Distributed and Parallel Algorithms for Set Cover Problems with Small Neighborhood Covers

Archita Agarwal, Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, Sambuddha Roy, Yogish Sabharwal

IBM Research - India, New Delhi
Set Cover

- **Input:** Set system $\langle E, S \rangle$
  - $E : e_1, e_2, e_3, \ldots, e_m$
  - $S : \{\ldots\}, \{\ldots\}, \{\ldots\}, \ldots, \{\ldots\}$
    - $c(S_1)$, $c(S_2)$, $c(S_3)$, $c(S_n)$
- **Output:** $R \subseteq S$ having minimum cost such that all the $e \in E$ are covered
Prior Work

- **Sequential setting**
  - $O(\log \Delta)$ approximation ratio, $\Delta$ is the maximum cardinality of the sets in $S$
  - $f$ approximation ratio, where $f$ is the *frequency parameter* which is the maximum number of sets of $S$ that any element belongs to
  - $\Omega(\log m)$-inapproximable

- **Parallel setting**
  - [RV '98] $O(\log m)$ approximation ratio
  - [KVY '94] $O(f + \epsilon)$ approximation ratio

- **Distributed setting**
  - [Kuhn et al. '06] $O(\log \Delta)$ approximation ratio
  - [KY '11] $f$ approximation ratio
  - both the above algorithms run in $O(\log m)$ communication rounds
Interval Cover

- Non constant $f$ but
  - constant factor by Primal Dual
  - optimal by DP
Tree Cover

- Non constant $f$ but
  - constant factor by Primal Dual
  - optimal by DP
Other Problems with non constant $f$

- Priority Interval Cover
  - optimal by Primal Dual
- Bag Interval Cover (NP Hard)
  - constant factor by Primal Dual
Our contributions

- Introduction of SNC property for a class of set cover problems
  - A wide range of problems fall under this framework

- For set cover problems with non-constant $f$ but satisfying the SNC property, we give constant factor approximation algorithms in
  - sequential setting,
  - parallel setting, and
  - distributed setting
The neighborhood of an element \( e \in E \) is the set of all the elements with which it shares some set in \( S \):

- \( S: \{e_1, e_2\}, \{e_1, e_2, e_3\}, \{e_1, e_4, e_6\}, \{e_2, e_5\} \)
- \( S: \{e_1, e_2\}, \{e_1, e_2, e_3\}, \{e_1, e_4, e_6\}, \{e_2, e_5\} \)
- \( \text{neighborhood}(e_1): \{e_1, e_2, e_3, e_4, e_6\} \)
- \( S: \{e_1, e_2\}, \{e_1, e_2, e_3\}, \{e_1, e_4, e_6\}, \{e_2, e_5\} \)
- \( \text{neighborhood}(e_2): \{e_1, e_2, e_3, e_5\} \)

An element is a \( \tau \)-SNC element if its neighborhood can be covered by at most \( \tau \) sets:
- \( e_1 \rightarrow 3\text{-SNC} \)
- \( e_2 \rightarrow 2\text{-SNC} \)
SNC property

- For any subset $X$, the set system restricted to $X$ is defined as $\langle X, S' \rangle$ where $S' = \{ S \cap X : S \in S \}$

- For a $\tau$, if any restriction of the set system contains at least one $\tau$-SNC element, we call the set system as $\tau$-SNC set system.
Examples: Vertex Cover : 2-SNC

- Each edge (element) is incident upon 2 vertices (sets)
Examples: Interval Cover : 1-SNC

- The leftmost element is a 1-SNC element – Its neighborhood can be covered by the interval spanning it and extending most towards the right.
Examples: Interval Cover : 1-SNC

- After removing this 1-SNC element, we can find another 1-SNC element
After removing this 1-SNC element, we can find another 1-SNC element.
Examples: Interval Cover : 1-SNC

- Only after we have removed the leftmost 1-SNC element, we find a new 1-SNC element
Examples: Interval Cover : 2-SNC

- Can be viewed as a 2-SNC set system as well.
- The neighborhood of any timeslot (element) can be covered by 2 intervals (sets) - the one extending most towards the left and the one extending most towards the right.
Examples: Tree Cover : 1-SNC

- All the leaves are 1-SNC elements – Their neighborhood can be covered by the interval extending most towards the root.
Examples: Tree Cover : 1-SNC

- Any restriction of a tree is a tree in itself. After removing all the 1-SNC element, we can find other 1-SNC elements
Examples: Tree Cover : 1-SNC

- Any restriction of a tree is a tree in itself. After removing all the 1-SNC element, we can find other 1-SNC elements.
Examples: Tree Cover : 1-SNC

- Any restriction of a tree is a tree in itself. After removing all the 1-SNC element, we can find other 1-SNC elements.
Layer Decomposition

- In the given set system, find all $\tau$-SNC elements. Put them in layer $Z_1$. 
Layer Decomposition

- Restrict the set system to $E - Z_1$. Find all $\tau$-SNC elements in the restriction
Layer Decomposition

- Continue till all the elements belong to some layer $Z_k$
Layer Decomposition: Length

- Interval Cover
  - 1-SNC → $L = \Omega(m)$
  - 2-SNC → $L = 1$
Layer Decomposition: Length

- Interval Cover
  - 1-SNC $\rightarrow L = \Omega(m)$
  - 2-SNC $\rightarrow L = 1$

- Tree Cover
  - 1-SNC $\rightarrow L = \Omega(m)$
  - Theorem: For a 2-SNC tree cover system, the number of layers $L = O(\log m)$
Layer Decomposition: Length

There exists a procedure for computing the layer decomposition of a given \(\tau\)-SNC set system.

- In the sequential setting, it can be implemented in polynomial time.
- In the distributed setting, it can be implemented in \(O(L)\) communication rounds.
- In the parallel setting, the algorithm takes \(L\) iterations each of which can be implemented in NC.

- For parallel and distributed setting the decomposition length \(L\) should be \(O(\log m)\)
Our Results

- \( \tau \) sequential approximation algorithm for any \( \tau \)-SNC set system
- \( 8\tau^2 \) parallel approximation algorithm for a \( \tau \)-SNC set system of logarithmic length
- \( \tau \) distributed approximation algorithm for a \( \tau \)-SNC set system of logarithmic length
Primal Dual Framework

Primal
\[
\begin{align*}
\min \quad & \sum_{S \in \mathcal{S}} x(S) \cdot w(S) \\
\sum_{S \in \mathcal{S} : e \in S} x(S) & \geq 1 \quad (\forall e \in E) \\
x(S) & \geq 0 \quad (\forall S \in \mathcal{S})
\end{align*}
\]

Dual
\[
\begin{align*}
\max \quad & \sum_{e \in \mathcal{E}} \alpha(e) \\
\sum_{e \in \mathcal{S}} \alpha(e) & \leq w(S) \quad (\forall S \in \mathcal{S}) \\
\alpha(e) & \geq 0 \quad (\forall e \in \mathcal{E})
\end{align*}
\]
Primal Dual Framework

A primal-dual algorithm would generally have
- Forward Phase
- Reverse Delete Phase

**Forward Phase**
- Increase the dual variables
- If some constraint becomes tight, pick the variable corresponding to that constraint in the primal solution
- Output \( \langle A, \alpha \rangle \), where \( A \) and \( \alpha \) are the primal and dual solutions respectively

**Reverse Delete Phase**
- Take the primal solution \( A \) produced in the forward phase and drop redundant items from it
- Output \( \langle B, \alpha \rangle \), where \( B \subseteq A \)
We say that the pair \( \langle A, \alpha \rangle \) is \( \lambda \)-maximal, if for any \( S \in A \), the corresponding dual constraint is approximately tight:

\[
\sum_{e \in S} \alpha(e) \geq \lambda \cdot w(S)
\]

We say that the pair \( \langle B, \alpha \rangle \) satisfies the primal slackness conditions approximately, if for any \( e \in E \), if \( \alpha(e) > 0 \) then

\[
|\{ S \in B : S \text{ covers } e \}| \leq \mu.
\]

The overall factor achieved is \( \frac{1}{\lambda} \cdot \mu \)
Primal Dual Framework: Forward Phase

**Dual**

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} \alpha(e) \\
\sum_{e \in S} \alpha(e) & \leq w(S) \quad (\forall S \in S) \\
\alpha(e) & \geq 0 \quad (\forall e \in E)
\end{align*}
\]
$O(\tau)$ sequential algorithm: Forward Phase

- Arrange the elements according to layer decomposition order
- Scan them from left to right

\[
\sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in S)
\]
$O(\tau)$ sequential algorithm: Forward Phase

- Raise the first uncovered element to the maximum possible cost

$\sum_{e \in S} \alpha(e) \leq w(S)$ \hspace{1cm} (\forall S \in S)
\( O(\tau) \) sequential algorithm: Forward Phase

- Include the \textit{tight} set in the solution
- Remove the elements that got \textit{freely} covered by this picked up set

\[
\sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in \mathcal{S})
\]

50
\[e_1 \quad e_2 \quad e_3\]

70
\[e_4 \quad e_6\]

90
\[\not{e_1}\]

40
\[e_7 \quad e_8\]

40
\[e_6 \quad e_8\]

20
\[e_4\]

90
\[e_5 \quad e_6 \quad e_8\]

Cost = 50

Raise \( e_1 = \min(50, 90) \)
$O(\tau)$ sequential algorithm: Forward Phase

- Look for the next uncovered element in the ordering and repeat the process

\[
\sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in S)
\]
**$O(\tau)$ sequential algorithm: Forward Phase**

**Ordering by LD**

- **e₁**
- **e₂**
- **e₃**
- **e₄**
- **e₅**
- **e₆**
- **e₇**
- **e₈**

- **Cost = 70**
- **Raise e₄ = min(70, 20)**

\[ \sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in S) \]
$O(\tau)$ sequential algorithm: Forward Phase

Ordering by LD

$\sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in S)$
$O(\tau)$ sequential algorithm: Forward Phase

Ordering by LD

$\sum_{e \in S} \alpha(e) \leq w(S) \ (\forall S \in S)$
$O(\tau)$ sequential algorithm: Forward Phase

Ordering by LD

$\sum_{e \in S} \alpha(e) \leq w(S) \quad (\forall S \in S)$
**$O(\tau)$ sequential algorithm: Reverse Delete Phase**

- Only look at the elements *raised* in the forward phase
- Arrange them in the order they were raised and scan them in the reverse direction

![Diagram showing reverse delete phase](image)
It’s possible that more than $\tau$ sets are spanning an element raised in the forward phase.
$O(\tau)$ sequential algorithm: Reverse Delete Phase

- Bad!! *Compress them!!*
- Choose at most $\tau$ out of them

Choose $\tau$ out of the them
After the compression, can we still guarantee coverage?

Elements to its left are not dependent on it for their coverage.

No set picked up in $i - 1$ iterations can span $e_i$ else $e_i$ wouldn’t have been raised in the forward phase.
What about the elements to its right?

- \( e_i \) is \( \tau \) SNC in the restriction to its right (Layer decomposition)

- \( \lambda = 1 \), and

- \( \mu = \tau \). Hence, we have \( O(\tau) \) approximation algorithm
Parallel Algorithm: Comparison with [KVY '94]

Constant $f$ sequential setting
- Raise one variable at a time
- Produce maximal solutions ($\lambda = 1$)
- # of iterations $= \Omega(m)$

[KVY'94] parallel setting
- Raise multiple variables simultaneously
- Produce near maximal solutions ($\lambda = 1 - \epsilon$)
- # of iterations $= O\left(\frac{1}{\epsilon} f \log m\right)$

Our Results in Parallel setting
- Go according to layer decomposition. Inside each layer,
  - Raise multiple variables simultaneously
  - Produce $(1/8)$ maximal solutions (worse than [KVY])
  - # of iterations $O(\log m)$ (got rid of $f$)
- Total # of iterations $O(L \log m)$
Parallel Algorithm: Reverse Delete

- Complicated!
- $\mu = \tau^2$
- # of iterations $O(L^2)$
Open problems in parallel setting

- Approximation Ratio: $O(\tau^2) \rightarrow O(\tau)$
- # of iterations: $O(L^2) \rightarrow O(L)$
- Find algorithms for $\tau$-SNC systems beyond logarithmic length
Thank you!