Title: Min Sum Set Cover and its Generalizations
Name: Sungjin Im[1]
Affil./Addr. Electrical Engineering and Computer Science, University of California, Merced, CA, USA
Keywords: Approximation; Greedy algorithm; Randomized rounding; Latency; Covering problems; Set cover; Submodular
SumOriWork: 2004; Feige, Lovász, Tetali
2009; Azar, Gamzu, Yin
2010; Bansal, Gupta, Krishnaswamy
2011; Azar, Gamzu
2011; Skutella, Williamson
2012; Im, Nagarajan, Zwaan
2013; Im, Sviridenko, Zwaan

Min Sum Set Cover and its Generalizations

Sungjin Im[1]

Electrical Engineering and Computer Science, University of California, Merced, CA, USA

Years aud Authors of Summarized Original Work

2004; Feige, Lovász, Tetali
2009; Azar, Gamzu, Yin
2010; Bansal, Gupta, Krishnaswamy
2011; Azar, Gamzu
2011; Skutella, Williamson
2012; Im, Nagarajan, Zwaan
2013; Im, Sviridenko, Zwaan

Keywords

Approximation; Greedy algorithm; Randomized rounding; Latency; Covering problems; Set cover; Submodular

Problem Definition

The Min Sum Set Cover Problem (MSSC), which was introduced in [4], is a latency version of the Set Cover Problem. The input to MSSC consists of a collection of sets $\{S_i\}_{i \in [m]}$ over a universe of elements $[n] := \{1, 2, 3, ..., n\}$. The goal is to schedule elements, one at a time, to hit all sets as early on average as possible. Formally, we would like to find a permutation (one-to-one mapping) $\pi : [n] \to [n]$ of the elements $[n]$

($\pi(i)$ is the ith left-most element in the ordering) such that the average (or equivalently total) cover time of the sets $\{S_i\}_{i\in[m]}$ is minimized. The cover time of a set S_i is defined as the earliest time t such that $\pi(t) \in S_i$. For convenience, we will say that we schedule/process element $\pi(i)$ at time i.

Since MSSC problem was introduced in [4], its several generalizations have been studied. Here we discuss two of them. In the Generalized Min Sum Set Cover problem (GMSSC) [2], each set S_i has a requirement κ_i. In this generalization, a set S_i is covered at the first time t when κ_i elements are scheduled from S_i, i.e. $|\{\pi(1), \pi(2), ..., \pi(t)\} \cap S_i| \geq \kappa_i$. Note that MSSC is a special case of GMSSC when $\kappa_i = 1$ for all $i \in [n]$.

Another interesting generalization is the Submodular Ranking (SR) [1]. In SR, each set S_i is replaced with a non-negative and monotone submodualr function $f_i : 2^{[n]} \rightarrow [0,1]$ with $f_i([n]) = 1$; function f is said to be submodular if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq [n]$, and monotone if $f(A) \leq f(B)$ for all $A \subseteq B$. The cover time of each function f_i is now defined as the earliest time t such that $f_i(\{\pi(1), \pi(2), ..., \pi(t)\}) = 1$. Note that GMSSC is a special case of SR when $f_i(A) = \min\{|S_i \cap A|/\kappa_i, 1\}$. Also it is worth noting that SR generalizes the Set Cover.

Key Results

There is a 4-approximation for MSSC, and there is a matching lower bound $4 - \epsilon$ unless $P = NP$ [4]. Interestingly, the tight 4-approximation is achieved by a very simple greedy algorithm that schedules an element at each time that covers the largest number of uncovered sets. The analysis in [4] uses histograms very cleverly. Inspired by this analysis, Azar and Gamzu gave a greedy $O(\log \max_i \kappa_i)$ approximation for GMSSC [2].

Bansal et al. show that the analysis of the greedy algorithm in [2] is essentially tight, and uses a Linear Programming relaxation and randomized rounding to give the first $O(1)$-approximation for GMSSC; the precise approximation factor obtained was 485. The LP used in [3] is a time-indexed LP strengthened with Knapsack Covering inequalities. The rounding procedure combines threshold rounding and randomized "boosted-up" independent rounding. The approximation was later improved to 28 by [7], subsequently to 12.4 by [6]. The key idea for these improvements was to use α-point rounding to resolve conflicts between elements – α-point rounding is popular in scheduling literature.

There is a greedy $O(log(1/\epsilon))$-approximation known for SR where ϵ is the minimum marginal positive increase of any function f_i [1]. Note that this immediately implies an $O(\log \max_i \kappa_i)$-approximation for GMSSC. The algorithm in [1] is an elegant greedy algorithm which schedules an element e at time t with the maximum $\sum_i (f_i(A \cup \{e\}) - f_i(A))/(1 - f_i(A))$ – here A denotes all elements scheduled by time $t-1$, and if $f_i(A) = 1$, then f_i is excluded. Note that this algorithm becomes the greedy algorithm in [4] for the special case of MSSC. The analysis is also based on histograms. Later, Im et al. [5] gives an alternative analysis of this greedy algorithm which is inspired by the analysis of other latency problems. We also note that the algorithm that schedules element e that gives the maximum total marginal increase of $\{f_i\}$ has a very poor performance.

As we discussed above, there are largely three analysis techniques used in this line of work: histogram based analysis, latency argument based analysis, and LP rounding. We will sketch these techniques closely following [4; 5; 3]. We chose these papers since they present the techniques in a simpler manner, not necessarily they give the best approximation guarantees or more general results. We first set up some additional notation which will be used throughout the illustration. Let R_t denote the uncovered sets at time t, and N_t the sets that are first covered at time t. It is easy to see that $\sum_{t \in [n]} |R_t|$ is the algorithm's total cover time. We begin with the analysis tools developed for greedy algorithms. To present the key ideas more transparently, we will focus on MSSC.

Histogram based analysis. We sketch the analysis of the 4-approximaiton in [4]. In the analysis, we represent the optimal and the algorithm's solutions using histogram. First, in the optimal solution's histogram sets are ordered in increasing order of their cover times, and set S_i has width 1 and height equal to its cover time. In the algorithm's solution, as before, sets are ordered in increasing order of their cover times, but set S_i has height equal to it price. That is, we let S_i pays price $|R_t|/|N_t|$ where t is S_i's cover time. In other words, here we are charging the increase in the algorithms objective at time t to sets N_t uniformly. Note that the areas of both histograms are equal to the optimal cost and the algorithm's cost, respectively. Then we can show that after shrinking the algorithm's histogram by a factor of 2, both horizontally and vertically, one can place it completely inside the optimal solution's histogram. This analysis is very simple and is based on a clever observation on the greedy solution's structure.This type of analysis was also used in [2; 1].

Latency argument based analysis. This analysis does not seem to yield tight approximation guarantees, but could be more flexible since it does not compare two histograms directly. We first observe that we only need to focus on geometrically increasing time steps. That is, $\sum_{j \geq 0} 2^j |R_{2^{j+1}}| \leq ALG \leq \sum_{j \geq 0} 2^j |R_{2^j}|$. Then, it is easy to see that it suffices to show $|R_{2^j}| \leq \frac{1}{4}|R_{2^{j-1}}| + O(1)|R^*_{2^j}|$. Here $R^*_{2^j}$ is similarly defined for the optimal solution as R_{2^j} is for the algorithm. What this implies is the following: if the algorithm has a large number of uncovered sets compared to the optimal solution, then it must be the case that the algorithm covers a large fraction of new sets during time $[2^{j-1}, 2^j]$. Intuitively, (a weak version of) this claim should be true. The optimal solution could cover sets $R_{2^{j-1}} \setminus R^*_{2^j}$ till time 2^j, and hence at time 2^{j-1}, the greedy algorithm should be able to cover at least $|R_{2^{j-1}} \setminus R^*_{2^j}|/2^j$ new sets. This argument can be extended to every time step during $[2^{j-1}, 2^j]$, and be modified to prove $|R_{2^j}| \leq \frac{1}{4}|R_{2^{j-1}}| + O(1)|R^*_{2^j/100}|$, which is sufficient for our purpose. This analysis is easily generalized to GMSSC, SR, and more general metric settings [5].

We now discuss the linear programming based approach. Bansal et el. discuss why it seems hard to use greedy algorithms to get an $O(1)$-approximation for GMSSC [3].

LP and Randomized rounding. The LP is time-indexed. There is a variable x_{et} which denotes how much element e is processed at time t. Let $y_{et} := \sum_{t' \leq t} x_{et'}$ denote how much element e is processed by time t. Also let z_{it} denote how much set S_i is covered by time t. The objective is $\sum_t \sum_i (1 - y_{it})$; if set S_i is not covered at time t, it has to pay cost 1 at the time. If we use the most natural constraint $\sum_{e \in S_i} y_{et} \geq \kappa_i z_{it}$, the LP has a large integrality gap [3]. Hence, [3] strenthens the LP with Knapsack covering inequalities: $\sum_{e \in S_i \setminus A} y_{et} \geq (\kappa_i - |A|) z_{it}$ for all $A \subseteq S_i$, i, t. Although these are exponentially many, there is an easy separation oracle for these constraints, hence we can solve the LP in polynomial time. Note that the LP pays cost at least $t/2$ for element i if $y_{et} \geq 1/2$. Let h_e be the latest time such that $y_{et} \geq 1/2$. The first step in the analysis in [3] is to show that each element e is scheduled with a constant probability, say $7/8$, before the linear programming does by half at time h_e. By repeating this, more precisely by trying to schedule the element again within $2h_e$, $4h_e$, $8h_e$,$\cdots$, one can show that the expected cover time of e is $O(1)h_e$. The constants need to be carefully chosen and the schedules from different phases in the iteration need to be mixed cleverly, but this is the high-level idea.

Applications

The MSSC problem and its closely related problems have various applications in adaptive query processing, distributed resource allocation problems. Also GMSSC has applications in web page ranking and broadcast scheduling. For details, see [4; 1]. Min sum set cover problems are at least loosely connected to all problems where the goal is to satisfy multiple demands with the overal minimum latency.

Open Problems

One outstanding open problem is to settle down the approximability of GMSSC. As mentioned before, GMSSC captures MSSC (all $\kappa_i = 1$), for which there is a 4-approximation known along with a matching lower bound [4]. The other extreme case is when $\kappa_i = |S_i|$ for all i. This problem is essentially equivalent to a classic precedence constrained scheduling problem $1|prec|\sum_j w_j C_j$ for which there are several 2-approximations known. See [3] for pointers. However, the best approximation factor known for GMSSC is 12.4. Im et al. conjecture that GMSSC admit a 4-approximation [6].

Recommended Reading

1. Azar Y, Gamzu I (2011) Ranking with submodular valuations. In: SODA, pp 1070–1079
2. Azar Y, Gamzu I, Yin X (2009) Multiple intents re-ranking. In: STOC, pp 669–678
3. Bansal N, Gupta A, Krishnaswamy R (2010) A constant factor approximation algorithm for generalized min-sum set cover. In: SODA, pp 1539–1545
4. Feige U, Lovász L, Tetali P (2004) Approximating min sum set cover. Algorithmica 40(4):219–234
5. Im S, Nagarajan V, van der Zwaan R (2012) Minimum latency submodular cover. In: ICALP (1), pp 485–497
6. Im S, Sviridenko M, Zwaan R (2013) Preemptive and non-preemptive generalized min sum set cover. Mathematical Programming pp 1–25
7. Skutella M, Williamson DP (2011) A note on the generalized min-sum set cover problem. Oper Res Lett 39(6):433–436