing the problem:
Input :A set of intervals: $I_i = (s_i.t_i, b_i, v_i)$: s_i - the start of the interval, t_i- the end of the interval, b_i,- the interval's bandwidth, v_i- the interval's value.
Feasible solution: Subset S of intervals, s.t at any given time, the sum of the bandwidths of the intervals is at most 1.
Goal Function: Maximize the sum of values of S.

Figure 5: An example of Interval Scheduling with intervals of various bandwidths:



The Algorithm:

Phase 1: Separate the input into 2 groups: $I_1 = \{i|b_i > \frac{1}{2}\}$; $I_2 = \{i|b_i \leq \frac{1}{2}\}$
Phase 2: Solve each group separately:
Output: Take the better solution among the two.

Solution to I_1: since $b_i > \frac{1}{2}$, at each given moment, there can be placed at most 1 interval. This is the regular Interval Scheduling problem- we have shown how to solve this problem optimally.
Solution to I_2 :, i.e., $b_i \leq 1/2$.

1. If there are interval s.t $v_i \leq 0$ - remove them. If the input is $\phi - return\ \phi$.
2. The weight function $w_2(v)$: Select the interval that that ends first - I_1 with value v and bandwidth b, and remove $\frac{v*b_i}{1-b}$ from all of $I_1's$ conflicting intervals. (b_i is the bandwidth of the conflicting interval). The weight function $w_1(v) = w(v) - w_2(v)$. Remove I_1.
3. Solve recursively.
4. Take the recursive solution and add I_1 to it if possible (= if interval I_1adding interval I_1 does not result in a time t s.t the sum of the weights at that time is over 1).

Claim 9: Optimal solution for I_2 with respect to the weight function w_2 is at most $2v$.
Proof : We note that

$$OPT \leq max\{v\frac{\sum b_i}{1-b}, v + v\frac{\sum b_i}{1-b}\} \leq max\{v\frac{1}{1-b}, v + v\frac{1-b}{1-b}\} = max\{2v, 2v\} = 2v$$

Explanation: The first option where the value of OPT is at most $v\frac{\sum b_i}{1-b}$ represents the case where we did not add $I_1$to the solution. In this case $\sum b_i \leq 1$ since it is a valid solution,

and since $b \leq 1/2$ we get that the value of OPT is at most $2v$. The second option (in which we add I_1 to the solution): the value of OPT is at most $v + v\frac{\sum b_i}{1-b}$ since we get v for I_1 and $v\frac{\sum b_i}{1-b}$ for all of $I_1's$ conflicting intervals. We can add I_1, so $\sum b_i + b \leq 1 \rightarrow \sum b_i \leq 1 - b$. We get that OPT$\leq 2v$.

Claim 10: the value of every maximal solution with respect to I_1 is at least v with the weight function w_2.

Proof: Alg$\geq \min\{v, v\frac{\sum b_i}{1-b}\} \geq \min\{v, v\frac{1-b}{1-b}\} = min\{v, v\} = v$.

Explanation: The first option Alg gets at least v since we add I_1 to the solution and hence the solution value is at least v. The second option(we cannot add I_1 to the solution): OPT$\geq v\frac{\sum b_i}{1-b}$. Since we can not add I_1 then $\sum b_i + b > 1 \rightarrow \sum b_i > 1 - b$. Hence Alg gets at least v .

Corollary 2: From claims 9 and 10 we get that any maximal solution created with the weight function w_2 is 2-approximate.

Claim 11: The solution to I_2 is 2-approximate.

Proof: The proof of claim 6 (the algorithm for interval scheduling with deadlines) also applies here.

Phase 3: We solve the problem for I_1 and I_2 and take the best solution.

Claim 12: The algorithm achieves 3-approximation.

Proof: OPT$\leq OPT(I_1) + OPT(I_2)$.

If OPT$(I_1)\geq \frac{1}{3}OPT$:Alg(I_1) returns an optimal solution, so we get :Alg$(I_1) = OPT(I_1) \geq \frac{1}{3}OPT$. We get a 3-approximation.

Otherwise, OPT$(I_2)\geq \frac{2}{3}OPT$. We got a 2-approximation to I_2, So We Get Alg$(I_2) \geq \frac{1}{2}$OPT$(I_2) \geq \frac{1}{2} * \frac{2}{3}OPT = \frac{1}{3}OPT$.

In both cases we get 3-approximation.