Towards Parallel Graph Mining in Distributed Memory Environments using Task-Parallel Language Tascell

Shingo Okuno[†1, a)], Tasuku Hiraishi[†1], Hiroshi Nakashima[†1], Masahiro Yasugi[†2], Jun Sese[†3]

†1: Kyoto University, †2: Kyushu Institute of Technology, †3: AIST, a): shingo@sys.i.kyoto-u.ac.jp

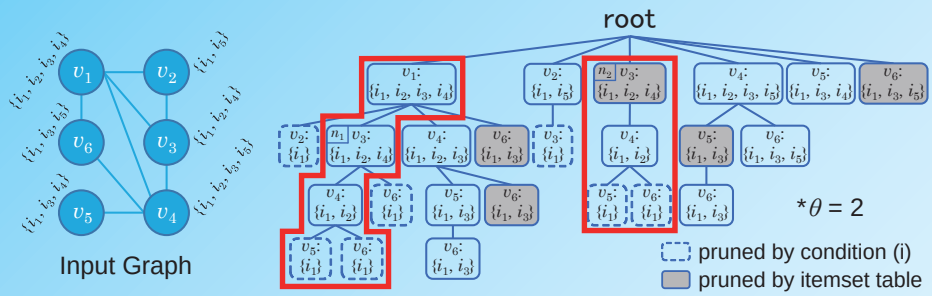
Problem Definition and Algorithm

Problem definition

- Input: graph $G = (V, E)$, set of items I, items associated with each vertex $\mathfrak{I}(v) \in \mathfrak{P}(I)$, and threshold θ
- Output: all connected subgraphs $G' = (V', E')$ of G that satisfies the following conditions:

(i) $\left| \bigcap_{v \in V'} \mathfrak{I}(v) \right| \geq \theta$

(ii) $\left| \bigcap_{v \in V' \cup \{v'\}} \mathfrak{I}(v) \right| < \left| \bigcap_{v \in V'} \mathfrak{I}(v) \right|$ for any v' connected to G'



COPINE algorithm

- An efficient backtrack search algorithm employing pruning to extract the desired subgraphs [J. Sese *et al.*, 2010.]
- We designed the parallel algorithm as an extension of COPINE to avoid excessive pruning [S. Okuno *et al.*, 2014.]
- We divide the search tree and assign a set of subtrees to each worker running in parallel

Itemset Table

When a vertex is added to the subgraph we are visiting...

- the common itemset of the resulting subgraph is added to the entry corresponding to the added vertex
- If a super-itemset exists, the search of the descendants can be skipped

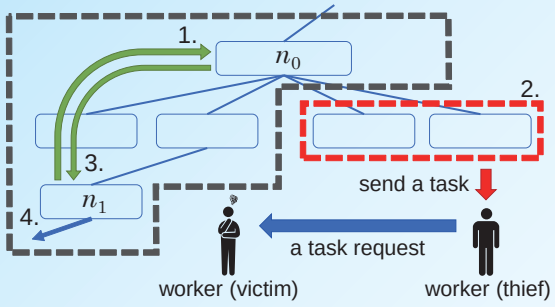
Itemset table before n_2 is visited

vertex	itemset
v_1	$\{i_1, i_2, i_3, i_4\}$
v_2	$\{i_1, i_5\}$
v_3	$\{i_1, i_2, i_4\}$
v_4	$\{i_1, i_2, i_3\}$
v_5	$\{i_1, i_3\}$
v_6	$\{i_1, i_3\}$

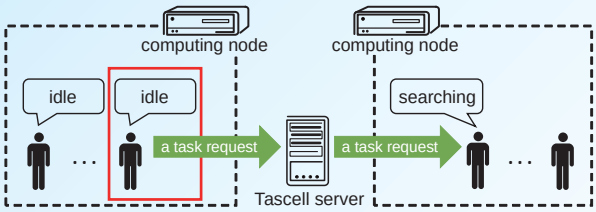
Parallel Implementation using the task-parallel language Tascell

Tascell†

- Parallel language (extended C) for irregular applications and enviornments [T. Hiraishi *et al.*, 2009.]
- A worker executes its own task sequentially and does not create any *logical threads* (c.f. *Lazy Task Creation*)
- When a worker receives a task request from another worker,
  1. it backtracks to the oldest-spawnable point,
  2. spawns a task to traverse the right subtree,
  3. returns from backtracking, and
  4. resumes its own computation
- We can delay copying between workspaces and reuse a single workspace while a sequential computation



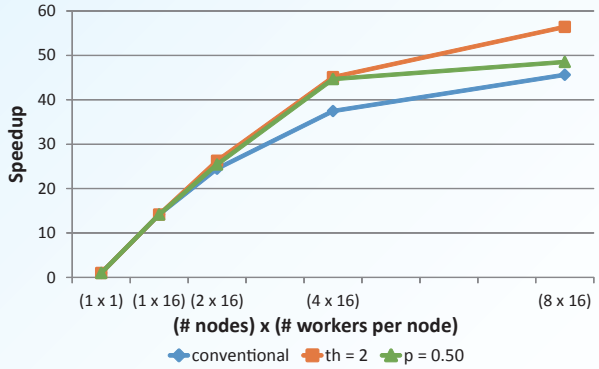
† **http://super.para.media.kyoto-u.ac.jp/tascell/**



Performance evaluation

- Intel Xeon E5 2.6GHz 8-core x 2 / node
- InfiniBand FDR x 2, 13.56 GB/s
- Parallel search without itemset table
- Input: a real protein network, and $\theta = 12$

$|V| = 15227$, $|E| = 225458$, $|I| = 158$, diameter: 12, average degree: 29.6, each node has 9.42 items in average



Work-stealing strategy

- Internode work-steals are relayed by Tascell servers
- The conventional work-stealing strategy in Tascell, which aims to minimize the number of internode work-steals, degrades the performance
- We implemented internode work-stealing strategies that promote workers to request tasks to external nodes to increase the opportunity to obtain large tasks
  - Sending a task request to an external node when # external tasks < th
  - Sending a task request to an external node with probability p

Future work

- An issue is how workers efficiently share information in the itemset table among workers
- All workers in a computing node share the single itemset table with a lock for each table entry for mutual exclusion
- We are implementing a sharing method where the difference of table information between computing nodes is sent when an internode work-steal occurs