Covering problems

Prototype problems: Covering problems

- Setting:
  - Universe of N elements $U = \{U_1,\ldots,U_N\}$
  - A set of n sets $S = \{s_1,\ldots,s_n\}$
  - Find a collection C of sets in S (C subset of S) such that $\bigcup_{c \in C} c$ contains many elements from U
- Example:
  - U: set of documents in a collection
  - s_i: set of documents that contain term t_i
  - Find a collection of terms that cover most of the documents

Prototype covering problems

- **Set cover problem:** Find a small collection **C** of sets from **S** such that all elements in the universe **U** are covered by some set in **C**
- **Best collection problem:** find a collection **C** of **k** sets from **S** such that the collection covers as many elements from the universe **U** as possible
- Both problems are NP-hard
- Simple approximation algorithms with provable properties are available and very useful in practice

Set-cover problem

- Universe of **N** elements $U = \{U_1,\ldots,U_N\}$
- A set of **n** sets $S = \{s_1,\ldots,s_n\}$ such that $\cup_i s_i = U$

- **Question:** Find the smallest number of sets from **S** to form collection **C** (**C** subset of **S**) such that $\cup_{c \in C} c = U$

- The set-cover problem is **NP-hard** (what does this mean?)

Trivial algorithm

- Try all subcollections of **S**
- Select the smallest one that covers all the elements in **U**
- The running time of the trivial algorithm is $O(2^{|S|}|U|)$
- This is way too slow

Greedy algorithm for set cover

- Select first the largest-cardinality set **s** from **S**
- Remove the elements from **s** from **U**
- Recompute the sizes of the remaining sets in **S**
- Go back to the first step

As an algorithm

- $X = U$
- $C = \{\}$
- **while** X is not empty **do**
  - For all $s \epsilon S$ let $a_s = |s \text{ intersection } X|$
  - Let s be such that a_s is maximal
  - $C = C \cup \{s\}$
  - $X = X \setminus s$

How can this go wrong?

- No global consideration of how good or bad a selected set is going to be

How good is the greedy algorithm?

- Consider a minimization problem
  - In our case we want to minimize the **cardinality** of set **C**
- Consider an instance **I**, and cost **$a^*(I)$** of the optimal solution
  - **$a^*(I)$**: is the minimum number of sets in **C** that cover all elements in **U**
- Let **$a(I)$** be the cost of the approximate solution
  - **$a(I)$**: is the number of sets in **C** that are picked by the greedy algorithm
- An algorithm for a minimization problem has approximation factor **F** if for all instances **I** we have that

$$a(I) \leq F \times a^*(I)$$

- **Can we prove any approximation bounds for the greedy algorithm for set cover ?**

How good is the greedy algorithm for set cover?

- **(Trivial?) Observation**: The greedy algorithm for set cover has approximation factor $F = s_{max}$, where s_{max} is the set in S with the largest cardinality
- **Proof:**
  – $a^*(I) \geq N/|s_{max}|$ or $N \leq |s_{max}|a^*(I)$
  – $a(I) \leq N \leq |s_{max}|a^*(I)$

How good is the greedy algorithm for set cover? A tighter bound

- The greedy algorithm for set cover has approximation factor $F = O(\log |s_{max}|)$
- **Proof**: (From CLR "Introduction to Algorithms")

Best-collection problem

- Universe of N elements $U = \{U_1,\ldots,U_N\}$
- A set of n sets $S = \{s_1,\ldots,s_n\}$ such that $\cup_i s_i = U$

- **Question:** Find the a collection C consisting of k sets from S such that $f(C) = |\cup_{c\in C} c|$ is **maximized**

- The best-colection problem is NP-hard

- Simple approximation algorithm has approximation factor $F = (e-1)/e$

Greedy approximation algorithm for the best-collection problem

- $C = \{\}$
- **for every** set s in S and **not** in C compute the gain of s:

$$g(s) = f(C \cup \{s\}) - f(C)$$

- Select the set s with the **maximum** gain
- $C = C \cup \{s\}$
- **Repeat until** C has k elements

Basic theorem

- The **greedy** algorithm for the best-collection problem has approximation factor **$F = (e-1)/e$**

- **C^* : optimal** collection of cardinality **k**
- **C :** collection output by the **greedy** algorithm
- **$f(C) \geq (e-1)/e \times f(C^*)$**