Efficient Parallel Algorithm for Bi-core Decomposition

Claire Wang
cwang23@andover.edu

Michael (Yihao) Huang
yhuang23@andover.edu

PRIMES 2021
Mentored by: Jessica Shi, Julian Shun
Graphs

Motivation

- a vertex represents an object of interest in a study or dataset
- an edge represents a relationship between two vertices.

Dense Subgraph Discovery

Motivation

Bipartite Graphs

Motivation

- A graph $G$ made up of two mutually exclusive sets of vertices with edges that connect them
- Model the relationship between two groups

Authorship graph

Diseases and their correlated lncRNA loci

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0087797
(α, β)-core

(3, 2) core means that every U node has at least 3 edges and every V node has at least 2 edges within the subgraph.

Alpha and beta maxes.
Fraudster Detection

Applications of Bi-core Decomposition

$\alpha, U$

High $\beta$ low $\alpha$

↓

Suspicious and flagged

$\beta, V$
Parallelism

Increase in size of graphs and # of cores.

© 2019 Julian Shun Slide adapted from 6.172 (Charles Leiserson and Susan Amoratogos)
Preliminaries
Work-span Model

Preliminary

\[ T_p = \text{Runtime with } p \text{ processors} \]
\[ T_1 = \text{Work} \]
\[ T_\infty = \text{Span} \]

Brent’s Law:

\[ T_p \leq T_\infty + \frac{T_1 - T_\infty}{p} \]

Work Efficiency: same Work Complexity as the best sequential algorithm
Bi-core Decomposition

Goal: find $\alpha_{\text{max}}(\beta(v))$ for every $\beta$ and $v$ and find $\beta_{\text{max}}(\alpha(u))$ for every $\alpha$ and $u$

Process: Peeling-based—remove vertices with min degree—repeat until empty

For $\beta = 1$ to $\delta$:
Peel from $\alpha = 1$ to its maximum value

For $\alpha = 1$ to $\delta$:
Peel from $\beta = 1$ to its maximum value

Sequential Bi-core Decomposition

In the yellow, U partition find the vertex with minimum induced deg

For each such vertex:

Delete it

Update blue vertex degree

Check the blue partition for vertices with degree < β

For each blue node < β:

Delete node

Update yellow vertex degree

Update yellow

Finding all cores up to the (4,2) core

Sequential Bi-core Decomposition

After peeling one side, we peel the other.

\[ V, \beta = 2 \]

\[ U, \alpha = 3 \]

In the yellow, \( U \) partition find the vertex with minimum induced deg.

For each such vertex:
- Delete it
- Update blue vertex degree

Check the blue partition for vertices with degree < \( \beta \).

For each blue node < \( \beta \):
- Delete node
- Update yellow vertex degree
- Update yellow

Sequential Bi-core Decomposition

In the yellow, \(U\) partition find the vertex with minimum induced deg

For each such vertex:
- Delete it
- Update blue vertex degree

Check the blue partition for vertices with degree < \(\beta\)

For each blue node < \(\beta\):
- Delete node
- Update yellow vertex degree

Update yellow

We have computed the (4,2) core

Algorithm
In the **yellow**, U partition find all vertices with minimum induced deg

Par for each such vertex:
- Delete it
- Update blue neighbor vertex’s degree in parallel

Obtain vertices in **blue**, V partition with degree < β

Par for each blue node < β:
- Delete node
- Update yellow vertex degree in parallel

---

**Parallel Bi-core Decomposition**

<table>
<thead>
<tr>
<th>U, α = 1</th>
<th>V, β = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>

Finding all cores up to the (4,2) core

---

Parallel Bi-core Decomposition

V, $\beta = 2$

After peeling one side, we peel the other

U, $\alpha = 3$

In the yellow, U partition find all vertices with minimum induced deg

Parfor each such vertex:

- Delete it
- Update blue neighbor vertex’s degree in parallel

Obtain vertices in blue, V partition with degree < $\beta$

Parfor each blue node < $\beta$:

- Delete node
- Update yellow vertex degree in parallel

Parallel Bi-core Decomposition

We have computed the (4, 2) core

\[ V, \beta = 2 \]

In the yellow, \( U \) partition find all vertices with minimum induced deg

Parfor each such vertex:
- Delete it
- Update blue neighbor vertex’s degree in parallel

Obtain vertices in blue, \( V \) partition with degree < \( \beta \)

Parfor each blue node < \( \beta \):
- Delete node
- Update yellow vertex degree in parallel

## Complexity Results

<table>
<thead>
<tr>
<th>Work</th>
<th>Liu et al.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(\delta m)$ or $O(m^{1.5})$</td>
<td>$O(\delta m)$ or $O(m^{1.5})$</td>
</tr>
<tr>
<td>Span</td>
<td>$O(m)$</td>
<td>$O(\rho \log n)$</td>
</tr>
</tbody>
</table>

$(\alpha, \beta)$-core decomposition is P-complete when $\alpha \geq 3$ or $\beta \geq 3$
Peeling-space Pruning

Optimization
Evaluation

• 30-core, 2-way hyperthreading, CPU @ 3.1 GHz
• has 60 vCPUs and 240 GB of memory
• We used the GBBS (graph based benchmark suite) to implement our parallel code
• Graphs were from the KONECT graph database
• Largest graph run: orkut (327 million edges)

| Graph Name  | Type        | |U|   | |V|   | n   | m   | dmax | δ    | ρmax |
|-------------|-------------|-------------------|------|-------------------|------|------|------|------|------|------|------|
| Orkut       | Membership  | 2.78M             | 8.73M| 11.51M            | 327M | 318K | 466  | 12100|
| Web Trackers| Inclusion   | 27.7M             | 284K | 40.43M            | 140.6M| 11.57M| 437  | 4542 |
| LiveJournal | Membership S| 3.20M             | 7.49M| 13.89M            | 112M | 1.05M| 108  | 6831 |
| TREC        | Inclusion   | 556K              | 1.17M| 1.73M             | 83.6M| 457K | 508  | 6029 |
| Reuters     | Inclusion   | 781K              | 284K | 1.06M             | 60.6M| 345K | 192  | 4767 |
| Epinions    | Rating      | 120K              | 755K | 880k              | 13.67M| 162K | 151  | 3049 |
| Flickr      | Membership  | 396K              | 104K | 500k              | 8.55M| 35K  | 147  | 2300 |

Table 2. Graphs Statistics

KONECT -- The Koblenz Network Collection. Jerone Kunegis 2013. konect.cc/networks

Theoretically Efficient Parallel Graph Algorithms Can Be Fast and Scalable: https://github.com/ParAlg/gbbs, 2018
**Runtime comparison**

- 4.1x speedup over Liu et al.’s parallelization
- 16.2—35.5x self-relative speedup

![Graph showing sequential vs parallel run times with data points for Orkut, Web Trackers, TREC, Livejournal, Reuters, Epinions, and Flickr. The y-axis represents the run times on a log scale.]
Parallel Speedup for different graphs (self-relative ratios).
Conclusion

• A work-efficient shared memory algorithm that improves upon the span of previous work
• We achieve 35.5x max self-relative speedup
• Github: https://github.com/clairebookworm/gbbs

Future Work

• Dynamic bi-core peeling
• Extrapolate to bi-clique decomposition (which is a generalization of butterfly decomposition)
• Study the tradeoff between work-efficiency and practical speed
Acknowledgements

We’d like to thank Jessica Shi and Prof. Julian Shun for their support and mentorship, as well as the MIT PRIMES program for this opportunity.

Any questions?