Greedy Δ-approximation Algorithm for Covering with Arbitrary Constraints and Submodular Cost

Christos Koufogiannakis and Neal E. Young

University of California, Riverside

ICALP 2009

Δ-approximation for covering problems

We consider Δ-approximation algorithms for covering problems. Δ is the maximum number of variables on which any constraint depends (i.e. for vertex cover $\Delta = 2$).

covering problems:

$$\left.\begin{array}{l}\text{Vertex cover}\\ \text{Set Cover}\\ \text{Covering Integer Programs}\\ \text{Facility Location}\\ \text{Paging}\\ \text{Weighted Caching}\\ \text{Connection Caching}\\ \text{File Caching}\\ \ldots\end{array}\right\}\ \text{Monotone Cover}$$

fundamental covering problems

Vertex Cover: $\min\{c \cdot x : x_u + x_v \geq 1 \ \ (\forall (u,v) \in E); x \in \{0,1\}^n\}$.

```
1. Let $x \leftarrow \mathbf{0}$.
2. While $\exists$ edge $(u,v)$ s.t. $x_u < 1$ and $x_v < 1$ do:
3.     Raise $x_u$ at rate $1/c_u$ and $x_v$ at rate $1/c_v$
       until $x_u = 1$ or $x_v = 1$.
4. Return $\{u | \ x_u = 1\}$.
```

-Local-Ratio 2-approximation [Bar-Yehuda and Even, 1981].

-Primal-Dual 2-approximation [Hochbaum, 1982].

fundamental covering problems

Set Cover: $\min\{c \cdot x : \sum_{s \ni e} x_s \geq 1 \ \ (\forall e); x \in \{0,1\}^n\}$.

The previous Local-Ratio and Primal-Dual algorithms give straightforwardly Δ-approximation algorithms for Set Cover.

Δ is the maximum number of sets in which any element is included (element size).

fundamental covering problems

Covering Integer Programs: $\min\{c \cdot x : Ax \geq b; x \leq u; x \in \mathbb{Z}_+^n\}$.

Example

$$\begin{array}{rl} \min & x_1 \\ \text{subject to:} & 10x_1 + 10x_2 \geq 11 \\ & x_1 + 3x_2 \geq 3 \\ & x_2 \leq 1 \end{array}$$

The integrality gap can be arbitrarily large.
Reduce the gap to Δ by using the KC-inequalities. The resulting LP has exponentially many constraints!
High-degree poly-time ellipsoid-based algorithm [Carr et al., 2000][Pritchard, 2009].

hardness results

0-1 CIP not approximable to $(\Delta - 1 - \varepsilon)$ for any fixed $\varepsilon > 0$ unless P$=$NP [Dinur, 2003].

Not approximable to $(\Delta - \varepsilon)$ under the unique games conjecture [Khot and Regev, 2003].

alternatives to Δ-approximation — log-approximation

There are algorithms for several covering problems with approximation-ratio $\log n$.

Δ-approximation vs log-approximation?

The smaller among these two is the best we can hope for!

other covering problems

Covering Mixed Integer Programs (CMIP): $\min\{c \cdot x : Ax \geq b; x \leq u; x_j \in \mathbb{Z}_+ \ (j \in I), x_j \in \mathbb{R}_+ \ (j \notin I)\}$.

(non-metric) Facility Location: Given a set of facilities and a set of customers, open some facilities minimizing the sum of opening costs of the facilities plus the sum of distances from each customer to its nearest facility. Δ is the max number of facilities that might serve any given customer.

online problems

An algorithm is called *online* if it can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the beginning.

An algorithm is k-competitive if the worst case of the ratio between the cost incurred by the algorithm and the optimal cost is k.

online problems

Paging: Given an online sequence of requests for pages in a cache of size k, minimize the number of page faults.

Which page to evict from cache to make room for the newly requested page? k-competitive algorithms:

Least Recently Used (LRU) [Sleator and Tarjan, 1985].

First-In First-Out (FIFO) [Sleator and Tarjan, 1985].

Flush When Full (FWF) [Sleator and Tarjan, 1985].

The best deterministic competitive-ratio is k.

(Randomized algorithms can to better; H_k-competitive.)

online problems

Weighted Caching: A generalization of the paging problem with non-uniform eviction costs.

Which page to evict from cache to make room for the newly requested page? k-competitive algorithms:

Harmonic [Raghavan and Snir, 1989]:

Balance [Chrobak et al., 1991].

Greedy-Dual [Young, 1991].

online problems

File Caching: Generalizes weighted caching. A file f has size size(f) and eviction cost cost(f).

Which file to evict from cache to make room for the newly requested file? k-competitive algorithms:

Landlord [Cao and Irani 1997], [Young, 1998].

online problems

Connection Caching: Online sequence of requests for connections on a graph G. A request $r_i = (u_i, v_i)$ activates the connection between u_i and v_i (if the connection is not already active in both endpoints). At a request, if the node has more than k other active connections, then a connection r_q has to be closed at cost $c(r_q)$. The goal is to satisfy all requests, minimizing the total cost of closing connections.

- Any k-competitive algorithm for Weighted Caching can be converted into a $2k$-competitive algorithm for Connection Caching [Cohen et al., 1999].

- A variant of Landlord is k-competitive [Albers, 2000].

online covering problems

Online Set Cover, Online CIP, Online CMIP: Constraints are revealed one at a time; the algorithm should increase x to meet the revealed constraint without knowing the remaining constraints.

No previous Δ-competitive algorithm.

(log-competitive algorithms are known only for fractional covering programs and unweighted set cover [Buchbinder and Naor, 2005].)

covering problems

covering problems:

Vertex cover
Set Cover
Covering Integer Programs
Facility Location
Paging
Weighted Caching
Connection Caching
File Caching
. . .

covering problems

covering problems:

$$\left.\begin{array}{l}\text{Vertex cover}\\ \text{Set Cover}\\ \text{Covering Integer Programs}\\ \text{Facility Location}\\ \text{Paging}\\ \text{Weighted Caching}\\ \text{Connection Caching}\\ \text{File Caching}\\ \ldots\end{array}\right\}\ \text{Monotone Cover}$$

monotone cover

Monotone Cover:

$\min\{c(x) : (\forall S \in \mathcal{C})\ x \in S, x \in \mathbb{R}^n_+\}$.

$\mathcal{C}$ is a collection of monotone constraints.

$c : \mathbb{R}^n_+ \longrightarrow \mathbb{R}_+$ is a non-negative, non-decreasing, submodular cost function

Δ is the maximum number of variables on which any constraint depends.

Monotone Cover generalizes all the aforementioned problems.

Submodular cost functions are useful to model some problems as a Monotone Cover instance (i.e. Facility Location).

Vertex Cover as Monotone Cover

Vertex Cover:

$$\begin{array}{rl} \min & c \cdot x \\ \text{subject to:} & x_u + x_v \geq 1 \quad (\forall (u,v) \in E) \\ & x_u \in \{0,1\} \quad (u \in V) \end{array}$$

Vertex Cover as Monotone Cover:

$$\begin{array}{rl} \min & c \cdot x \\ \text{subject to:} & \lfloor x_u \rfloor + \lfloor x_v \rfloor \geq 1 \quad (\forall (u,v) \in E) \end{array}$$

Note that $\Delta = 2$.

CMIP as Monotone Cover

Covering Mixed Integer Programs (CMIP):

$$\begin{array}{rl} \min & 3x_1 + x_2 + 6x_3 \\ \text{subject to:} & 0.5x_1 + 0.3x_2 \geq 1 \\ & x_1 + 0.9x_2 + 2x_3 \geq 5 \\ & x_1 \leq 3 \\ & x_3 \leq 5 \\ & x_1, x_2 \in \mathbb{Z}_+ \\ & x_3 \in \mathbb{R}_+ \end{array}$$

CMIP as Monotone Cover:

$$\begin{array}{rl} \min & 3x_1 + x_2 + 6x_3 \\ \text{subject to:} & 0.5\lfloor \min\{x_1, 3\} \rfloor + 0.3\lfloor x_2 \rfloor \geq 1 \\ & \lfloor \min\{x_1, 3\} \rfloor + 0.9\lfloor x_2 \rfloor + 2\min\{x_3, 5\} \geq 5 \end{array}$$

Paging as Monotone Cover

Paging: Given an online sequence of requests for pages in a cache of size k, minimize the number of page faults.

Paging as Monotone Cover [Bansal et al., 2007]:

Let x_t indicate whether page r_t is evicted before the next request to r_t after time t, so the total cost is $\sum_t x_t$.

For *any* k-subset $Q = \{r_s : s < t, r_s \neq r_t\}$, at least one page $r_s \in Q$ must have been evicted (s is the time of the most recent request to r_s), so the following constraint is met $\sum_{r_s \in Q} \lfloor x_s \rfloor \geq 1$.

greedy Δ-approximation algorithm for Monotone Cover

simplified version — linear costs

```
1. Let $x \leftarrow \mathbf{0}$.
2. While $\exists$ unsatisfied constraint $S$ do:
3.     Simultaneously for all $j \in \text{vars}(S)$, increase $x_j$ at rate $1/c_j$,
       until $S$ is satisfied.
4. Return $x$   (or $x$ transformed to the right domain).
```

$\text{vars}(S)$ denotes the indexes of the variables on which constraint S depends. Note $\Delta = \max_S |\text{vars}(S)|$.

Theorem

The algorithm returns a Δ-approximate solution.

analysis for 2-approximation Vertex Cover

```
1. Let $x \leftarrow \mathbf{0}$.
2. While $\exists$ edge $(u, v)$ s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < 1$ do:
3.     Raise $x_u$ at rate $\frac{1}{c_u}$ and $x_v$ at rate $\frac{1}{c_v}$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor = 1$.
4. Return $\lfloor x \rfloor$.
```

Let x^* be any feasible solution.

Each step starts with a *non-yet-covered* edge (u, v). So $x_u^* > x_u$ or $x_v^* > x_v$.

The step increases the cost by 2β but it reduces the potential $\phi = \sum_u \max\{0, x_u^* - x_u\} c_u$ by at least $\beta \implies$ 2-approximation.

analysis for 2-approximation Vertex Cover

```
1. Let $x \leftarrow \mathbf{0}$.
2. While $\exists$ edge $(u, v)$ s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < 1$ do:
3.     Raise $x_u$ at rate $\frac{1}{c_u}$ and $x_v$ at rate $\frac{1}{c_v}$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor = 1$.
4. Return $\lfloor x \rfloor$.
```

Let x^* be any feasible solution.

Each step starts with a *non-yet-covered* edge (u, v). So $x_u^* > x_u$ or $x_v^* > x_v$.

The step increases the cost by 2β but it reduces the potential $\phi = \sum_u \max\{0, x_u^* - x_u\} c_u$ by at least $\beta \implies$ 2-approximation.

Let $\text{residual}_c(x)$ be the min cost to increase x to full feasibility. In general, the step decreases $\text{residual}_c(x)$ by at least β.

results

covering problems

CMIP: Nearly linear-time Δ-approximation. Improves over previous ellipsoid-based, high-degree poly-time algorithm for CIP.

(non-metric) Facility Location: First Δ-approximation in linear-time. Δ is the max number of facilities that might serve any given customer.

Probabilistic CMIP: First Δ-approximation in quadratic-time.

results

online problems

Online Set Cover, Online CIP, Online CMIP: First Δ-competitive algorithm.

Paging, Weighted Caching, File Caching, Connection Caching: Generalize k-competitive algorithms, i.e. Harmonic, Greedy-Dual, Landlord.

Upgradable Caching: A $(k + d)$-competitive algorithm, where k is the cache size and d is the number of upgradable hardware parameters.

Upgradable Caching

Upgradable Caching: choose not only the caching strategy, but also the hardware configuration. In response to each request, the algorithm can pay to upgrade the hardware, thus reducing the cost of later evictions. Then if the requested item is not in cache, it is brought in.

i.e. you can pay to upgrade the cpu, the bus speed, etc...

Theorem

There is a $(k + d)$-competitive algorithm, where k is the max cardinality of any cachable set at any time and d is the number of upgradable parameters.

relation to other methods

Local-Ratio

If the variables take 0/1 values and the cost function is linear, it is equivalent to our algorithm.

Our algorithm for Monotone Cover has a (less intuitive) local-ratio interpretation, using residual cost instead of weight reduction.

Primal-Dual The Primal-Dual analysis is much more complicated.

KC-inequalities must be used to reduce the integrality gap to Δ. The typical forward-greedy solution doesn't work. A "tail-recursive" dual solution is required.

conclusions

- Our algorithm generalizes existing algorithms and their analyses.
- Substantially more general than before (e.g. CIP, Upgradable Caching).
- It is important to view the problem in the right representation! Once a problem is formulated as Monotone Cover, the algorithm and analysis are very simple.

To appear in PODC 2009:

- Distributed 2-approximation algorithm for Weighted Vertex Cover in in $O(\log n)$ rounds.
- Distributed Δ-approximation algorithm for Monotone Cover in $O(\log^2 n)$ rounds.

Thank you!

Facility Location as Monotone Cover

(non-metric) Facility Location: given a set of facilities $\mathcal{F}$ and a set of customers $\mathcal{C}$, open some facilities minimizing the sum of opening costs of the facilities plus the sum of distances from each customer to its nearest facility.

(non-metric) Facility Location as Monotone Cover:

$$\min\{\textstyle\sum_j f_j \max_i x_{ij} + \sum_{ij} d_{ij}x_{ij} :$$
$$\textstyle\sum_{j \in N(i)} \lfloor x_{ij} \rfloor \geq 1 \ (\forall i \in \mathcal{C})\}$$

Δ is the maximum number of facilities that might server any given customer.