Community Branching for Parallel Portfolio SAT Solvers

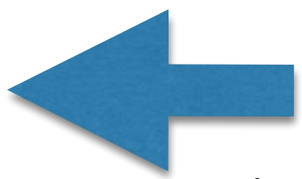
Tomohiro SONOBE*, Shuya KONDOH**, Mary INABA**
*National Institute of Informatics, Japan
**University of Tokyo, Japan

Outline

- Background
  - portfolio and diversification
- Proposal: community branching
  - A new type of diversification
- Experiment

Background

SAT solver
(for application instances)

- Sequential
  - DPLL + clause learning + backjump + VSIDS + restart + preprocess + phase saving + LBD + etc.
- Parallel
  - Using the sequential solvers as workers
  - Two major strategies:
    - Divide-and-conquer
    - Portfolio  our target

Portfolio

- Principle: let several differentiated but related solvers compete and cooperate to be the first to solve a given instance.
  - No effects in running same solvers as workers
- Diversification
  - Differentiation of each solver

Existing diversification

- Differentiation of search parameters of each worker
  - e.g. decision heuristic, learning scheme, frequency of restart, etc.
- Difficult to combine the parameters properly in order to avoid overlaps of search spaces between the workers

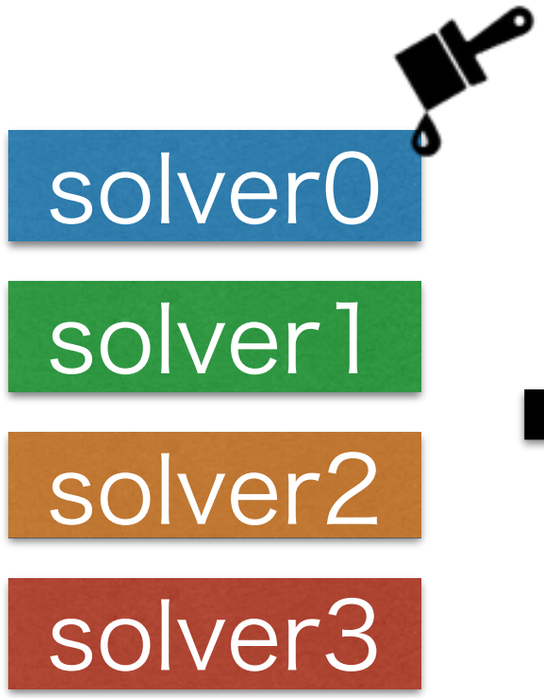
Our challenge

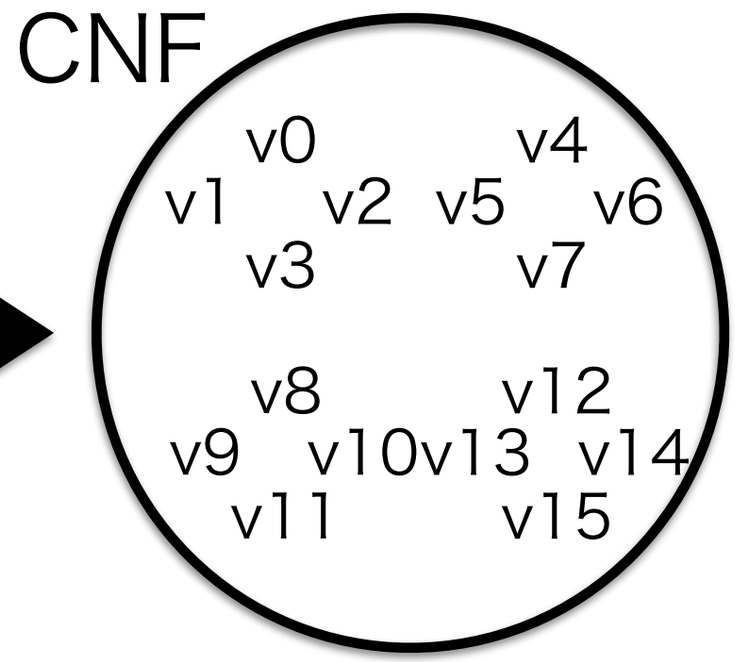
- Achieve a strong diversification by differentiating target variables of each worker
  - Target variables: each worker prioritizes them as decision variables

- A parallel search for different sets of the target variables can avoid the overlaps of the search spaces more vigorously and more easily

Image

existing diversification

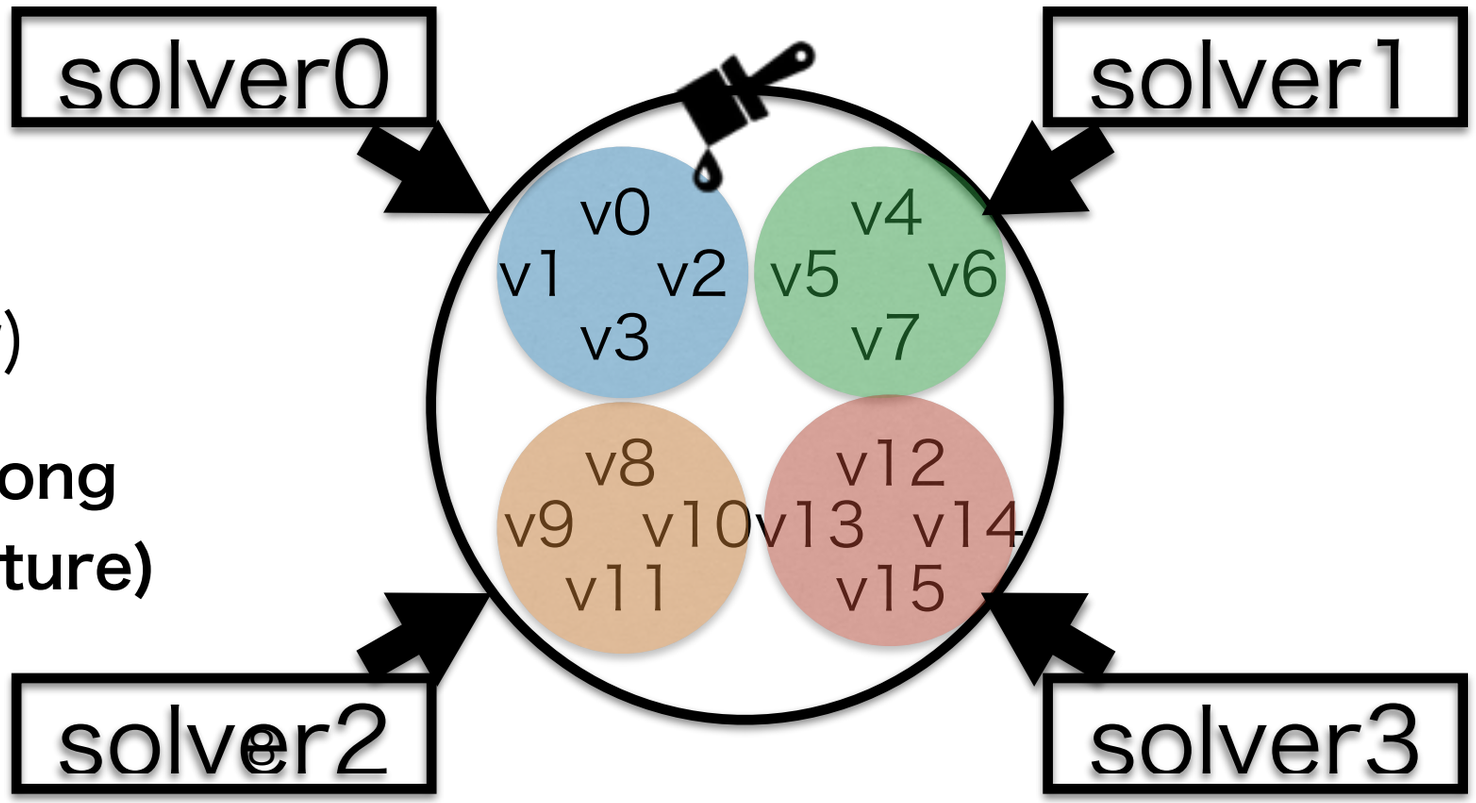




our diversification

(solvers should have a color)

variables in a cluster have strong relationships each other (structure)



Proposal

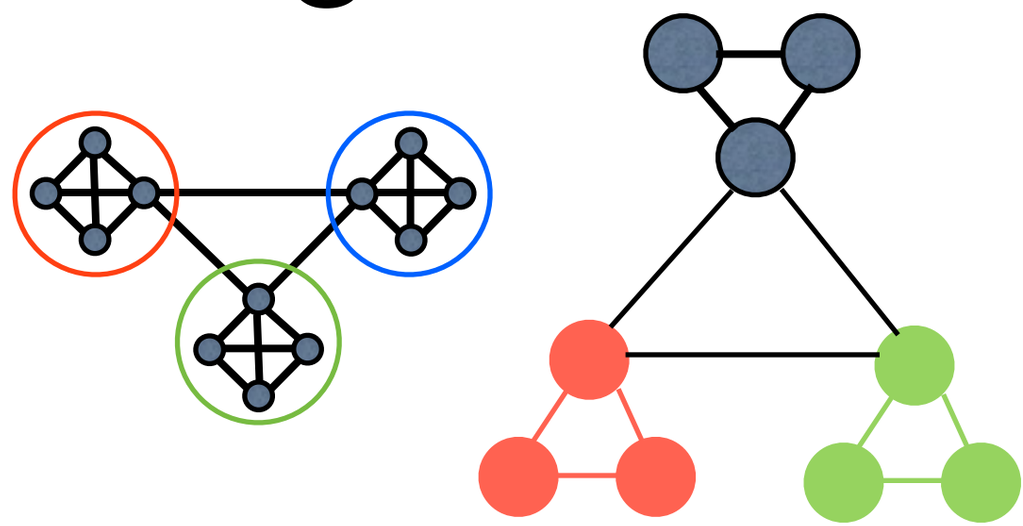
A new type of diversification

- Existing diversification: focusing on differentiating search activities of the workers
  - By differentiating the search parameters
- Our diversification: focusing on differentiating search spaces of the workers
  - By differentiating the target variables

How to organize the target variables?

- Random choice?
  - Possible to diversify the search but can be inefficient
- Something based on the structure of the application instance
  - Community detection algorithm

Community detection algorithm

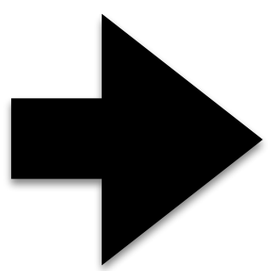


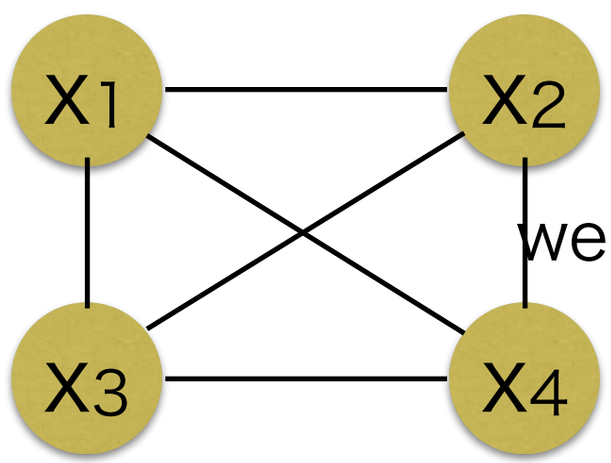
- Graph clustering
  - High edge density in a community
  - Low edge density between communities
- CNF -> Variable Incidence Graph (VIG)

Variable Incidence Graph (VIG) [C. Ansótegui et al., 2012]

- Node: variable
- Edge: existence of a clause relating two variables
  - $(x_1 \vee \ldots \vee x_n)$ results into ${}_nC_2$ edges.
  - A weight for an edge: $1/{}_nC_2$

$(x_1 \vee x_2 \vee x_3 \vee x_4)$





Community branching

- Procedure:
    - A worker conducts the community detection and assigns the detected communities to each worker.
        - Each worker conducts intensive searches for the assigned communities.
- We can achieve an efficient diversification easily, not considering the combination of the search parameters.
    - Focusing on the differentiation of the search spaces

Pseudo code

```
assign_communities() {
  coms = community_detection(convert_CNF_to_VIG(given_CNF
+ learnt_clauses));
  sort_by_size_in_descending_order(coms);
  for(i = 0; i < coms.size(); i++) {
    worker_id = i % worker_num;
    assign(coms[i], worker_id);
  }
}
```

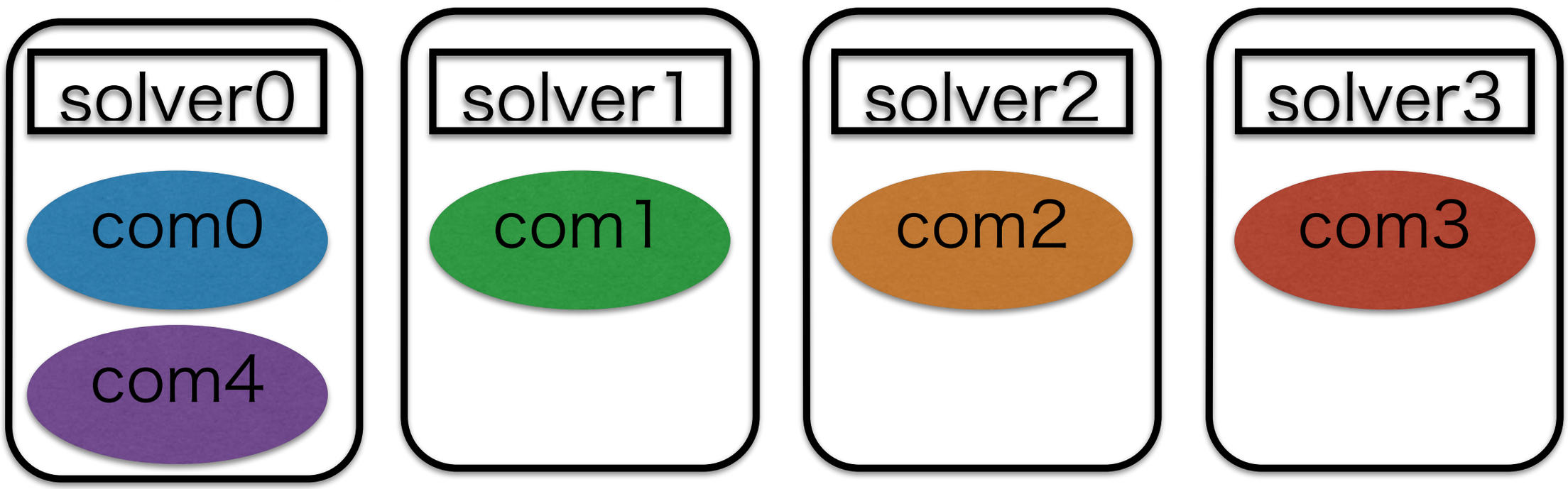
We will reconstruct the communities by using learnt clauses.

```
run_count = 0;
community_branching() {
  if (run_count++ % INTERVAL > 0)
    return;
  [choose one of the assigned communities in rotation]
  for each var in the chosen community
    bump_VSIDS_score(var, BUMP_RATIO);
}
```

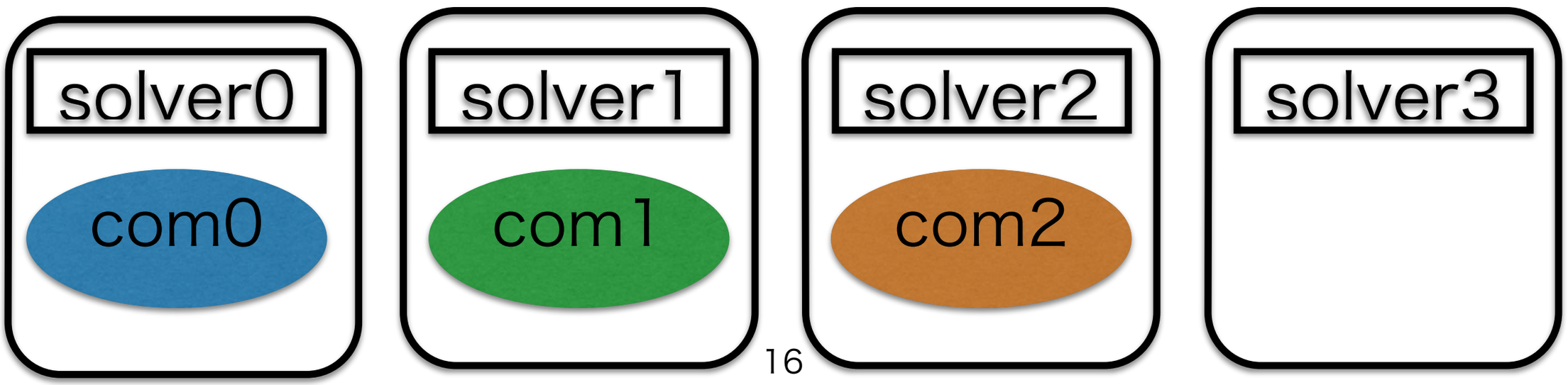
called for every restart

of communities vs. # of workers

community > worker



community < worker



Experiment

Experimental settings #1

- Benchmark: 300 instances from SAT Competition 2011 application track
  - Timeout for each instance: 5000 wall-clock seconds
  - Each instance was solved twice and shorter time was selected.
    - Instances not solved by any solvers were not included in the following results.
- Solver: PeneLoPe and parallelized version MiniSAT 2.2 (ParaMiniSAT)
- Community detection algorithm: graph folding algorithm (Louvain method) [V. D. Blondel et al., 2008]
  - Time-efficient in preliminary experiments(up to 100 seconds)

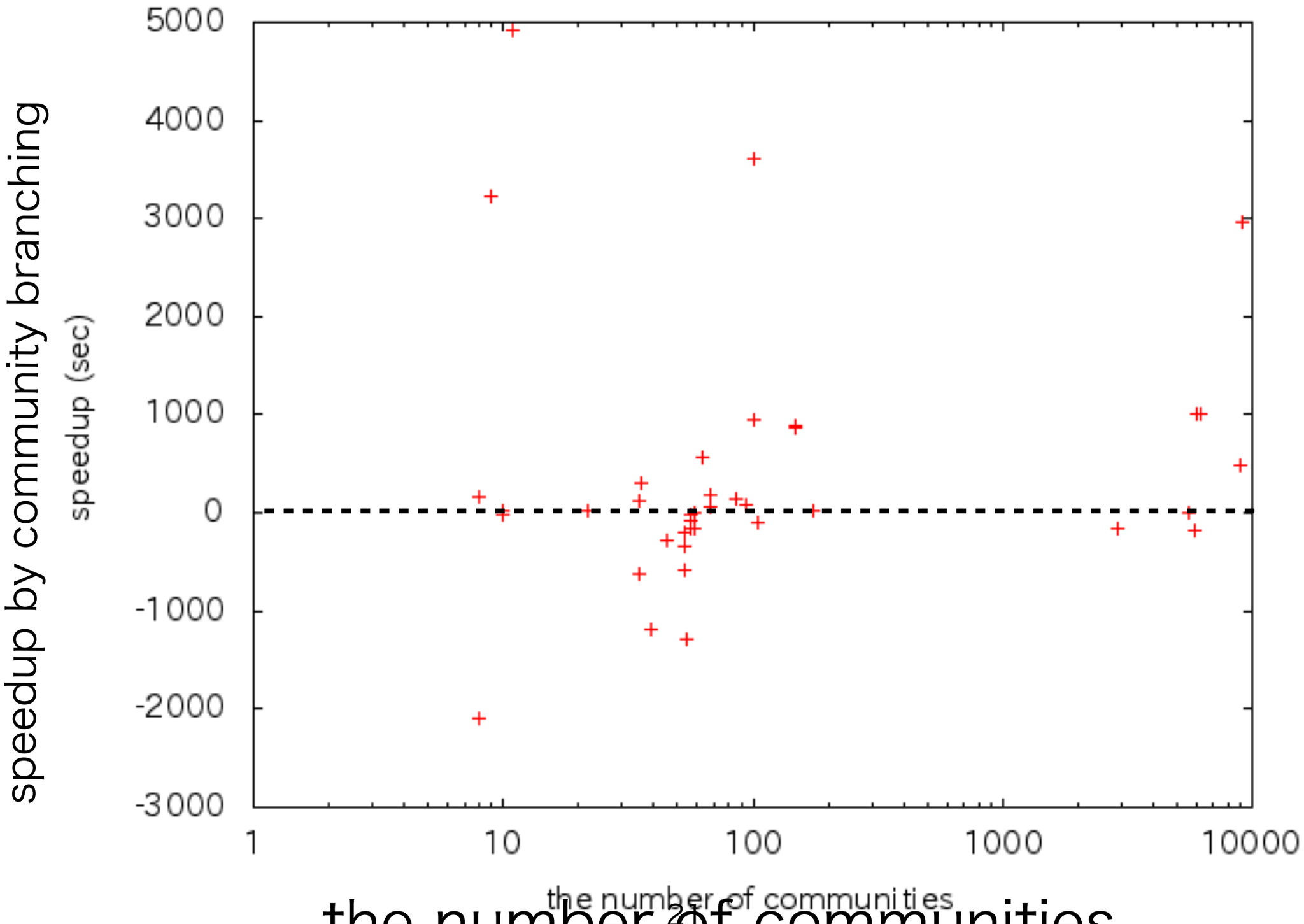
Experimental settings #2

- PC: two Intel Xeon six-core CPUs @3.33GHz, 144 GB of RAM
  - 8 and 12 threads
- Parameter settings
  - INTERVAL and BUMP_RATIO: chosen from preliminary experiments
  - Community reconstruction interval: every 3000 (or 500) restarts
  - Others: default settings
    - Clause sharing up to length 8 in ParaMiniSAT

Over all result: PeneLoPe@8threads with deterministic mode

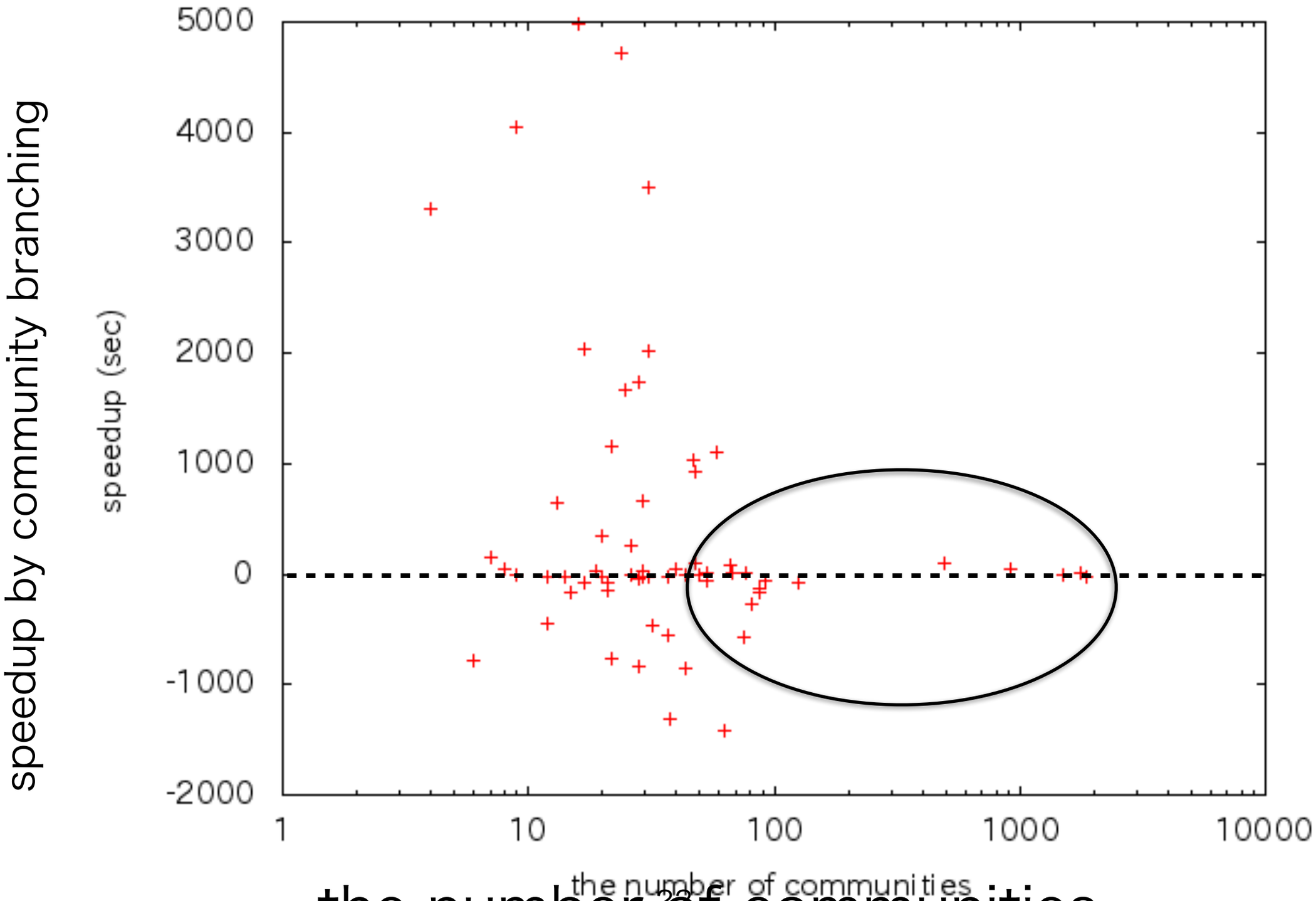
	SAT(105) (time)	UNSAT(126) (time)	total (time)
no_cb	103 (59660)	116 (128480)	219 (188140)
cri500_int1_bump100	103 (52000)	122 (104570)	225 (156570)
cri3000_int1_bump100	104 (47420)	126 (103450)	230 (150870)

of communities and speedup for SAT instances (over 100 seconds)



the number of communities

of communities and speedup for UNSAT instances (over 100 seconds)



the number of communities

Over all result: PeneLoPe@8threads without deterministic mode

	SAT(110) (time)	UNSAT(132) (time)	total (time)
no_cb	109 (32440)	126 (100680)	235 (133120)
cri3000_int1_ bump100	108 (42910)	131 (82350)	239 (125260)

Over all result: ParaMiniSAT @8 and 12threads

	SAT(111) (time)	UNSAT(135) (time)	total (time)
no_cb_t8	110 (33790)	122 (131640)	232 (165430)
no_cb_t12	110 (32420)	125 (11550)	235 (147920)
cri3000_int3_ bump10000_t8	109 (42630)	130 (107890)	239 (150520)
cri3000_int3_ bump10000_t12	109 (32700)	129 (102330)	238 (135030)

Future work

- Dynamic parameters
  - e.g. according to the number of detected communities and workers
- Other community detection algorithms
  - The number of detected communities can be different.
- Scalability
  - In case of workers > communities, we should divide a community into multiple communities.

Summary

- A new type of diversification: differentiating search spaces between workers
  - By differentiating target variables
- Community branching can boost the search efficiency in a simple manner.
  - Possible to coexist (cooperate) with existing diversification techniques