Web Algorithms

Eng. Fabio Persia, PhD

Alternative Approaches

Alternative approaches

So far we have considered approaches with

Guaranteed performance

- Pro:
  - Approximation guaranteed for every input instance
  - Running time guaranteed for every input instance
  - It takes into account the worst case

- Cons:
  - Several problems do not admit algorithms with guaranteed performance
  - For several problems algorithms with guaranteed performance are not known yet
  - Sometimes bad behavior in practice

Let us now see some alternative approaches.

Restriction of the set of the instances

Guaranteed performance on the subset of the input instances that are significative or of practical interest.

- Pros:
  - It allows to apply again the guaranteed performance approach

- Cons:
  - Guaranteed performance only for the chosen subset of instances or for particular cases

Example: metrical TSP (with triangular inequalities)

Average or probabilistic analysis

In general, assuming a probability distribution of the instances, it evaluates the average or expected performance, sometimes with high probability

- Pros:
  - It can detect good practical behavior of an algorithm
  - It is an analytical method, that is based on mathematical proofs

- Cons:
  - It does not have guaranteed performace
  - The analysis often is very complex
  - Often the distribution of the input instances is unknown

Heuristics

Algorithm with good practical behavior but usually with not provable performance.

- Pros:
  - Good practical behavior
- Cons:
  - Performance often not demonstrable

Randomized algorithms

- They make random choices during their execution.
- The returned solutions may be different for different executions on the same input. They are in fact random variables (for each instance there are several solutions, each returned with a certain probability determined according to the random choices of the algorithm).
- It is shown that, fixed any instance, the expected value of the performance is good or the performance is good with high probability (always according to the random choices).

- Pros:
  - They are generally simple
  - They are fast (both analytically and in practice)

- Cons:
  - Uncertainty of the result for each fixed instance
  - Impossibility of making real random choices (although they can be simulated)

Randomized algorithms

m $\rightarrow$ is a random variable

E(m) $\rightarrow$ is the expected value of *m* computed according to the random choices of the algorithm

Def: A randomized algorithm *A* is r-approximating if

$$\frac{E(m)}{m^*} \leq r \qquad \text{(MIN)}$$

$$\frac{E(m)}{m^*} \geq r \qquad \text{(MAX)}$$

Max Weighted Cut

- INPUT: Graph $G=(V,E)$ with a non negative weight ω_{ij} for each edge $\{v_i, v_j\} \in E$

- SOLUZIONE: Partition of V in two subsets V_1 and V_2 such t that

$$V_1 \cap V_2 = \emptyset \text{ and } V_1 \cup V_2 = V$$

- MISURA: weight of the cut, that is

$$\sum_{\{v_i, v_j\} \in E \mid v_i \in V_1 \wedge v_j \in V_2} \omega_{ij}$$

Algorithm Random Cut

Begin

$V_1 = \emptyset.$

$V_2 = \emptyset.$

For $i=1$ to n

Put v_i in V_1 with probability ½ independently from the other nodes (else in V_2).

Return V_1 and $V \setminus V_1$ $(\equiv V_2).$

End

Clearly the algorithm is polynomial

Theorem: Random-Cut is a ½-approximation algorithm

Proof: Let x_{ij} the random variable "the edge $\{v_i, v_j\}$ is in the cut".

Then $$m = \sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot x_{ij}$$

Expectation

Thus $$E(m) = E\left(\sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot x_{ij} \right) = \sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot E(x_{ij}) =$$

$$= \sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot P(x_{ij} = 1) = \sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot P((v_i \in V_1 \wedge v_j \in V_2) \vee (v_i \in V_2 \wedge v_j \in V_1)) =$$

Probabilità

$$= \sum_{\{v_i, v_j\} \in E} \omega_{ij} \cdot \left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \right) = \frac{1}{2} \cdot \sum_{\{v_i, v_j\} \in E} \omega_{ij} \geq \frac{m^*}{2}$$

Therefore $$\frac{E(m)}{m^*} \geq \frac{1}{2}$$ □

Min Weighted Set Cover

- INPUT: Universe $U=\{o_1,\ldots,o_n\}$ of n objects, family $\hat{S}=\{S_1,\ldots,S_h\}$ of h subsets of U, integer cost c_j associated to every $S_j \in \hat{S}$

- SOLUTION: Cover if U, that is subfamily $\hat{C} \subseteq \hat{S}$ such that

$$\bigcup_{S_j \in \hat{C}} S_j = U$$

- MEASURE: Total cost of the cover, that is $\sum_{S_j \in \hat{C}} c_j$

f = max frequency of an object in the subsets of $\hat{S}$, that is every object occurs in at most *f* subsets.

Min Weighted Set Cover

Given a set of elements $\{1, 2, \ldots, n\}$ (called the universe) and a collection S of m sets whose union equals the universe, the set cover problem is to identify the smallest sub-collection of S whose union equals the universe. For example, consider the universe $U = \{1, 2, 3, 4, 5\}$ and the collection of sets $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. Clearly the union of S is U. However, we can cover all of the elements with the following, smaller number of sets: $\{\{1, 2, 3\}, \{4, 5\}\}$.

Greedy algorithm for Min Weighted Set Cover

Remark:

In the choice of the subsets to be put in the cover:

- we cannot take into account only costs, because in proportion we could cover too few elements of *U*
- we cannot take into account only the number of covered objects, because we might incur an excessive cost

Thus greedy choice:

at every step choose the subset having minimum cost per new covered object

Greedy choice

At a given step j in which in the order the algorithm has selected j-1 subsets $S_1,\ldots,S_{j-1}$, the effectiveness of a not yet chosen subset S_k is defined as:

$$eff(S_k) = \frac{c_k}{\left|S_k \cap \overline{C_{j-1}}\right|}$$

where

c_k = cost of S_k

$C_{j-1} = (S_1 \cup \ldots \cup S_{j-1})$

$\overline{C_{j-1}} = U \setminus C_{j-1}$

Greedy choice

At step j choose a subset S_j of minimum effectiveness, that is such that

$$eff(S_j) = min \{ eff(S_k) \mid S_k \textit{ has not be chosen yet} \}$$

Algorithm Greedy-Min-Weighted-Set-Cover

Begin

$C=\emptyset$. // *covered objects*

$\hat{C}=\emptyset$. // *contains the subsets chosen in the cover*

$j=1$.

While $C \neq U$

Let S_j be a subset of minimum effectiveness.

Put S_j in $\hat{C}$.

$\forall$ object $o_i \in S_j \cap \overline{C}$ let $price(o_i)=eff(S_j)$.

$C=C \cup S_j$.

$j=j+1$.

Return $\hat{C}$.

End

Lemma:

$$m = \sum_{S_j \in \hat{C}} c_j = \sum_{i=1}^{n} price(o_i)$$

Proof: Trivial as the sum of the prices of the objects covered during step *j* is just c_j. In other words, the total cost is split among the covered objects.

□

Lemma: For every j, given any choice of subsets $S_j', \ldots, S_t'$ that form a cover with the subsets $S_1, \ldots, S_{j-1}$ chosen by the greedy algorithm at the beginning of step j, for every object o_i not yet covered at the beginning of step j $price'(o_i) \geq eff(S_j)$, where S_j is the subset chosen by the greedy algorithm at step j, $eff(S_j)$ its effectiveness, $price'(o_i)$ is the effectiveness of the subset S_l' covering o_i assuming that, starting from step j, the greedy choice is done among subsets $S_j', \ldots, S_t'$ only.

Proof: It is sufficient to observe that $eff(S_j)$ is the minimum covering price per object at step j and that the effectiveness of an unchosen subset can only increase during the steps, as its cost is fixed while some further objects in it can be covered during the steps due to the choice of other subsets.

Thus the price of o_i, that is the effectiveness of the subset S_l' with $l \geq j$ among $S_j', \ldots, S_t'$ that covers it, is at least equal to $eff(S_j)$. □

- **Lemma**: Let $o_1,\ldots,o_n$ be the objects listed in the covering order of the greedy algorithm, that is such that the objects covered during step j are listed after the ones covered in the previous steps and before the ones covered in the successive steps,

  Then, $\forall i$ such that $1 \leq i \leq n$,
$$price(o_i) \leq \frac{m^*}{n-i+1}$$

Proof: Consider any object o_i and let j the step in which it is covered.

At the beginning of step j, since the not yet chosen sets of an optimal solution can cover all the uncovered objects with overall cost at most m^*, there must exist a subset S_k of effectiveness at most $m^*/\ |C_{j-1}|$, where C_{j-1} is the subsets of objects not yet covered at the beginning of step j.

In fact, if this is not the case, since the price of o_i is equal to the effectiveness $eff(S_j)$ of the subset S_j chosen by the greedy algorithm, by the previous lemma, for every possible choice of remaining subsets to complete the cover, that is for any possible pricing of the remaining objects

$$\sum_{o_i \in \overline{C_{j-1}}} price'(o_i) \geq \sum_{o_i \in \overline{C_{j-1}}} eff(S_j) > \sum_{o_i \in \overline{C_{j-1}}} \frac{m^*}{|\overline{C_{j-1}}|} = |\overline{C_{j-1}}| \cdot \frac{m^*}{|\overline{C_{j-1}}|} = m^*$$

A CONTRADICTION to the hypothesis that there exists a choice of subsets that covers the remaining objects with cost at most m^*.

Thus, $$price(o_i) \leq \frac{m^*}{|\overline{C_{j-1}}|}$$

But $|\overline{C_{j-1}}| \geq n-i+1$ as $o_i, \ldots, o_n \in \overline{C}_{j-1}$ and as a consequence

$$price(o_i) \leq \frac{m^*}{|\overline{C_{j-1}}|} \leq \frac{m^*}{n-i+1} \quad \square$$

Theorem: The greedy algorithm for Min Weighted Set Cover is H_n-approximating, where $H_n=1+1/2+1/3+1/4+\ldots 1/n$.

Proof:

$$m = \sum_{i=1}^{n} price(o_i) \leq \sum_{i=1}^{n} \frac{m^*}{n-i+1} = m^* \cdot \left(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \ldots + 1\right) = m^* \cdot H_n$$

from which $$\frac{m}{m^*} \leq H_n$$

$\square$

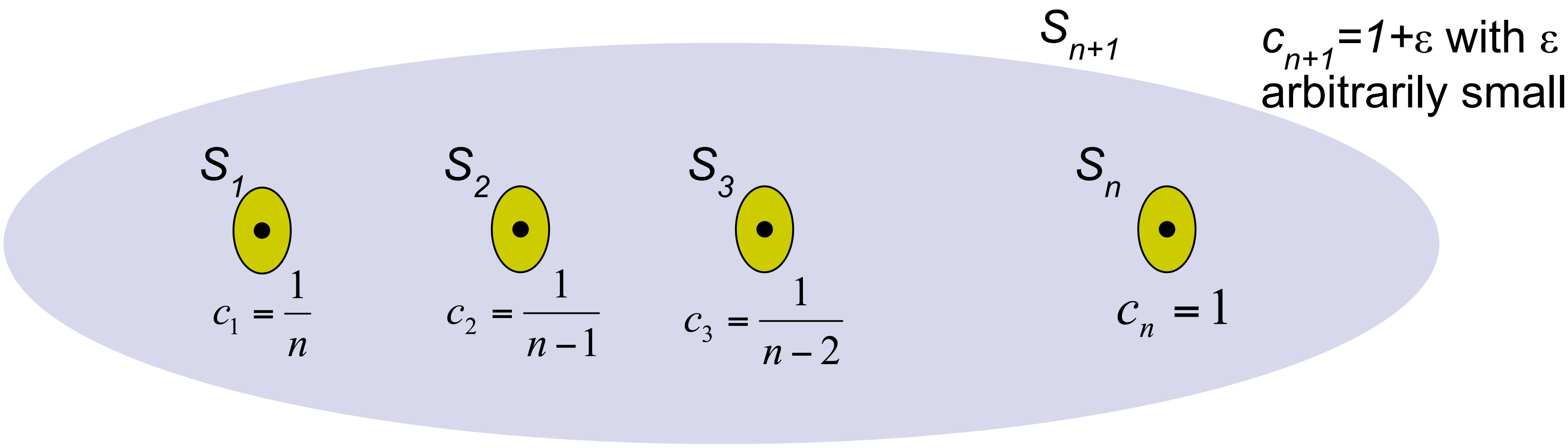
Remark. $\forall n>1$, $\ln(n+1) \leq H_n \leq \ln(n)+1$

Thus r has a logarithmic dependency from the input size

Example (strict ratio)

The approximation ratio of the greedy algorithm is at least H_n

Consider in fact the following instance:



At the first step:

- $$eff(S_{n+1}) = \frac{1+\varepsilon}{n}$$
- $$eff(S_1) = \frac{1}{n}$$

At the second step:

- $$eff(S_{n+1}) = \frac{1+\varepsilon}{n-1}$$
- $$eff(S_2) = \frac{1}{n-1}$$

Thus the algorithm will never choose S_{n+1}

$$m = \sum_{j=1}^{n} \frac{1}{j} = H_n$$

$m^*=1+\varepsilon$ (with ε arbitrarily small, even null)

Thus $$\frac{m}{m^*} = \frac{H_n}{1+\varepsilon}$$

Then, for $\varepsilon \to 0$, $$\frac{m}{m^*} \to H_n$$

Thus the measure of the solution returned by the greedy algorithm for some instances is H_n times the optimal one!