Pushing Forward Marginal MAP with Best-First Search: Auxiliary Material

Radu Marinescu
IBM Research – Ireland
radu.marinescu@ie.ibm.com

Rina Dechter and **Alexander Ihler**
University of California, Irvine
Irvine, CA 92697, USA
{dechter,ihler}@ics.uci.edu

1 Introduction

This document includes supplementary material for the following IJCAI-2015 paper: "Pushing Forward Marginal MAP with Best-First Search".

Section 2 contains additional experimental results and some implementation details of the search algorithms described in the main paper.

2 Detailed Experimental Evaluation

In the header of the summary plots (for each benchmark class), **wc** denotes the *constrained induced width* and **wu** denotes the *unconstrained induced width*, respectively.

For each problem instance, we generated 4 Marginal MAP problems, as follows, where 50% of the variables were selected as MAP variables:

- v0 - MAP variables are the first M variables from a breadth-first traversal of a minfill based pseudo tree; designed such that the constrained and unconstrained elimination orderings are relatively close to each other
- vH - MAP variables are the first M variables from a breadth-first traversal of a hypergraph decomposition based pseudo tree; designed such that the constrained and unconstrained elimination orderings are relatively close to each other
- v1 - MAP variables selected uniformly at random. The MAP instances generated this way tend to have very large constrained induced widths.
- v2 - similar to 'v1'.

Note that for each problem instance there should be 4 corresponding plots (v0, v1, v2, vH). However, if none of the algorithms could solve a problem instance within the 1 hour time limit, then we don't generate that respective plot.

Algorithms:

- `BBBTi`: BnB using incremental mini-cluster-tree heuristics (MCTE) along unconstrained ordering; search is restricted to a static variable ordering; SUM subproblems are solved by AND/OR search with full/adaptive caching;
- `BBBTd`: BnB using dynamic mini-cluster-tree heuristics (MCTE); search uses dynamic variable ordering heuristics; SUM subproblems are solved by AND/OR search with full/adaptive caching;
- `AOBB-MM`: AND/OR Branch-and-Bound using the static WMB-MM(i) heuristics (one pass moment matching).
- `AOBF-MM`: new Best-First AND/OR search using the static WMB-MM(i) heuristics (one pass moment matching).
- `RBFAOO-MM`: new Recursive Best-First AND/OR search using the static WMB-MM(i) heuristics (one pass moment matching).
- `AOBB-JG`: AND/OR Branch-and-Bound using the static WMB-JG(i) heuristics (join-graph cost shifting).
- `AOBF-JG`: new Best-First AND/OR search using the static WMB-JG(i) heuristics (join-graph cost shifting).
- `RBFAOO-JG`: new Recursive Best-First AND/OR search using the static WMB-JG(i) heuristics (join-graph cost shifting).
- `A*`: new A* using dynamic mini-cluster-tree heuristics (MCTE); SUM subproblems are solved by AND/OR search with full/adaptive caching;

2.1 Results for `pedigree` networks

Figure 1 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the pedigree benchmark (includes all instances).

Figure 2 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the `easy` as well as the `hard` instances of the pedigree benchmark.

Figures 3 through 9 plot the CPU time and number of nodes for for solving pedigree instances pedigree1—pedigree39.

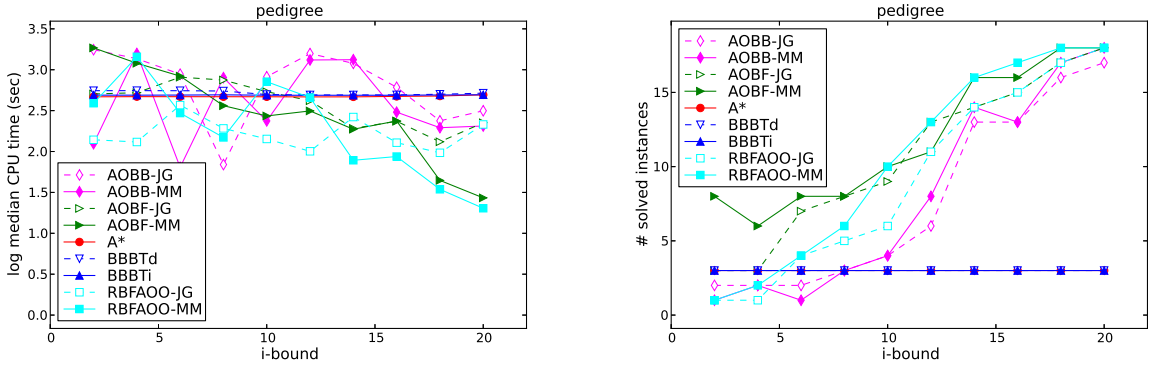

Figure 1: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `pedigree` benchmark.

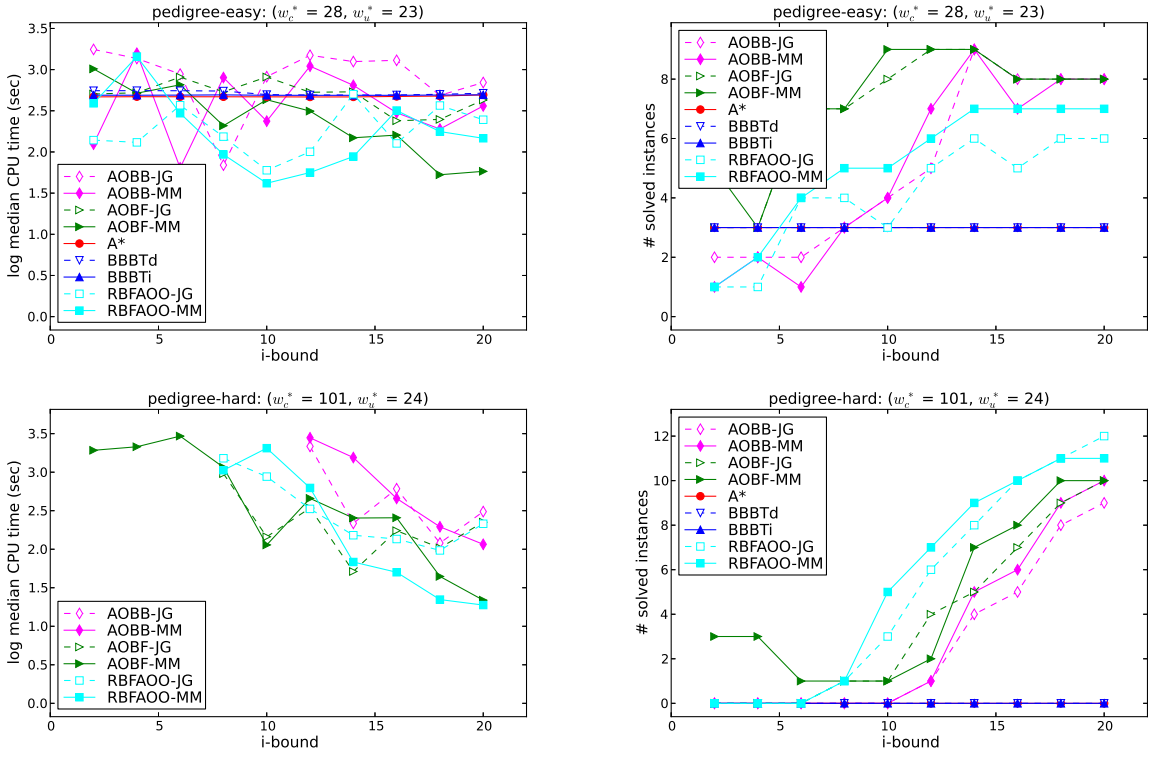

Figure 2: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `pedigree-easy` and `pedigree-hard` benchmark.

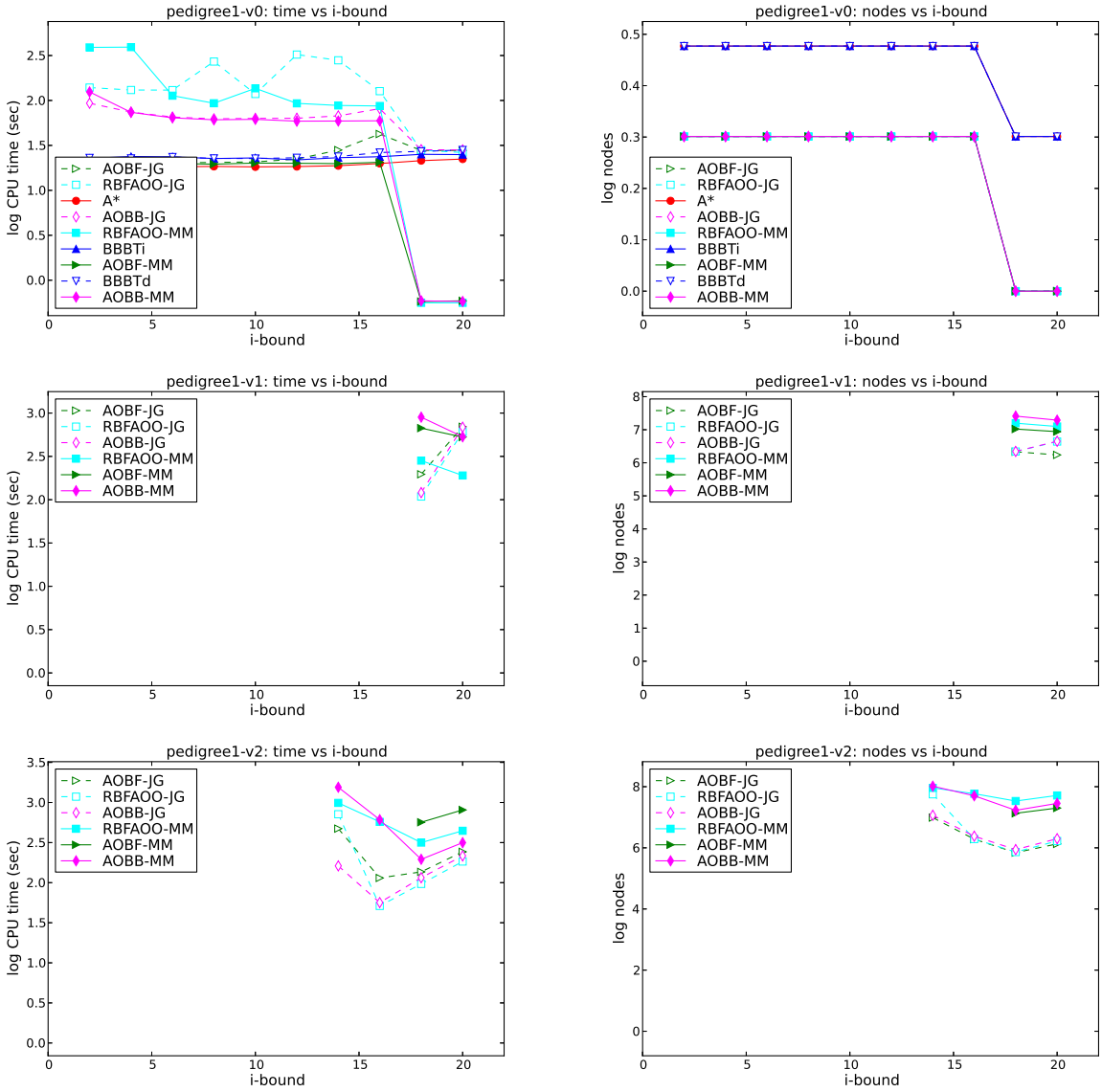

Figure 3: pedigree1 instance (time and nodes)

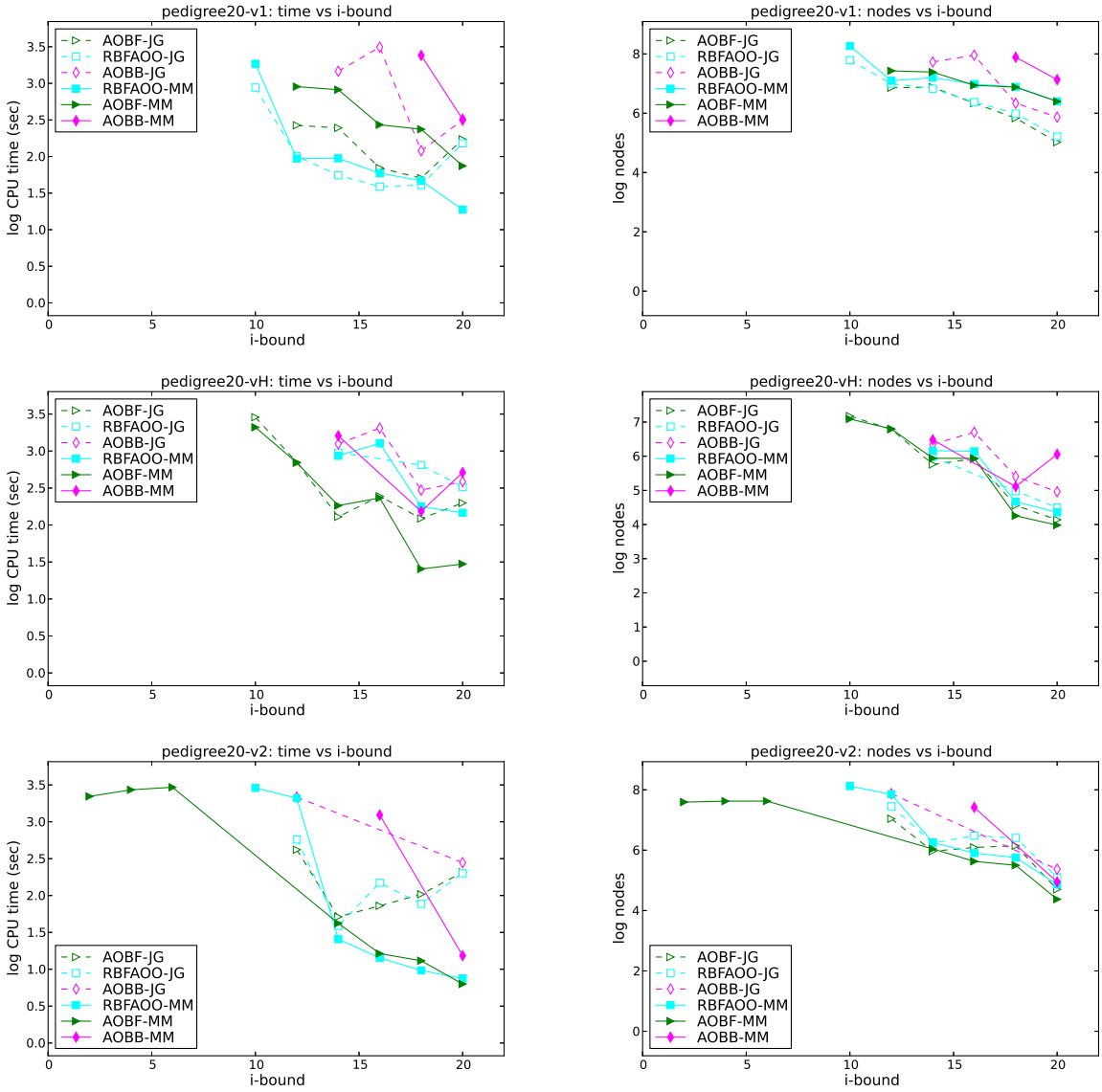

Figure 4: pedigree20 instance (time and nodes)

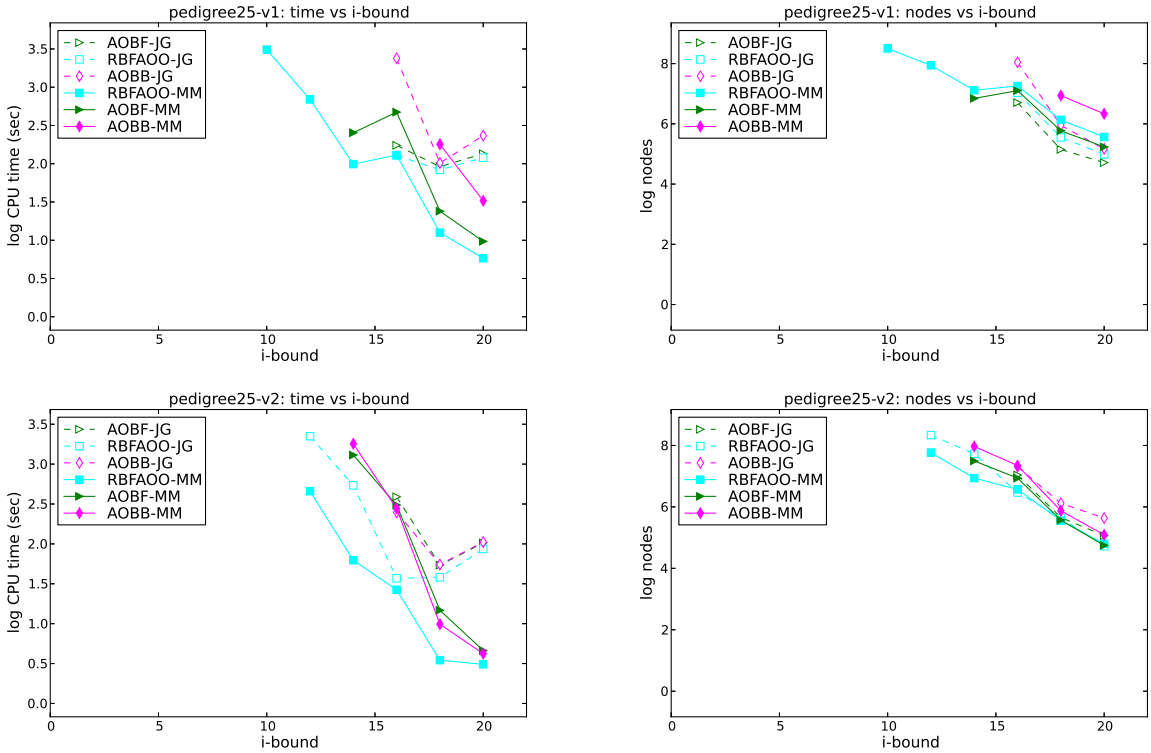

Figure 5: pedigree25 instance (time and nodes)

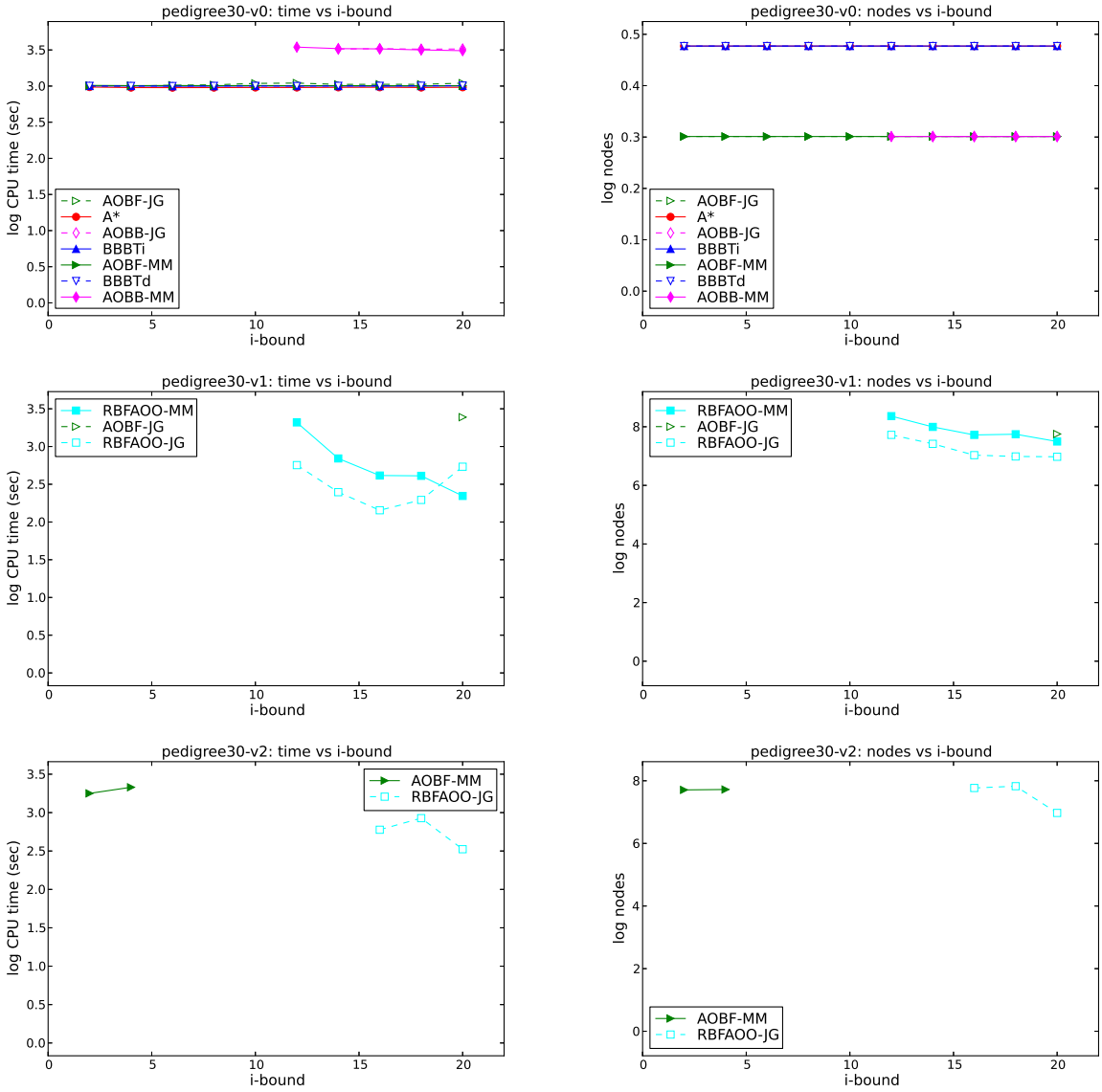

Figure 6: pedigree30 instance (time and nodes)

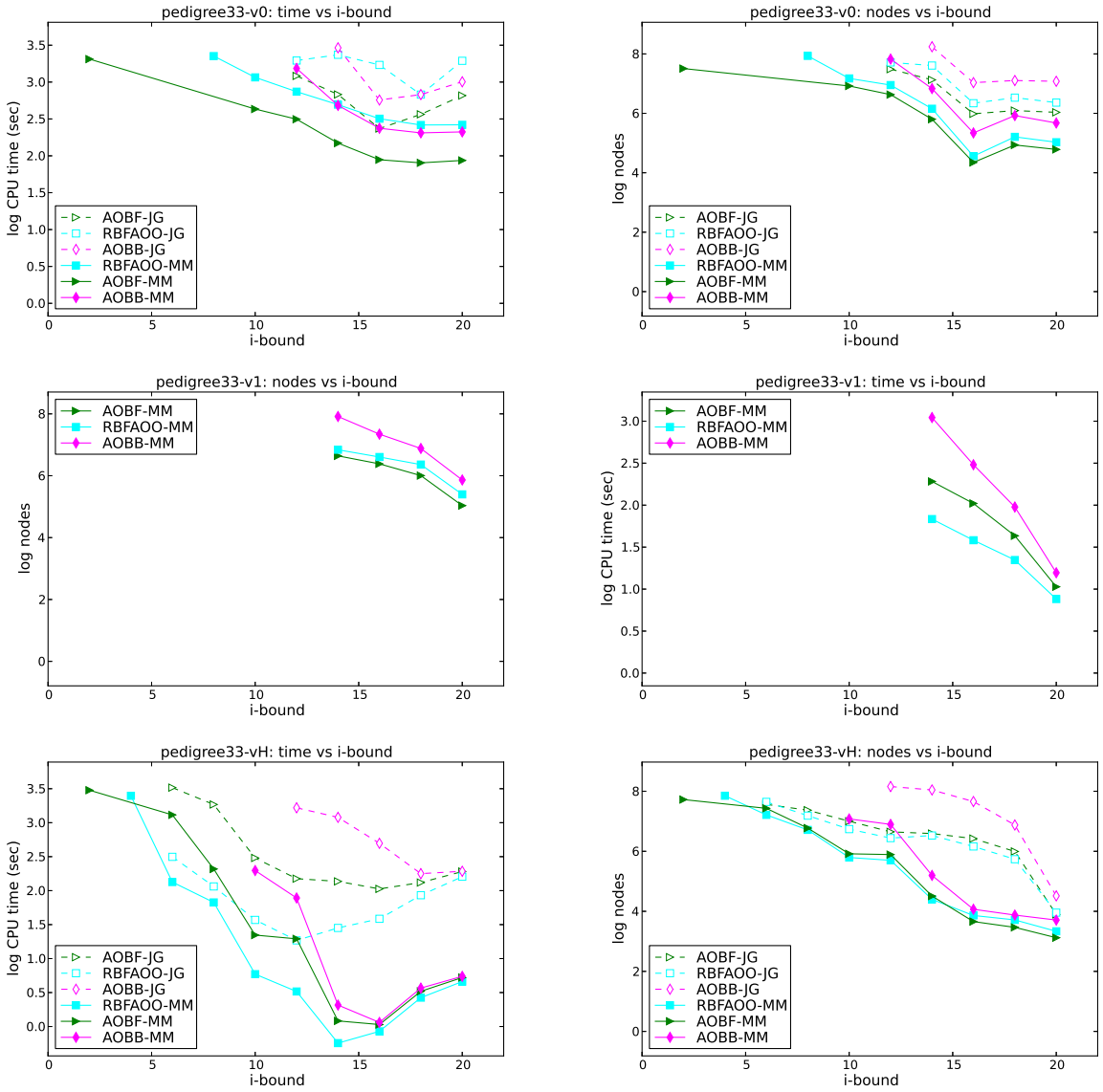

Figure 7: pedigree33 instance (time and nodes)

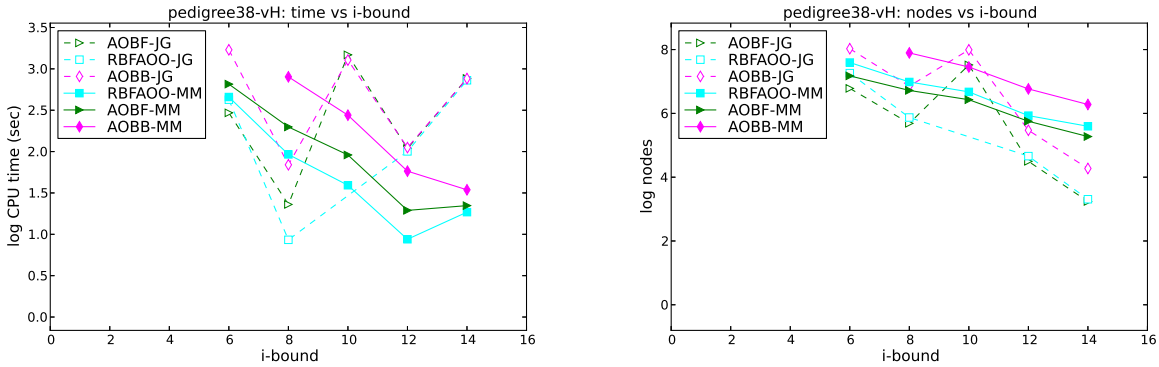

Figure 8: pedigree38 instance (time)

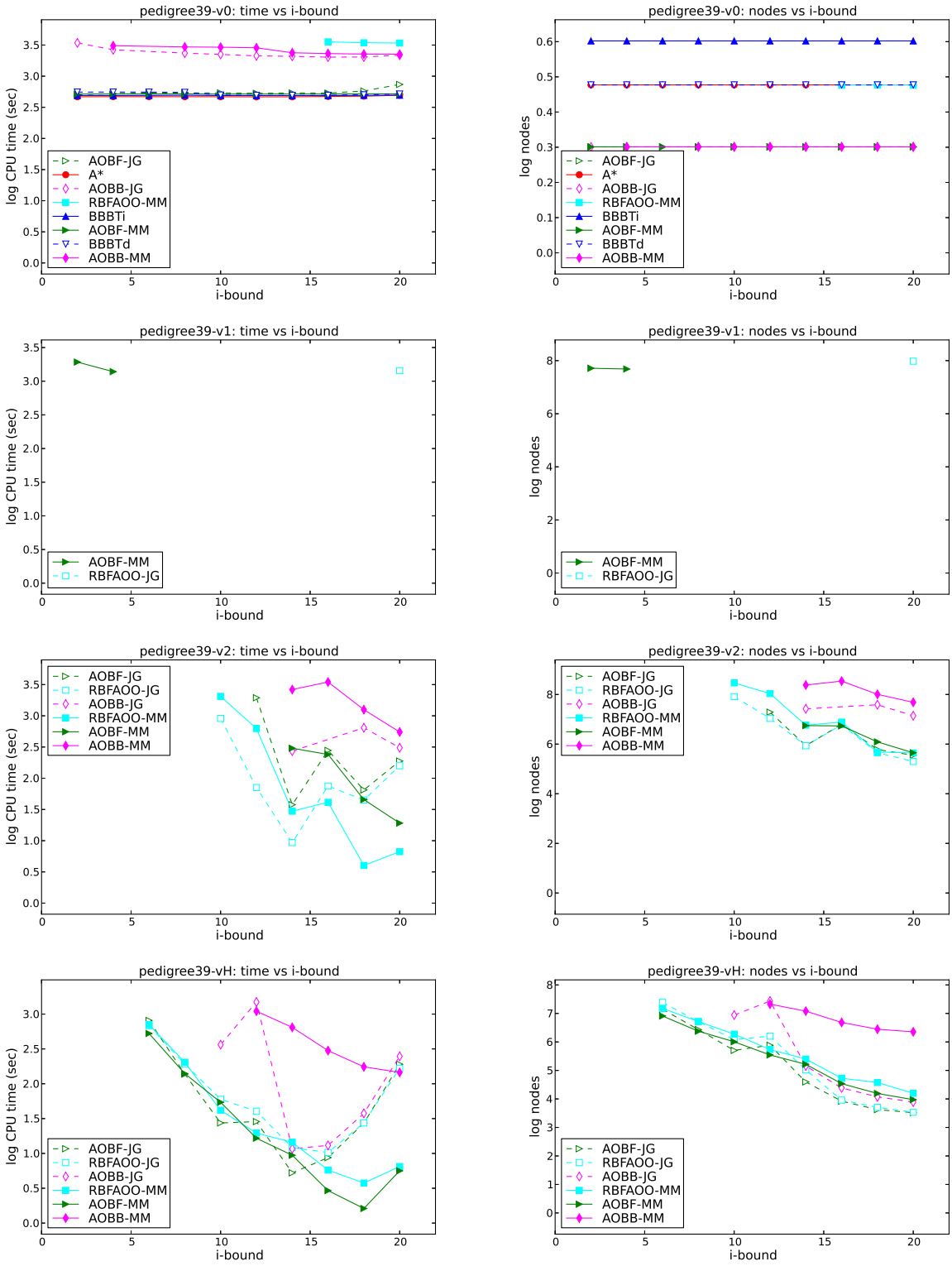

Figure 9: pedigree39 instance (time and nodes)

2.2 Results for `grid` networks

Figure 10 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the grid benchmark (includes all instances).

Figure 11 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the `easy` as well as the `hard` instances of the grid benchmark. The rest of the figures in this section show detailed plots for each of the instances from this benchmark.

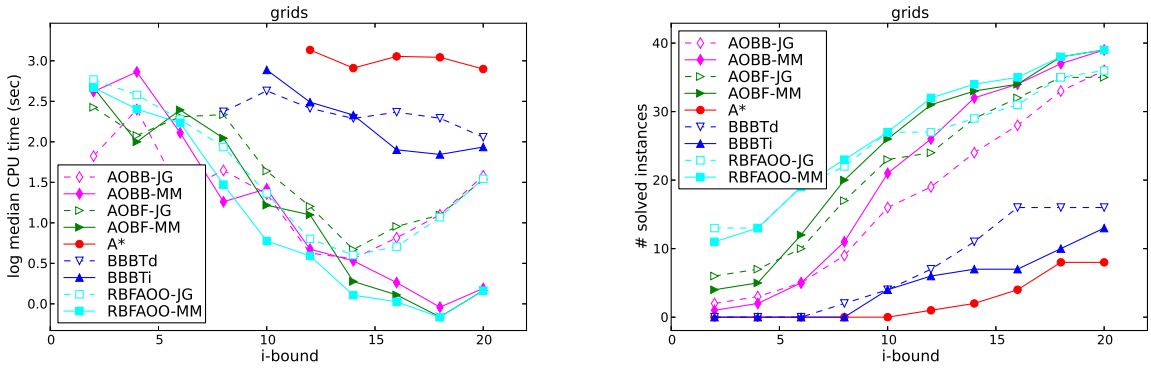

Figure 10: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `grids` benchmark.

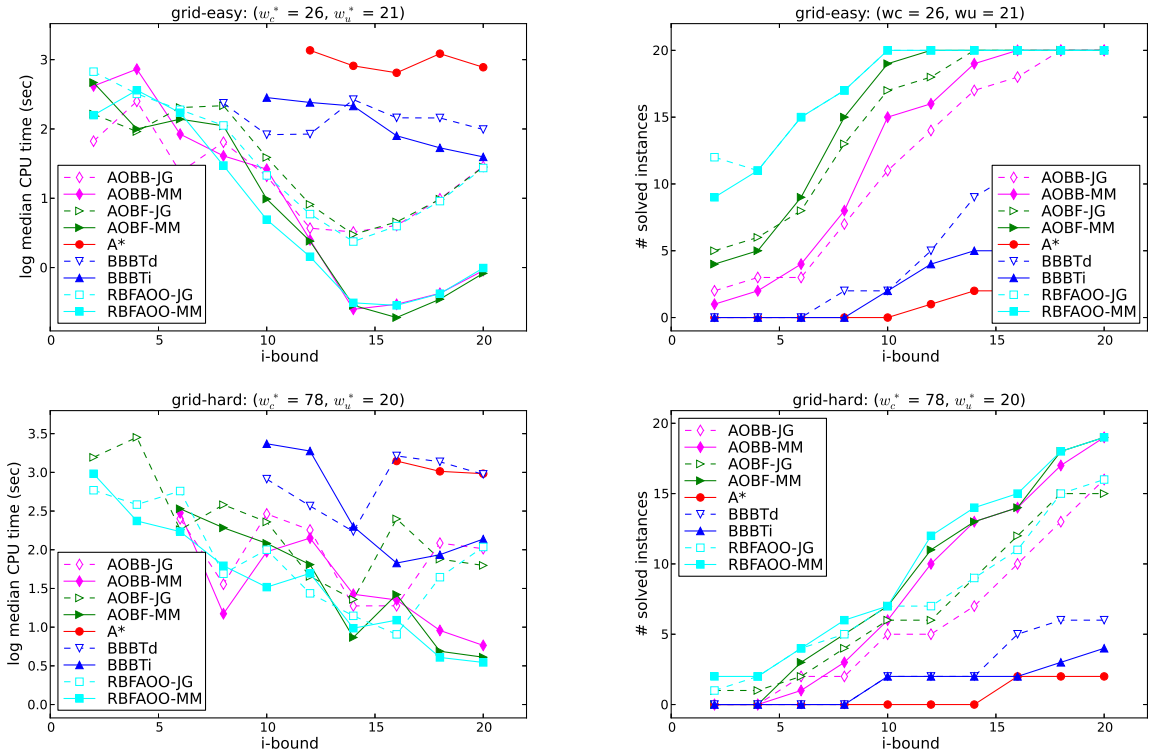

Figure 11: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `grids-easy` and `grids-hard` benchmark.

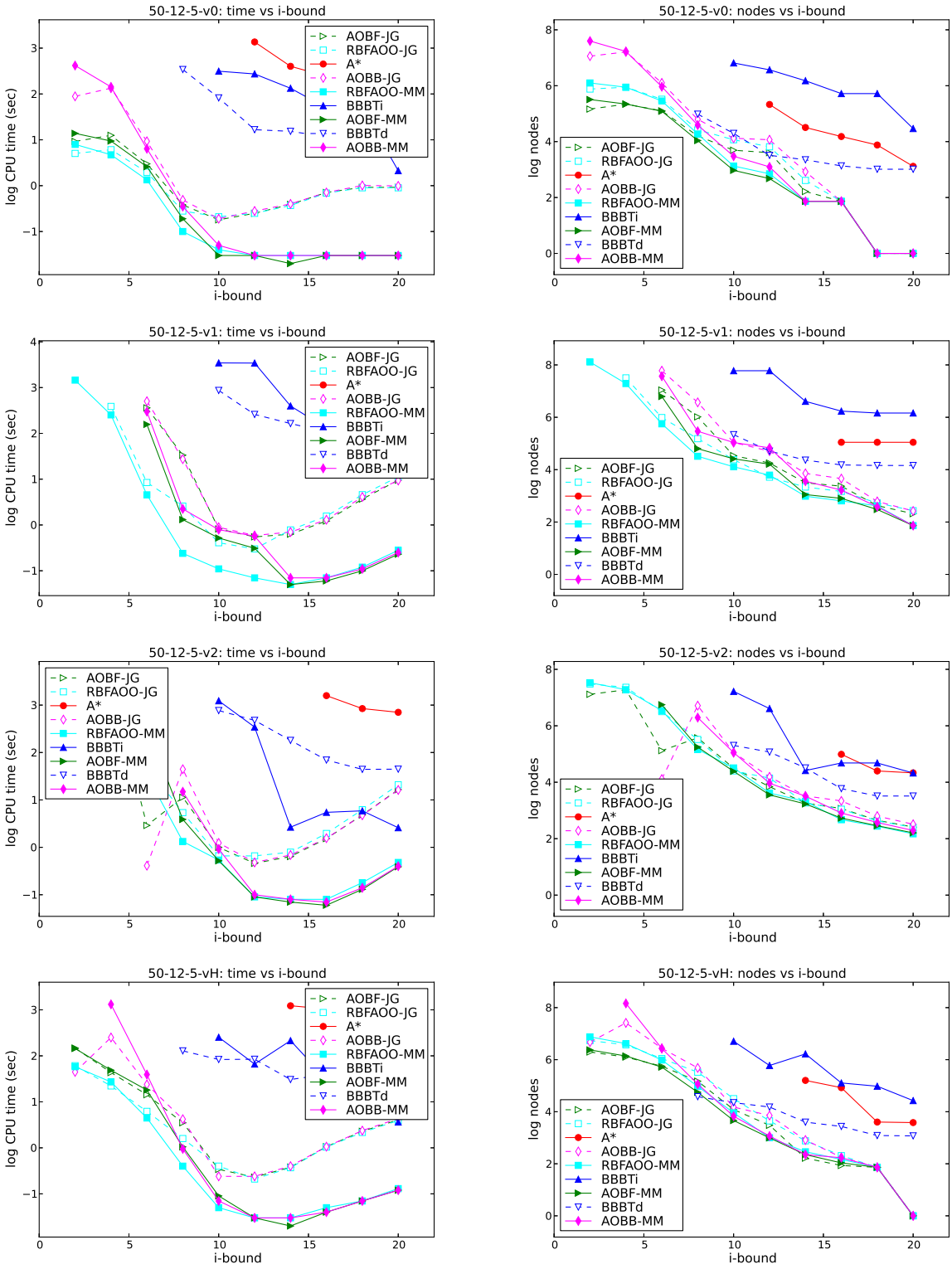

Figure 12: 50-12-5 instance (time and nodes)

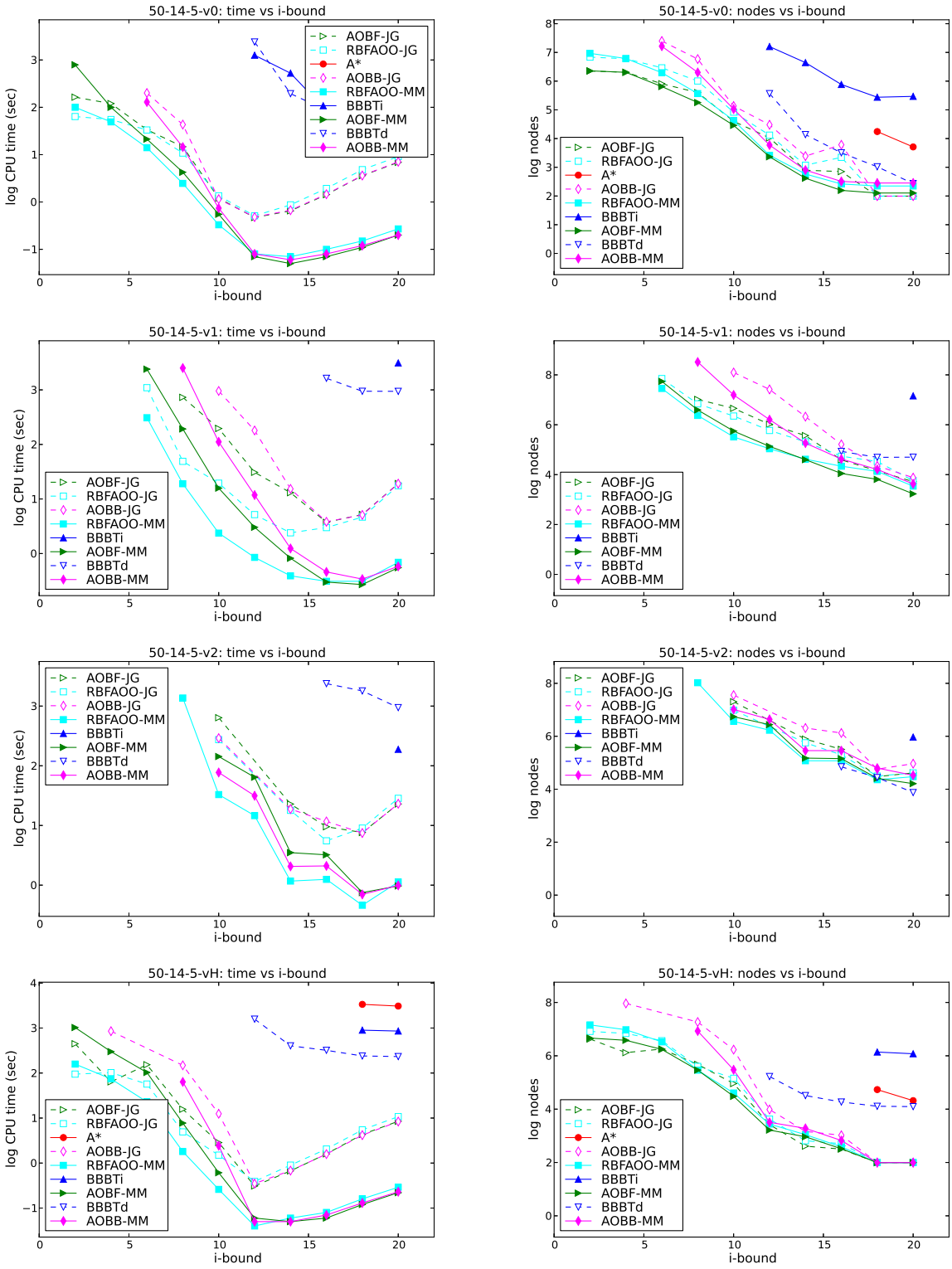

Figure 13: 50-14-5 instance (time and nodes)

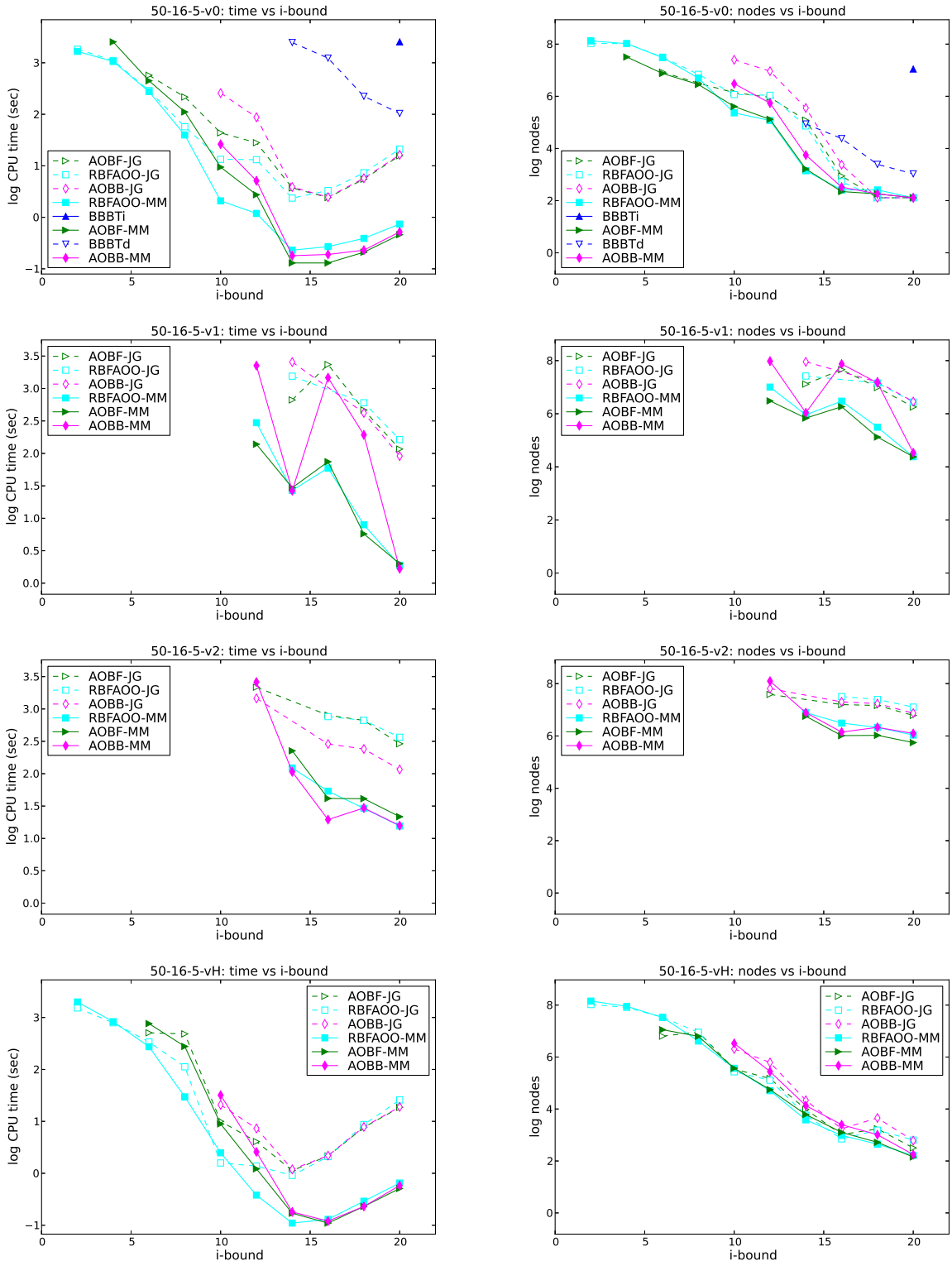

Figure 14: 50-16-5 instance (time and nodes)

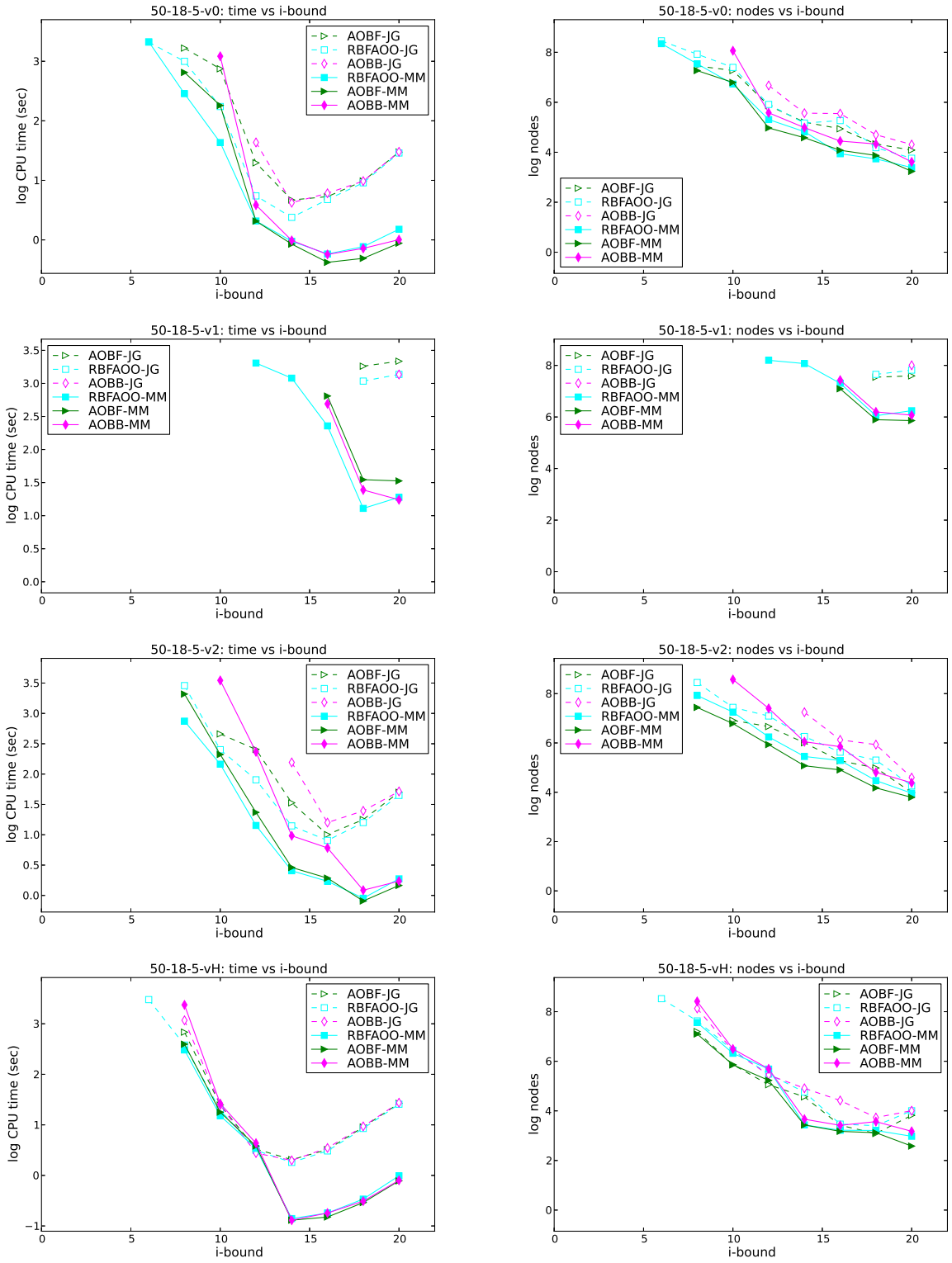

Figure 15: 50-18-5 instance (time and nodes)

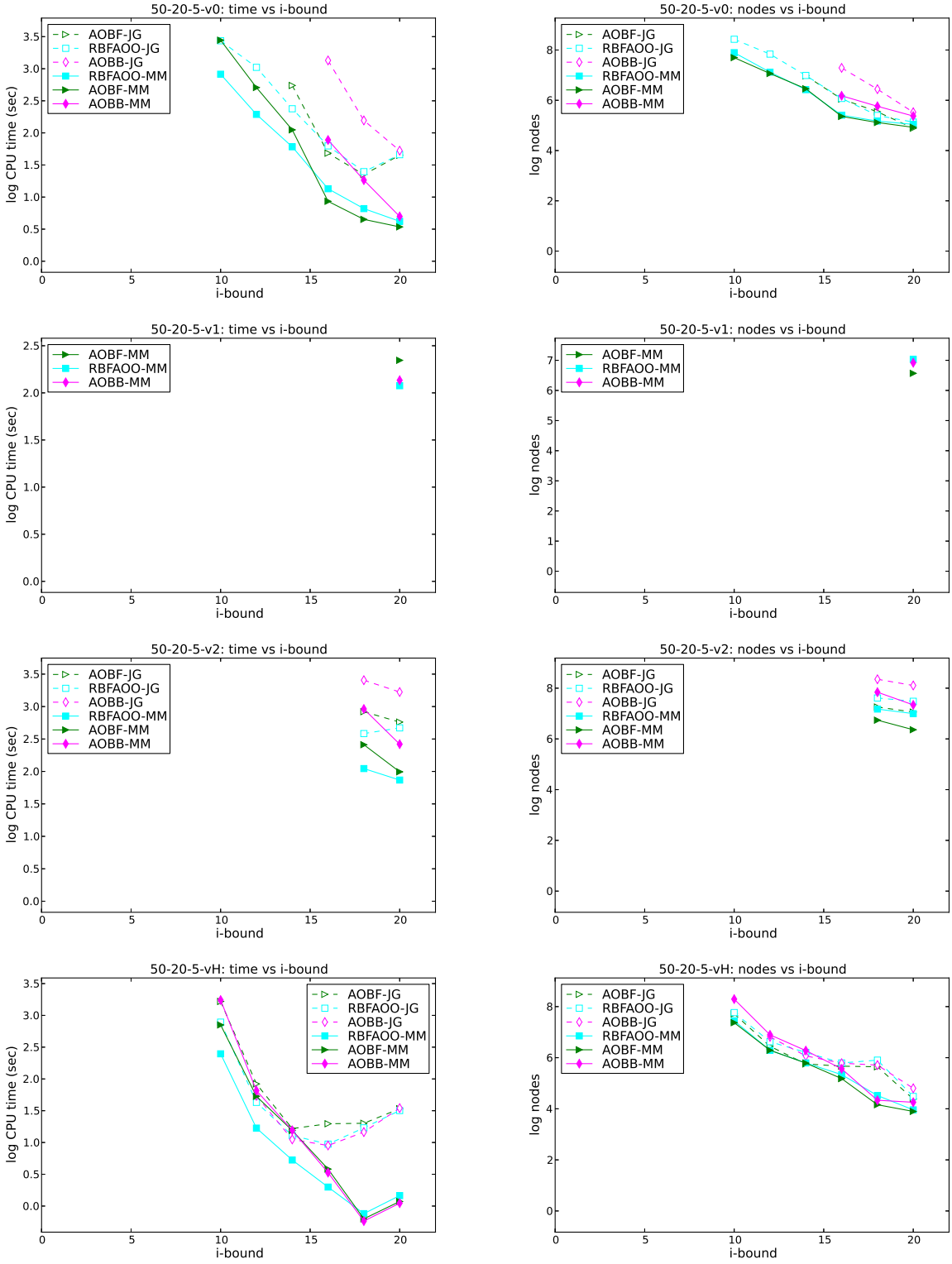

Figure 16: 50-20-5 instance (time and nodes)

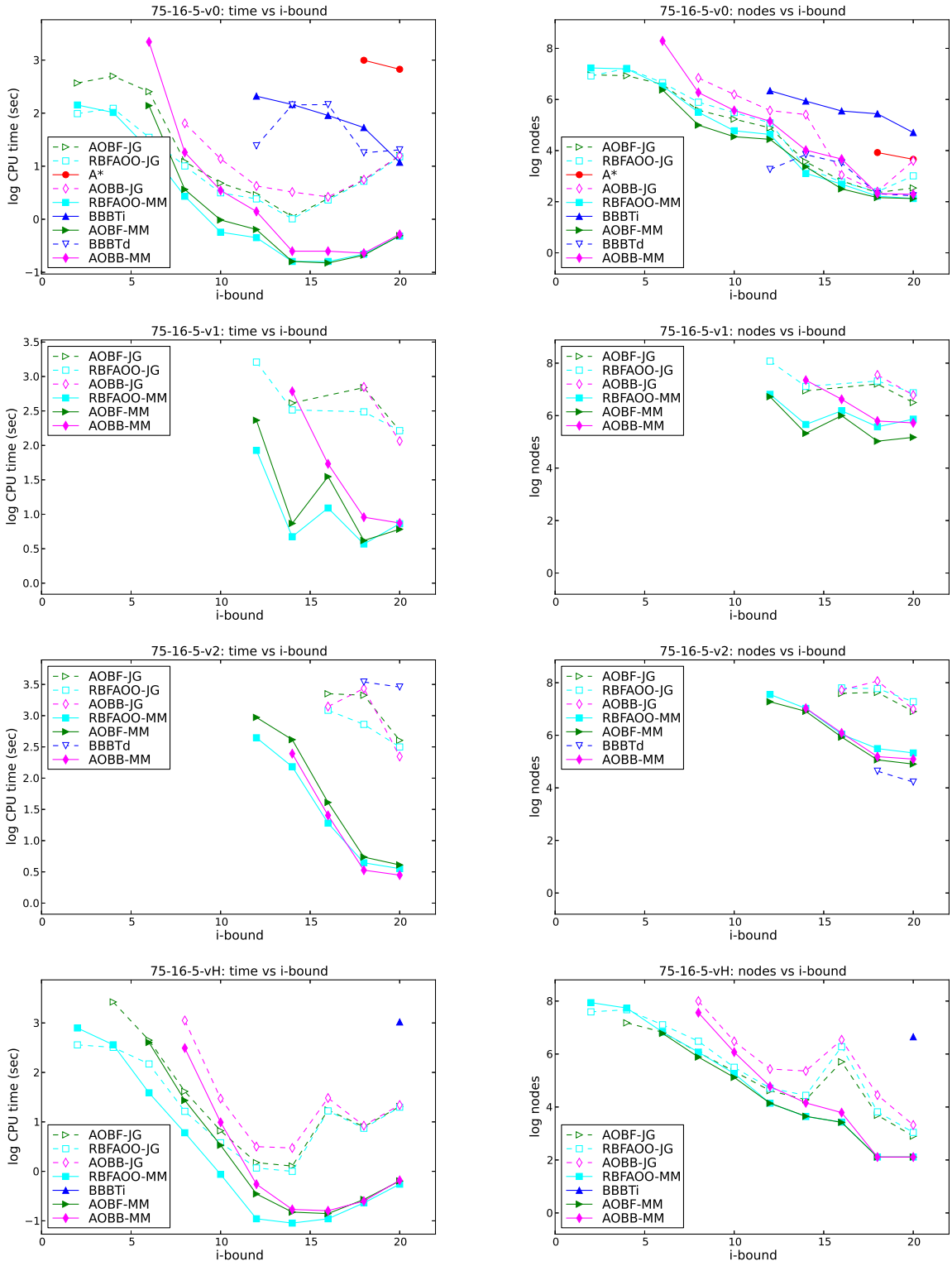

Figure 17: 75-16-5 instance (time and nodes)

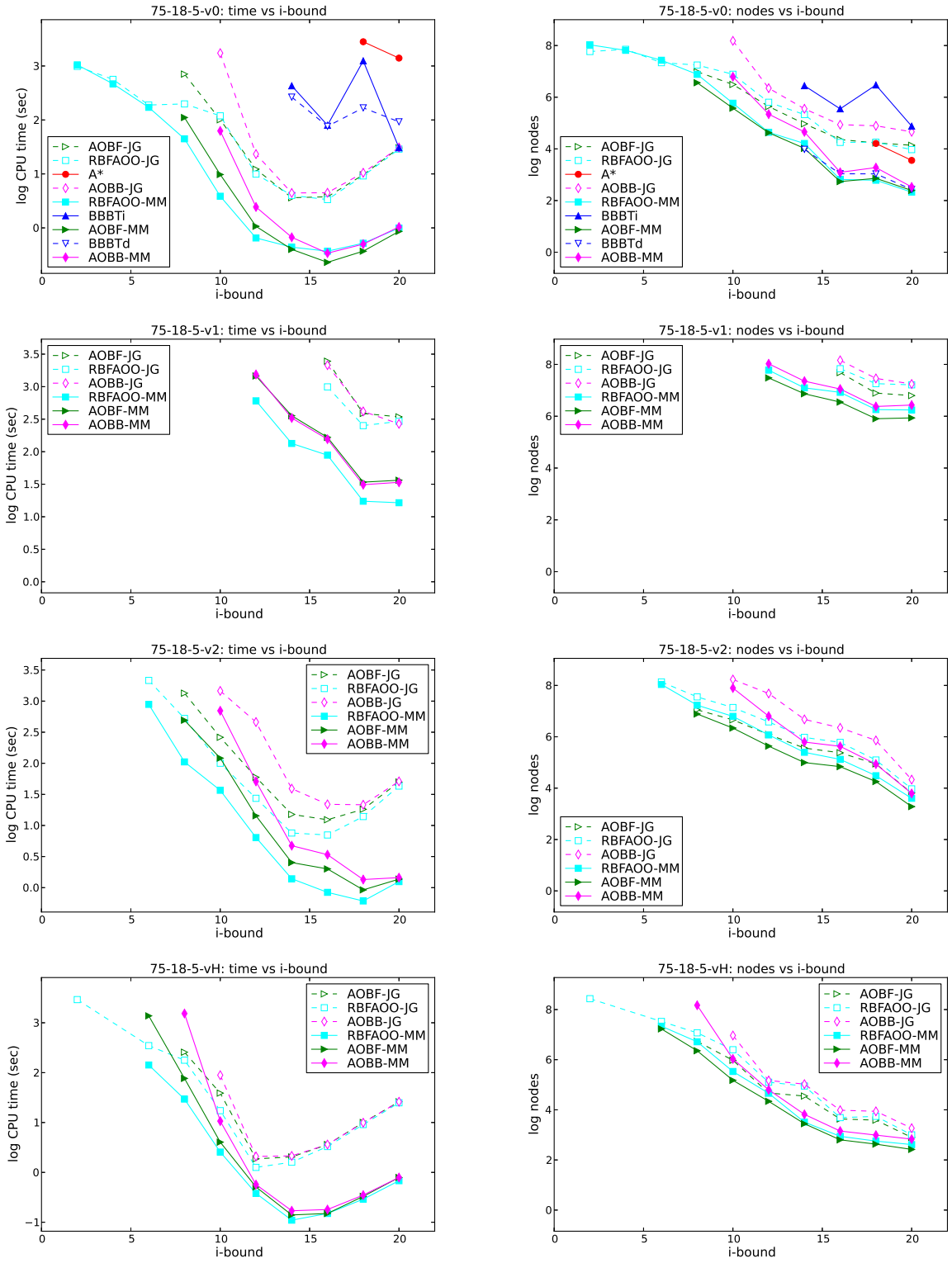

Figure 18: 75-18-5 instance (time and nodes)

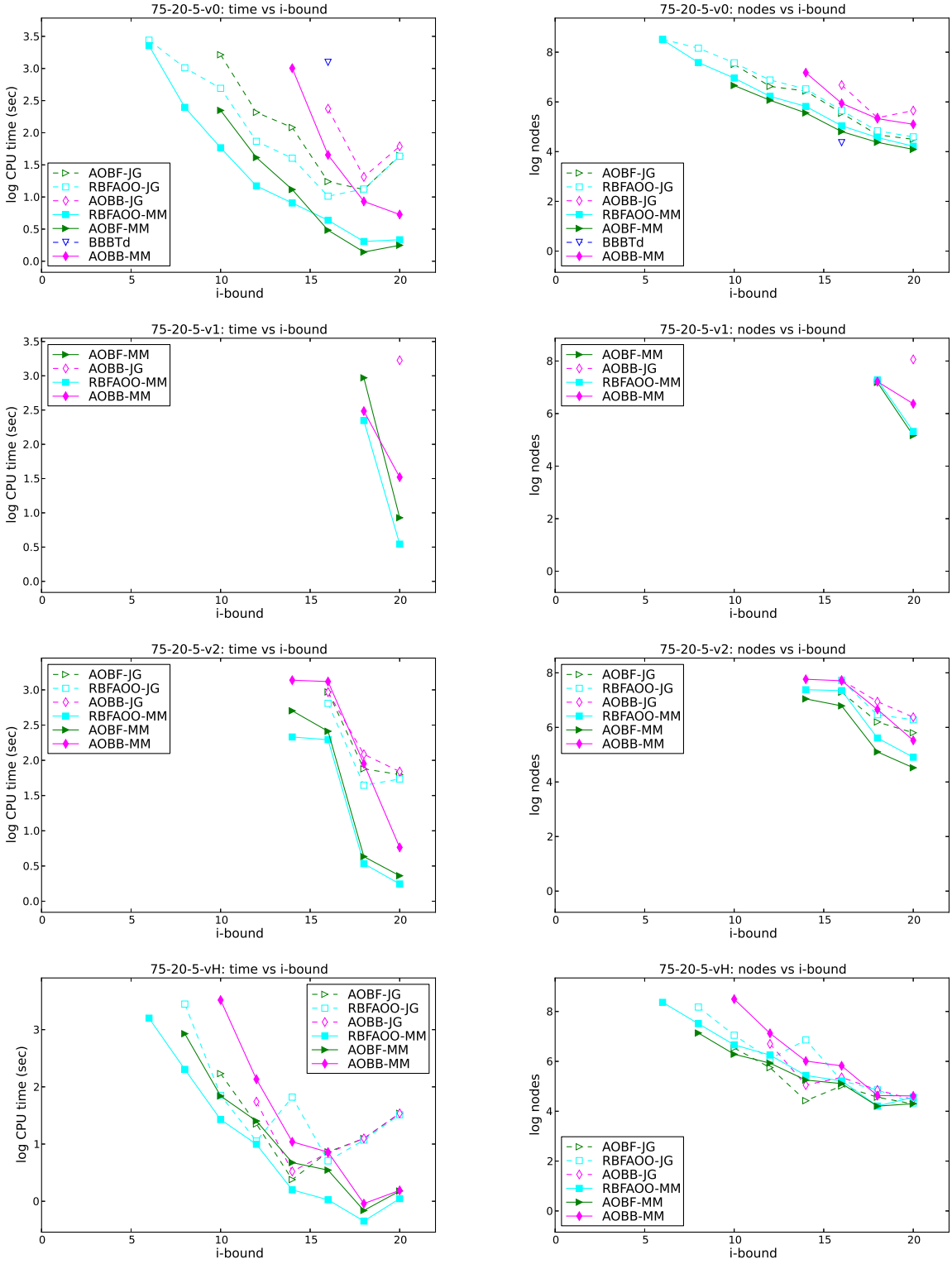

Figure 19: 75-20-5 instance (time and nodes)

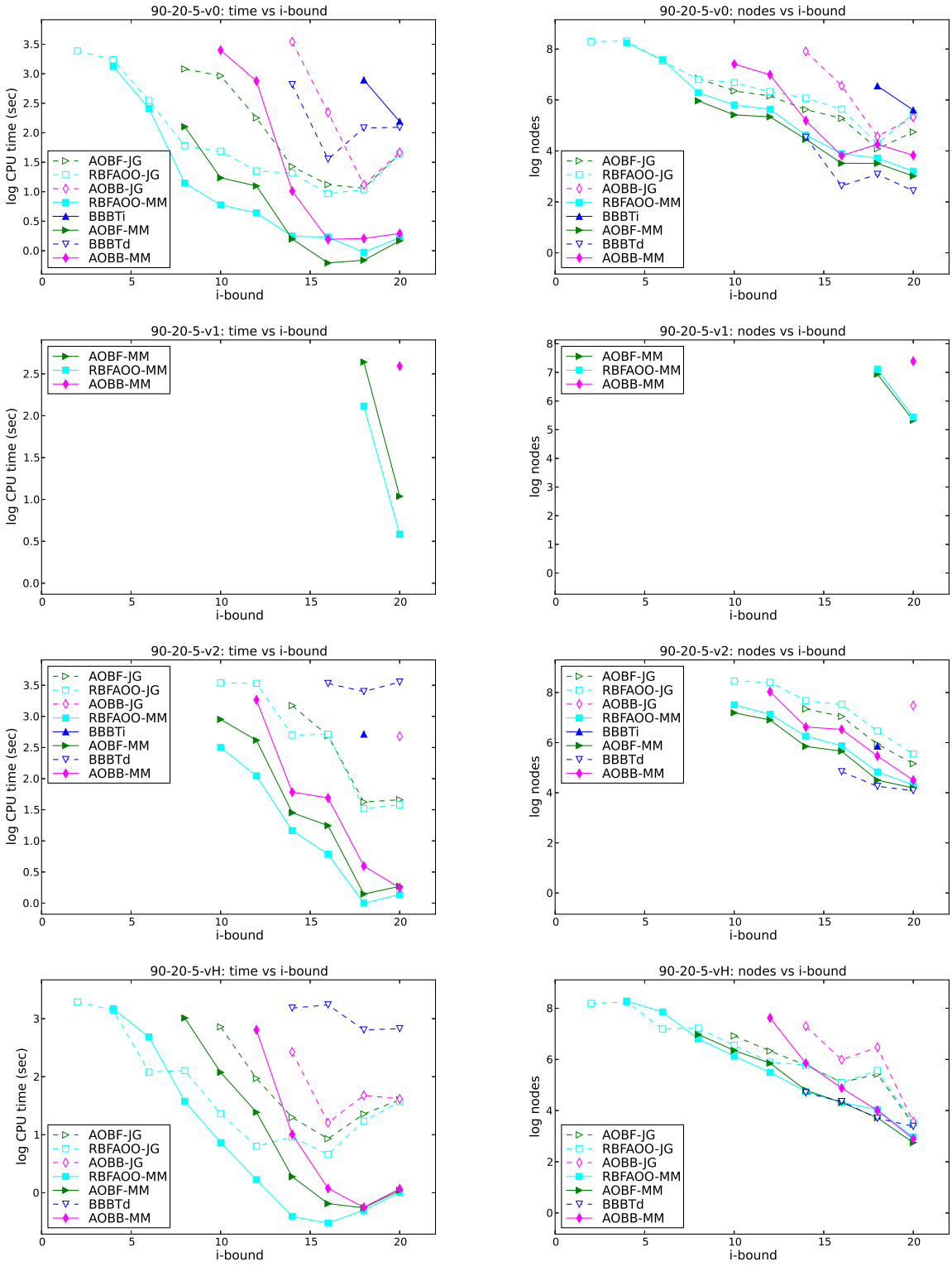

Figure 20: 90-20-5 instance (time and nodes)

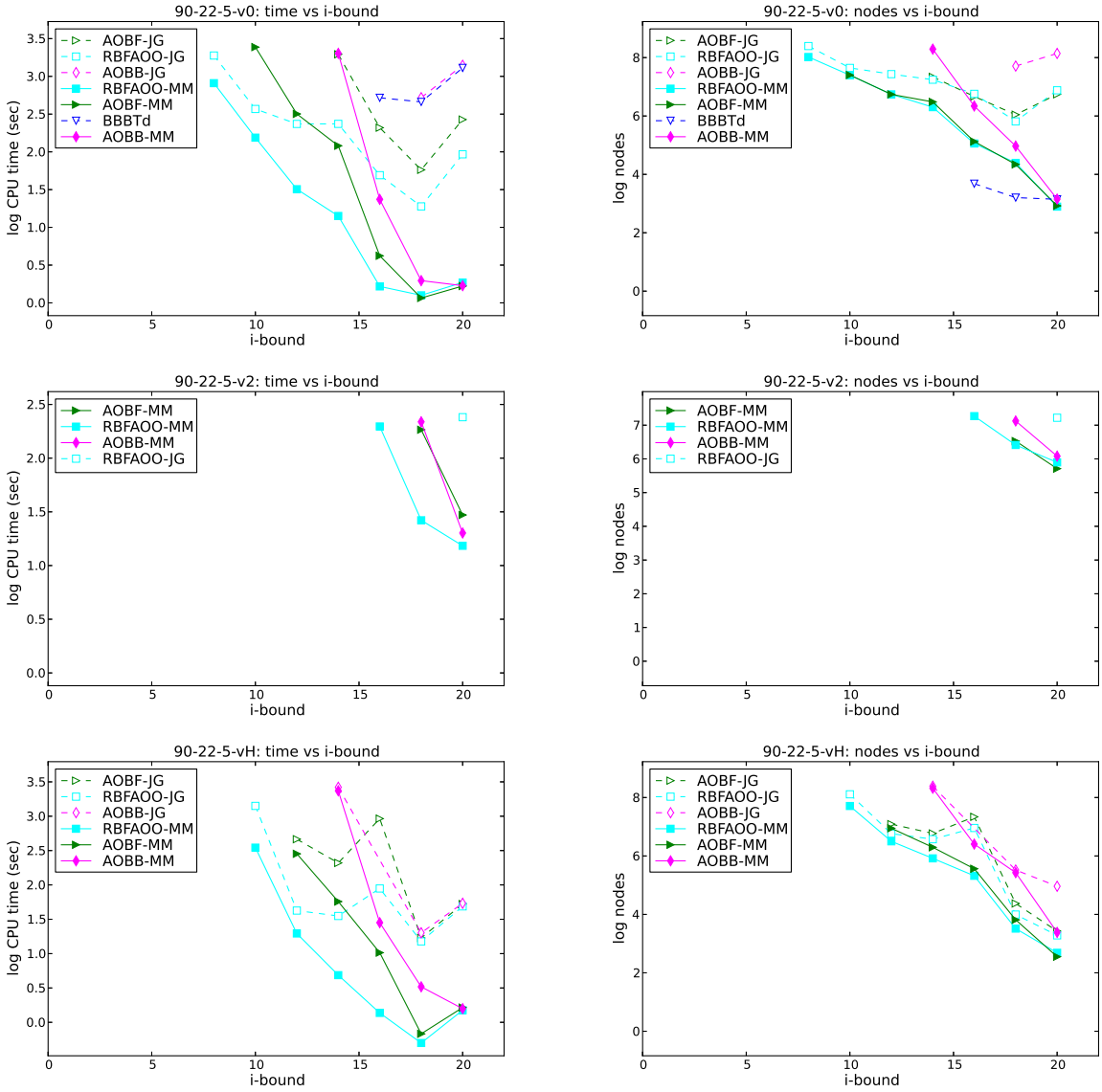

Figure 21: 90-22-5 instance (time and nodes)

2.3 Results for `promedas` networks

Figure 22 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the grid benchmark (includes all instances).

Figure 23 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the `easy` as well as the `hard` instances of the grid benchmark. The rest of the figures in this section show detailed plots for each of the instances from this benchmark.

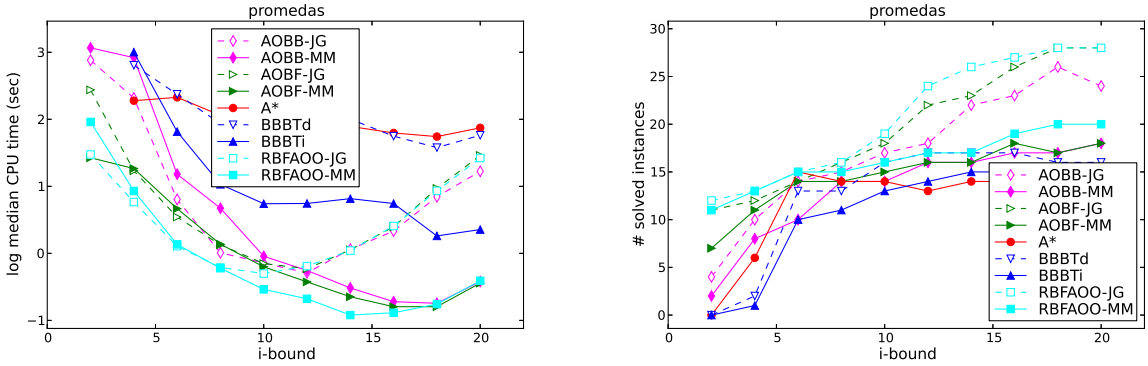

Figure 22: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `promedas` benchmark.

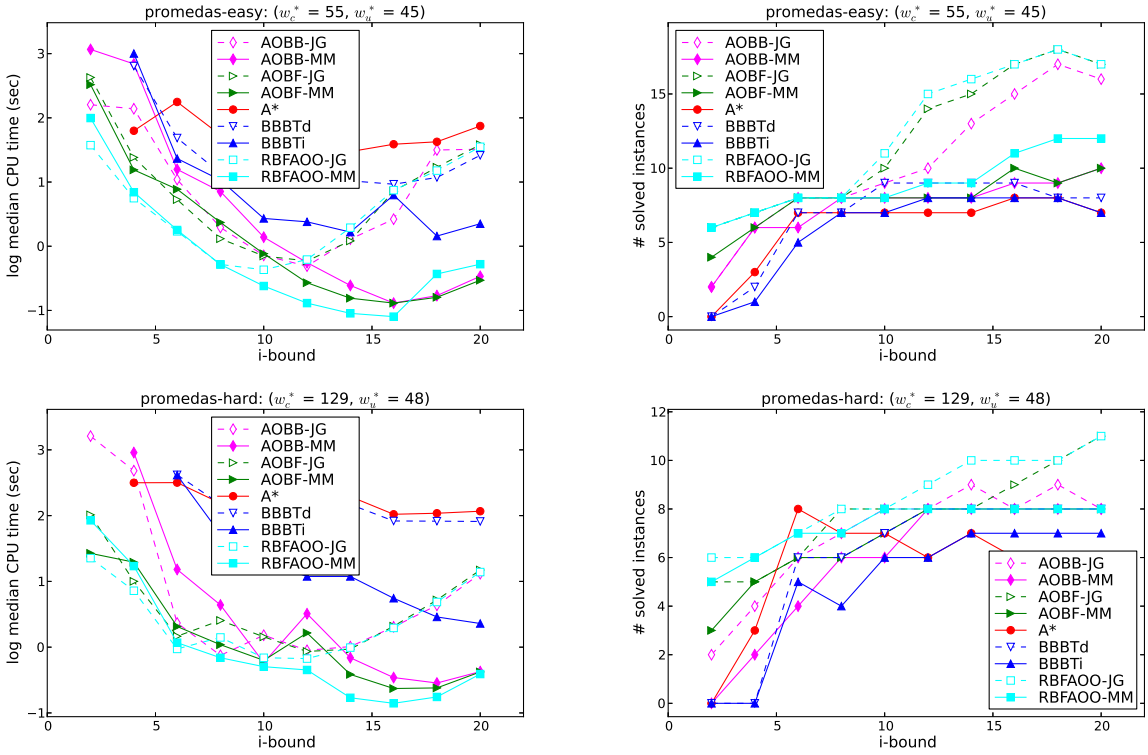

Figure 23: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `promedas-easy` and `promedas-hard` benchmark.

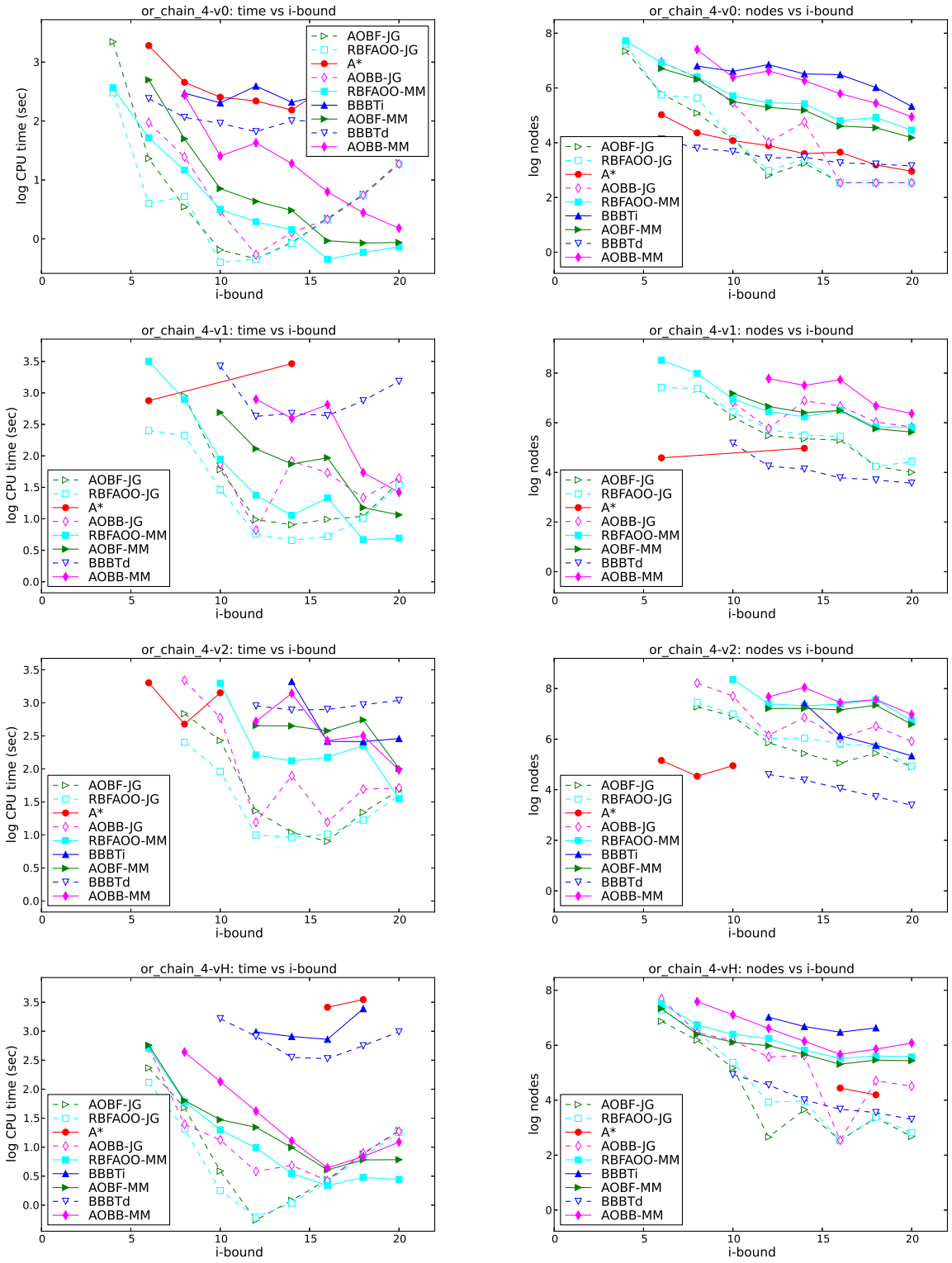

Figure 24: or-chain-4 instance (time and nodes)

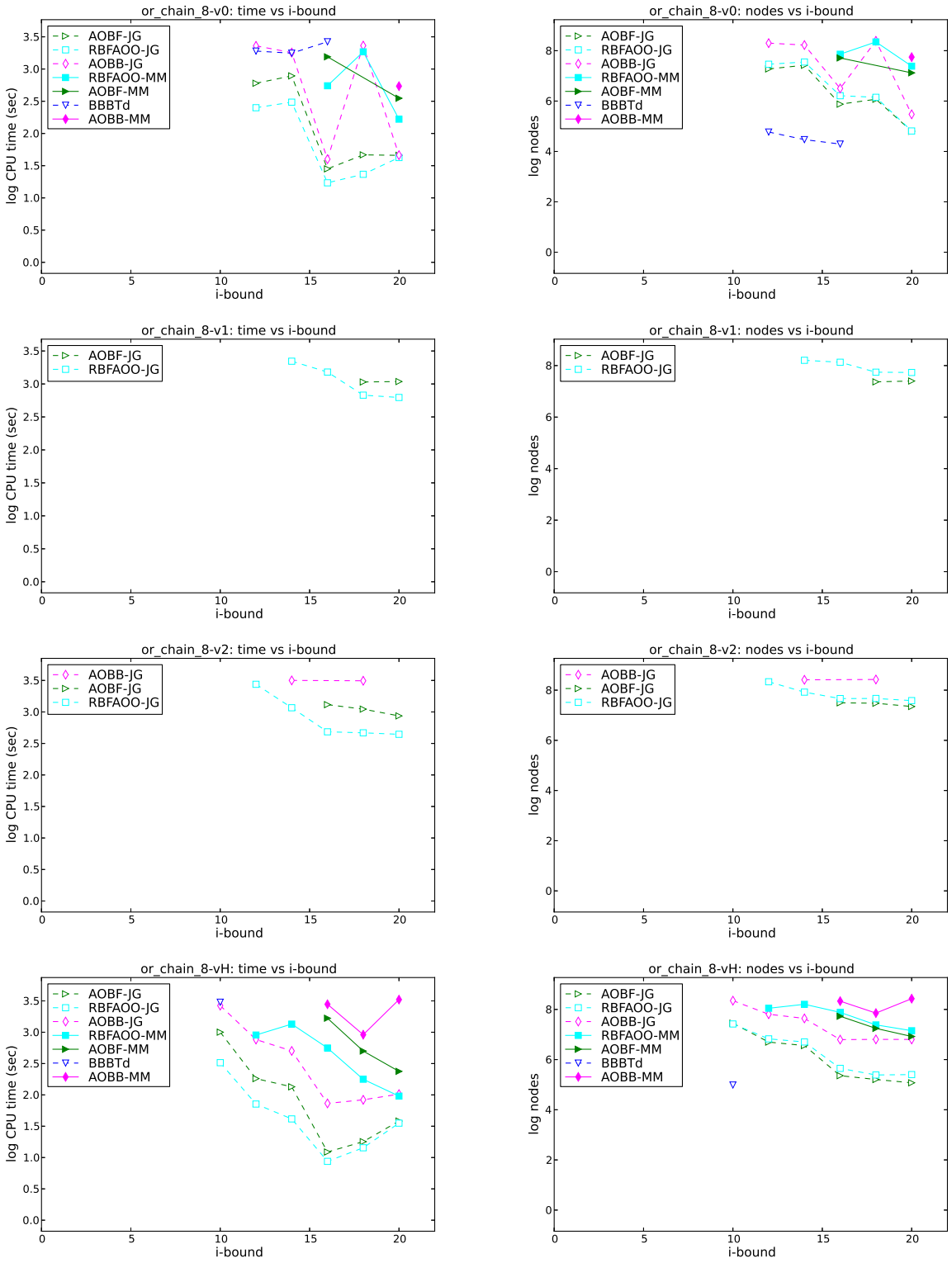

Figure 25: or-chain-8 instance (time and nodes)

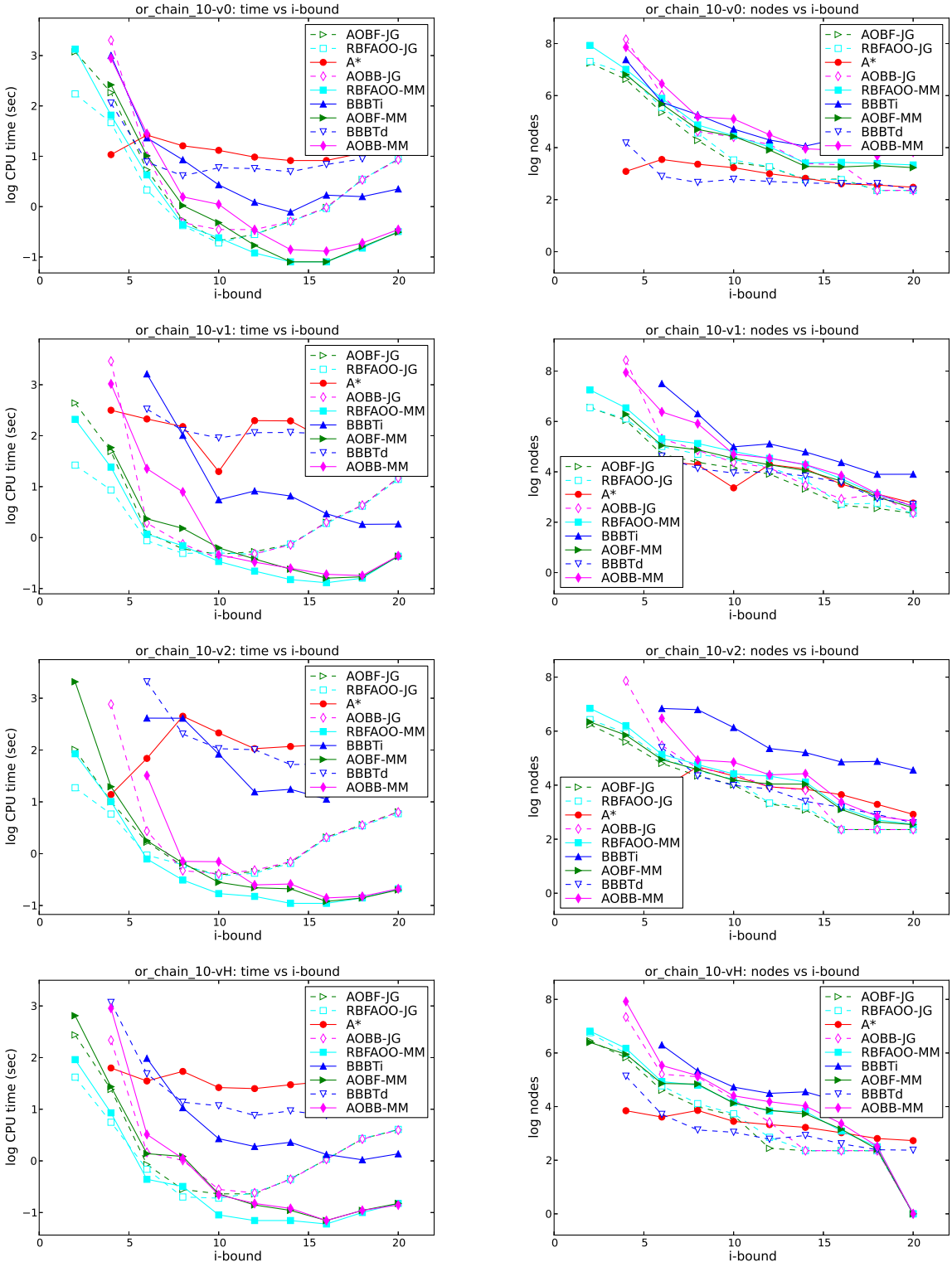

Figure 26: or-chain-10 instance (time and nodes)

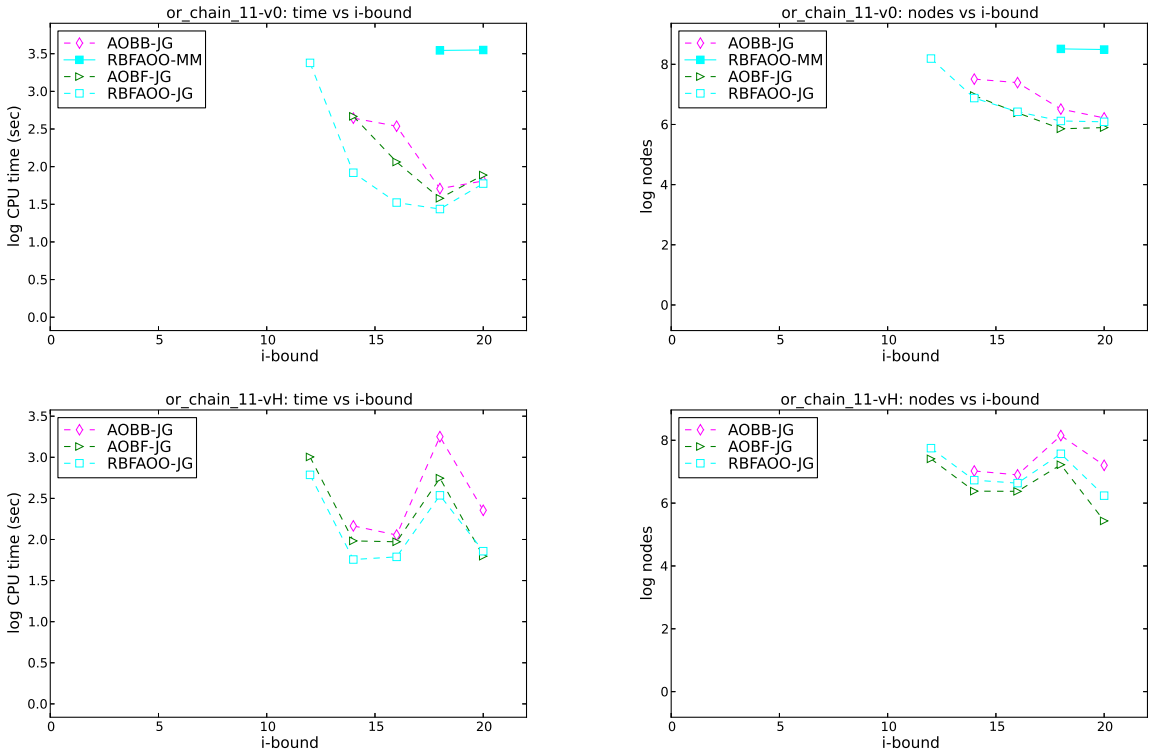

Figure 27: or-chain-11 instance (time and nodes)

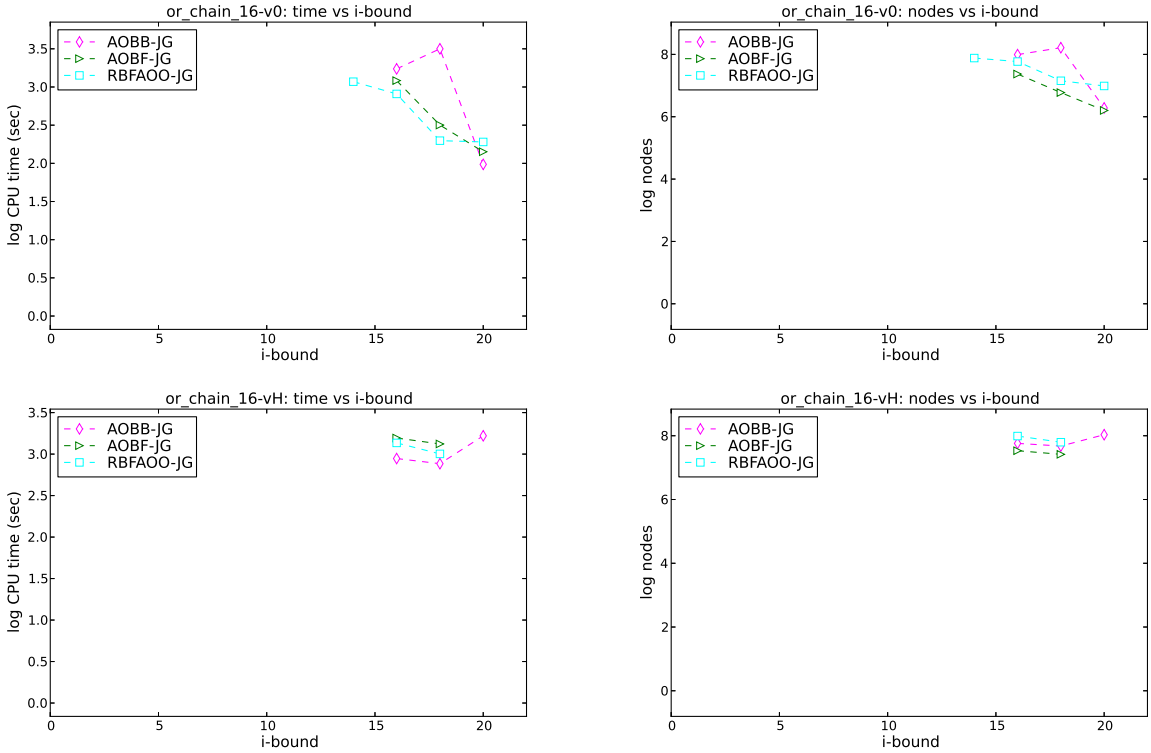

Figure 28: or-chain-16 instance (time and nodes)

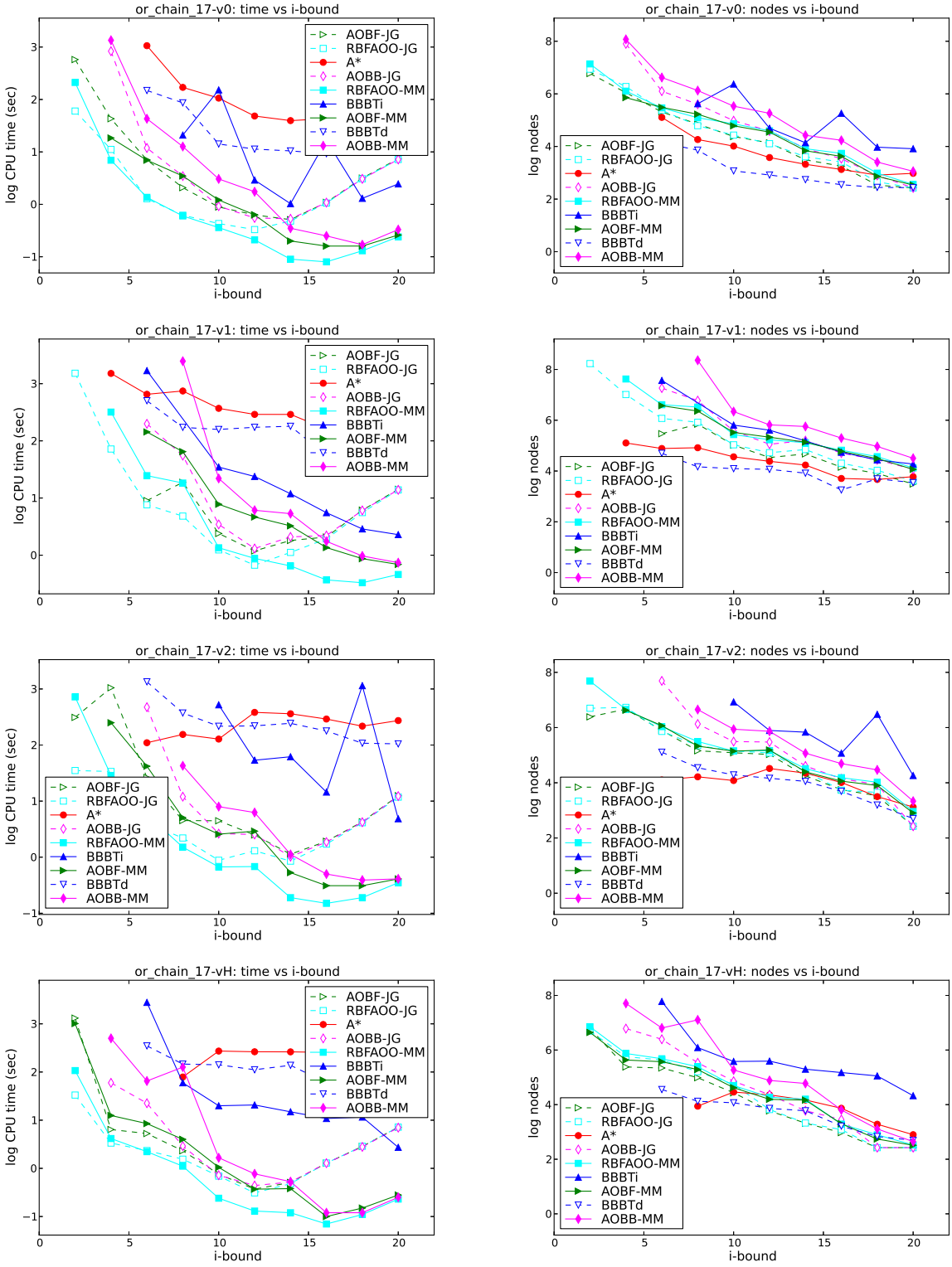

Figure 29: or-chain-17 instance (time and nodes)

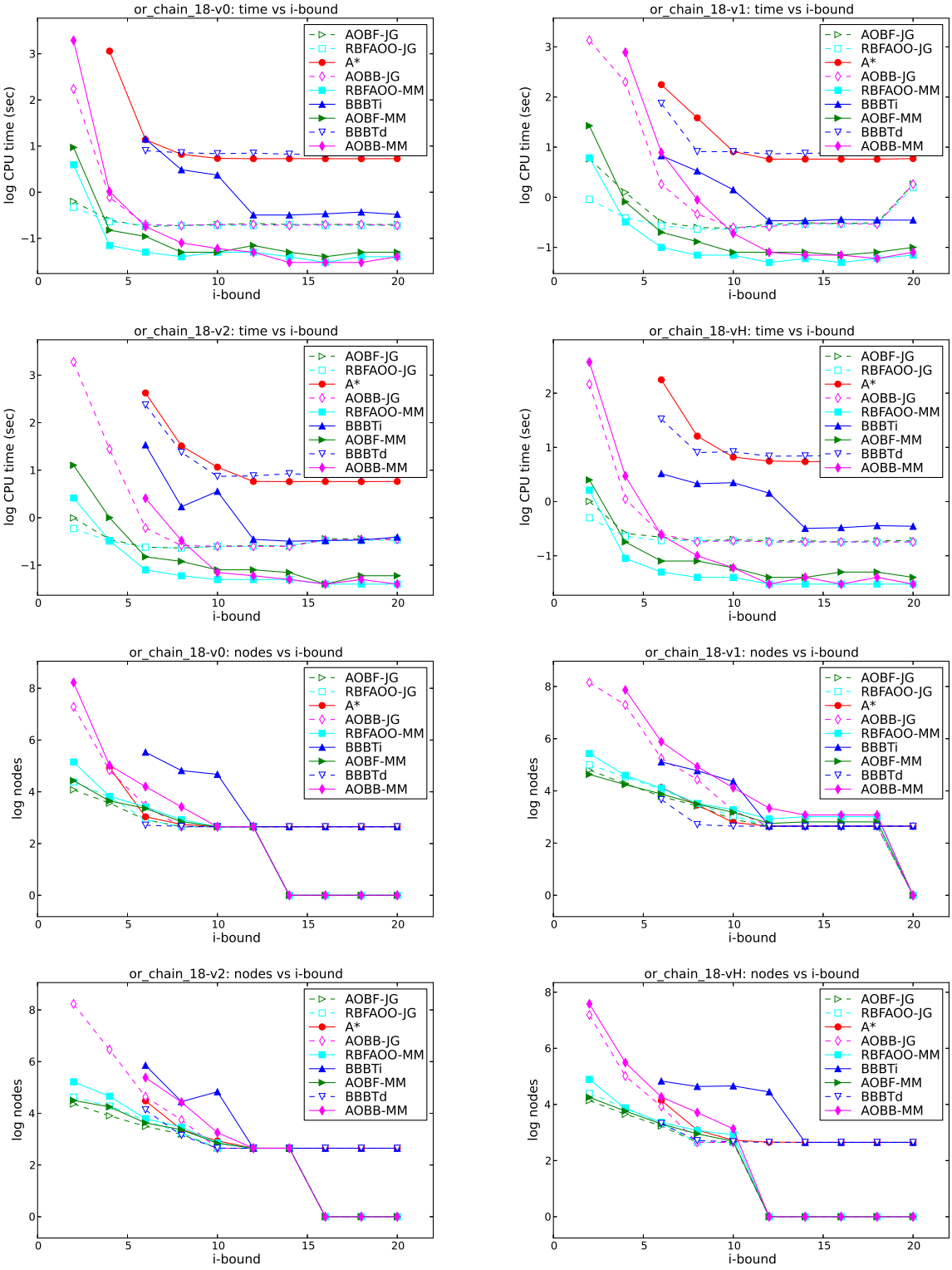

Figure 30: or-chain-18 instance (time and nodes)

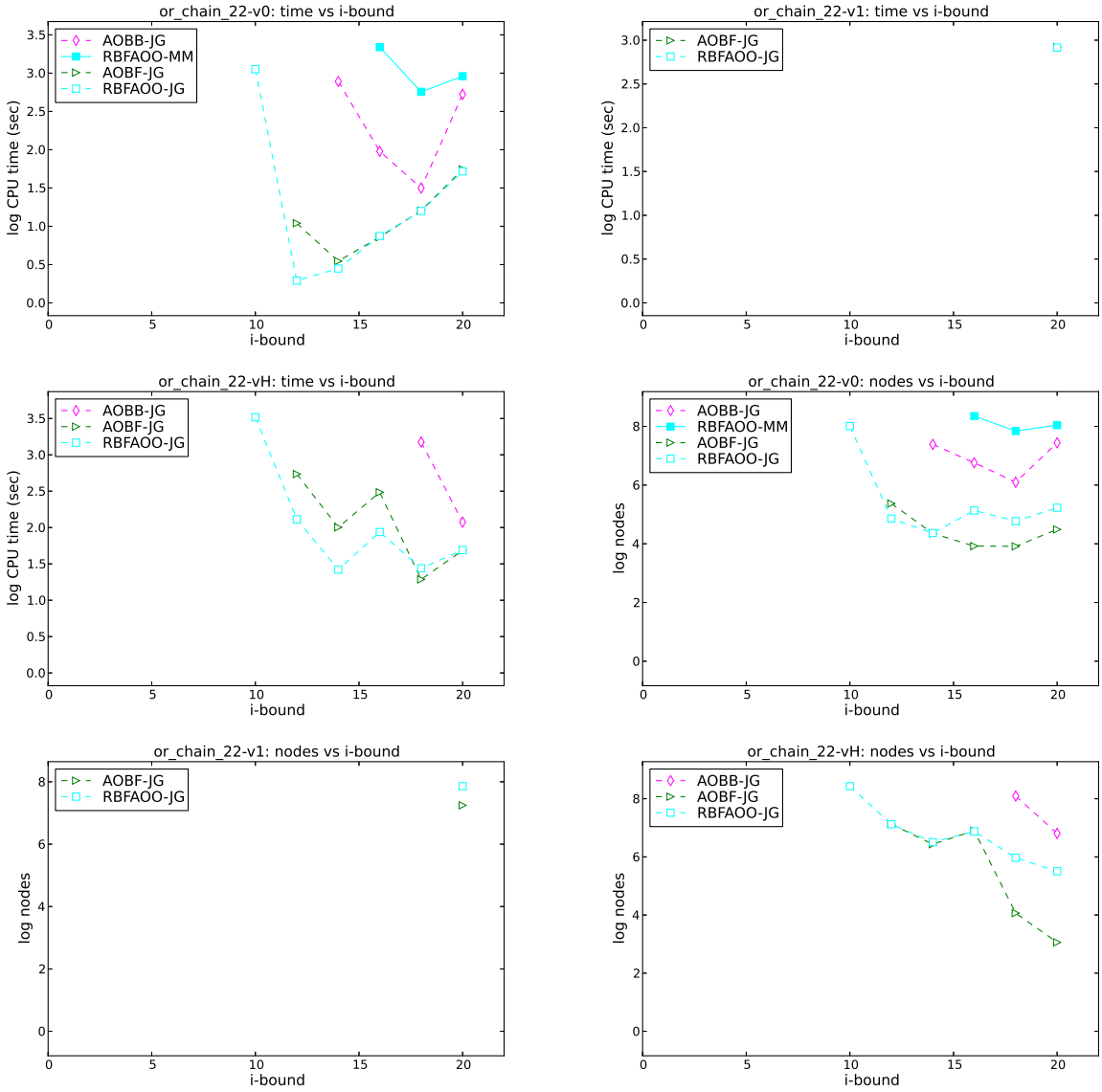

Figure 31: or-chain-22 instance (time and nodes)

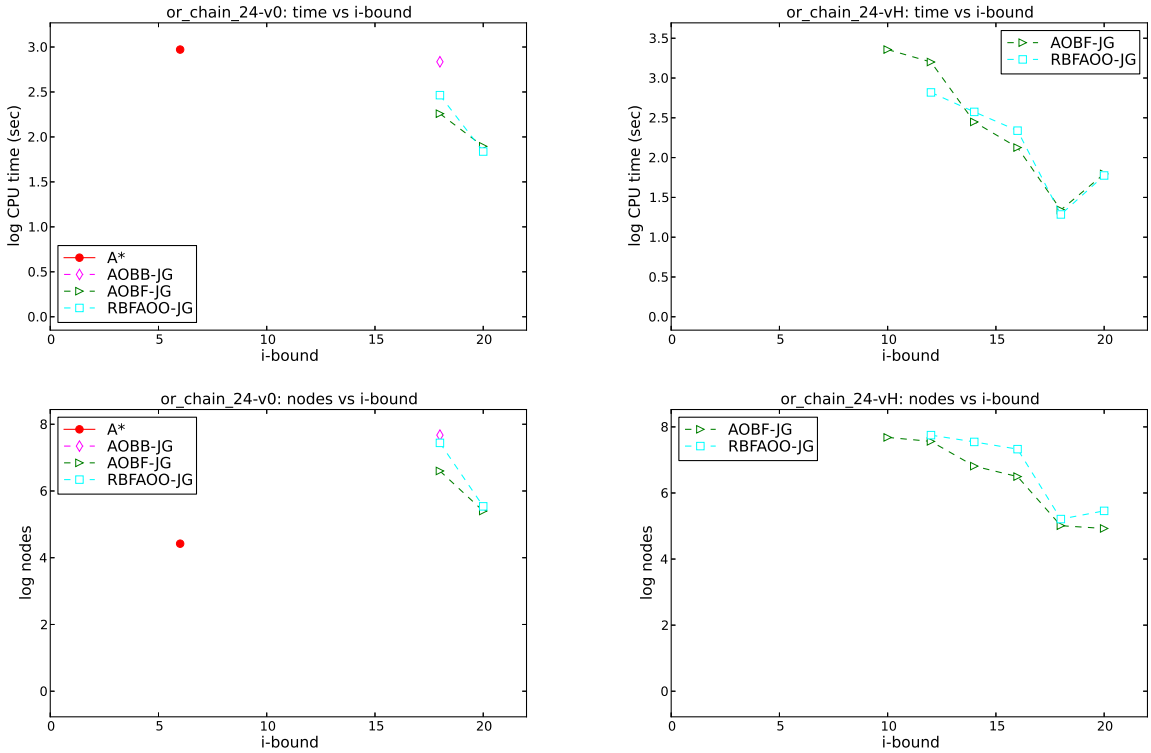

Figure 32: or-chain-24 instance (time and nodes)

2.4 Results for `segbin` networks

Figure 33 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the grid benchmark (includes all instances).

Figure 34 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the `easy` as well as the `hard` instances of the grid benchmark. The rest of the figures in this section show detailed plots for each of the instances from this benchmark.

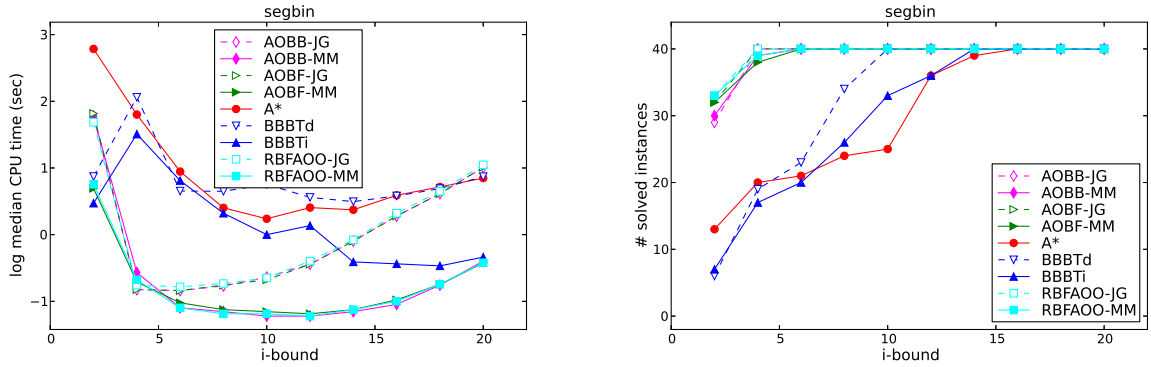

Figure 33: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `segbin` benchmark.

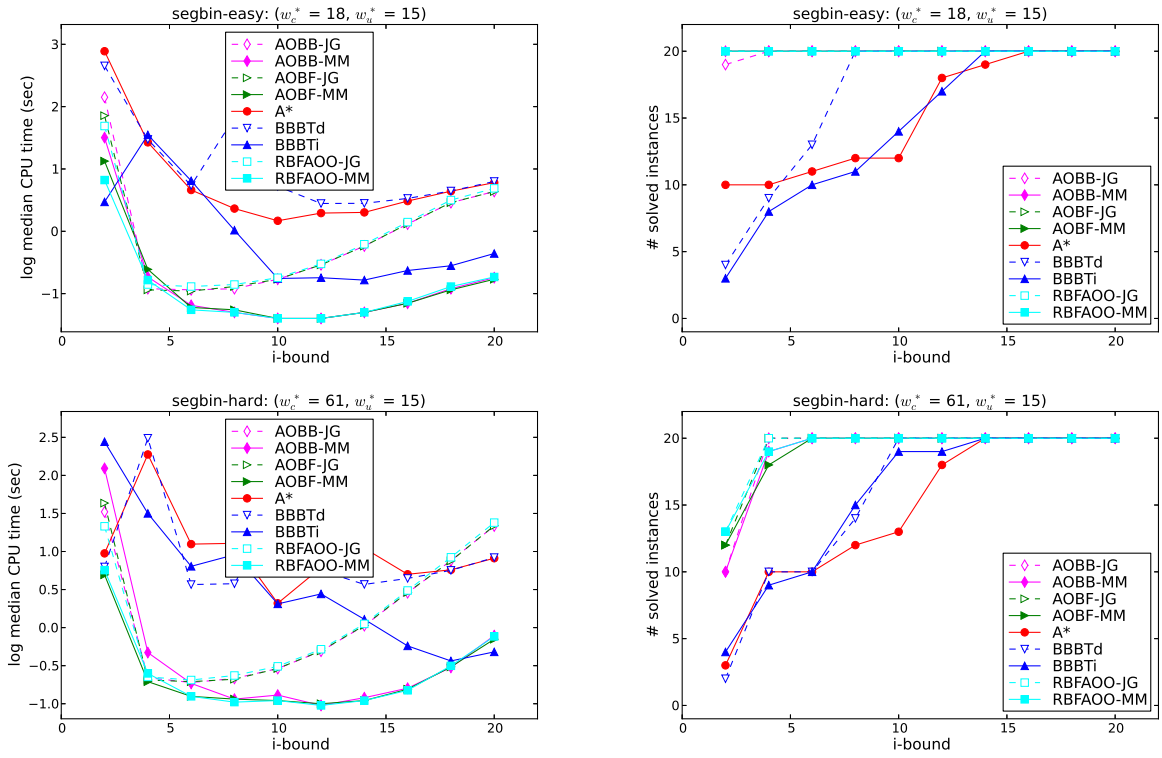

Figure 34: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `segbin-easy` benchmark.

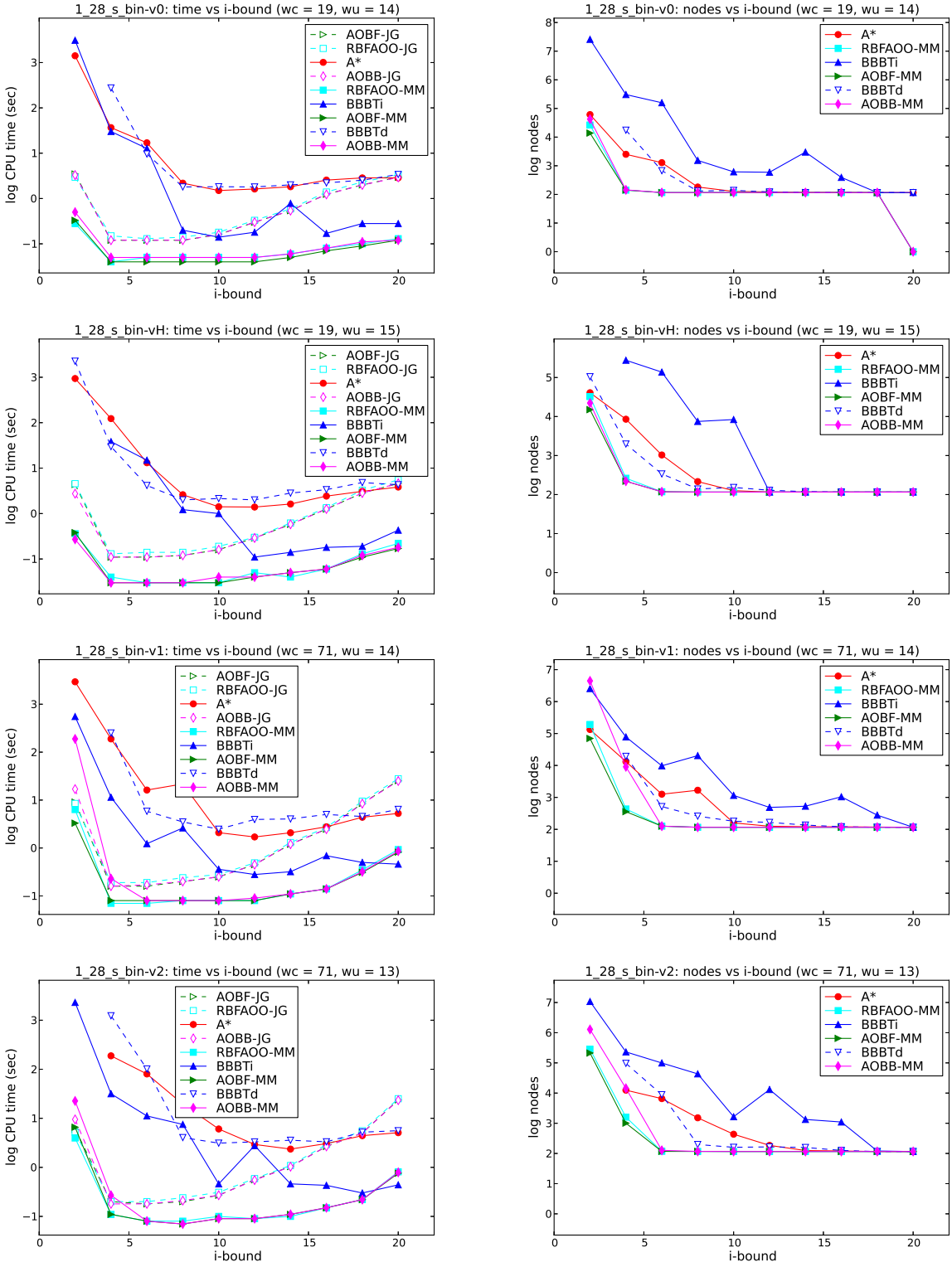

Figure 35: 1-28-s-bin instance (time and nodes)

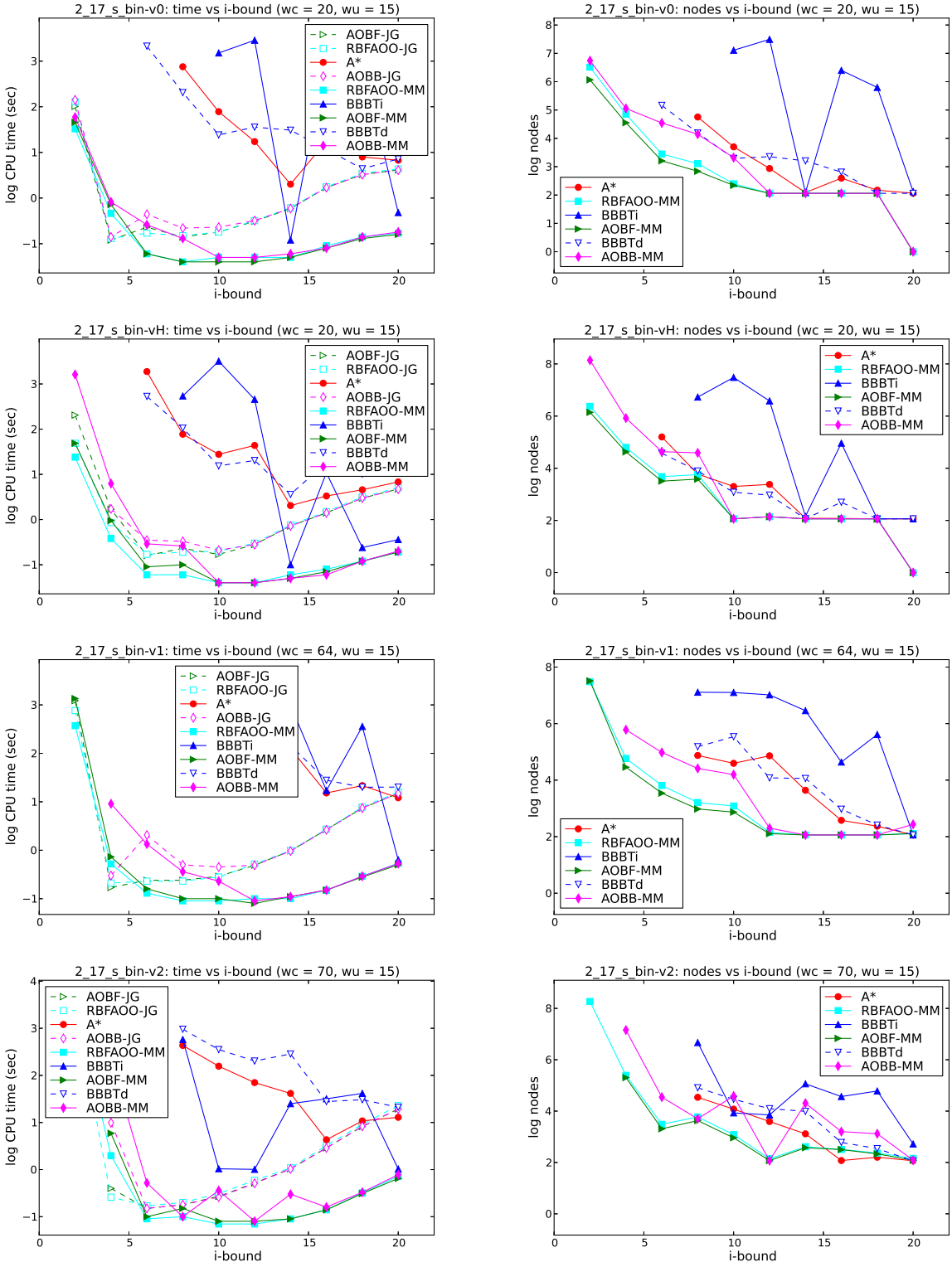

Figure 36: 2-17-s-bin instance (time and nodes)

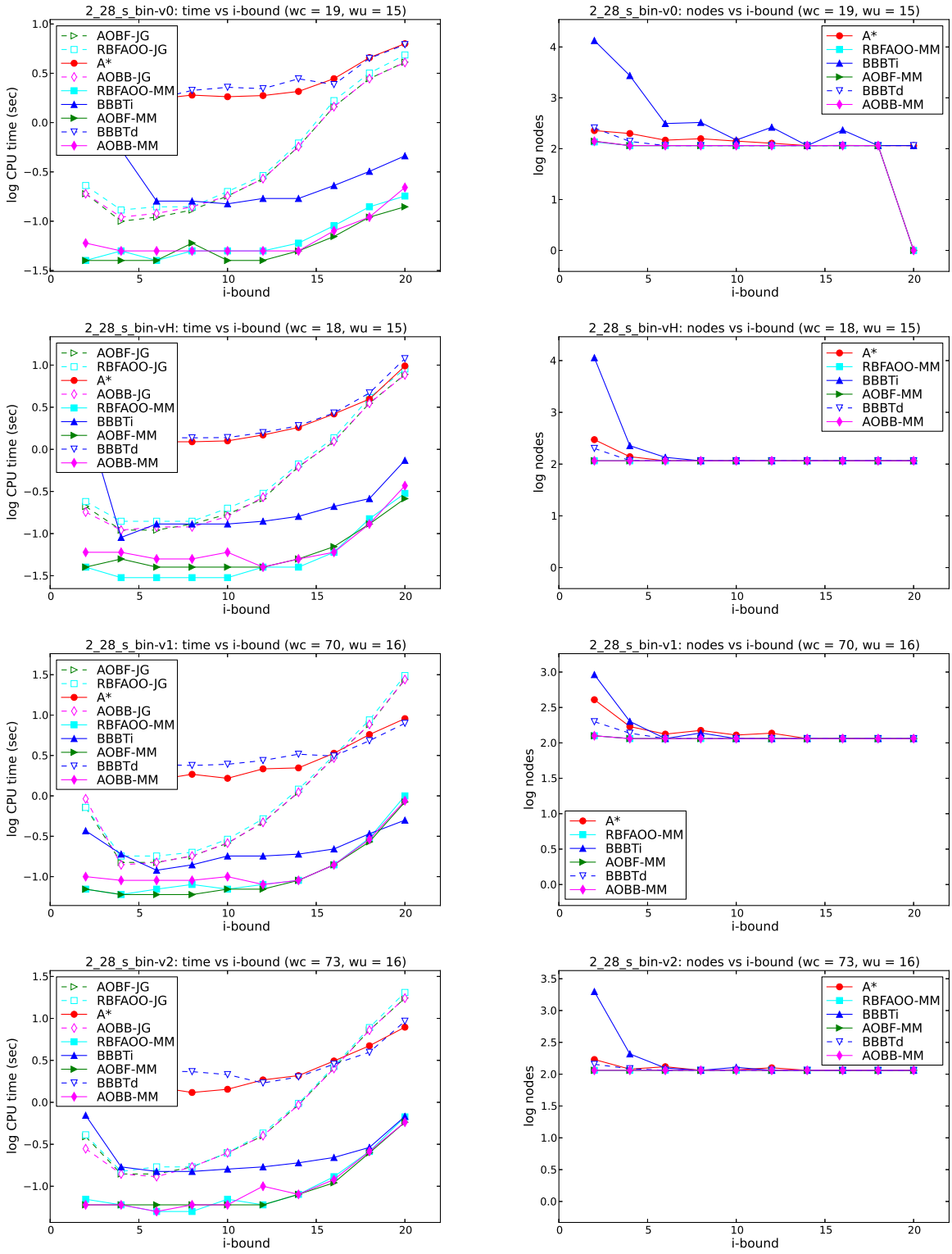

Figure 37: 2-28-s-bin instance (time and nodes)

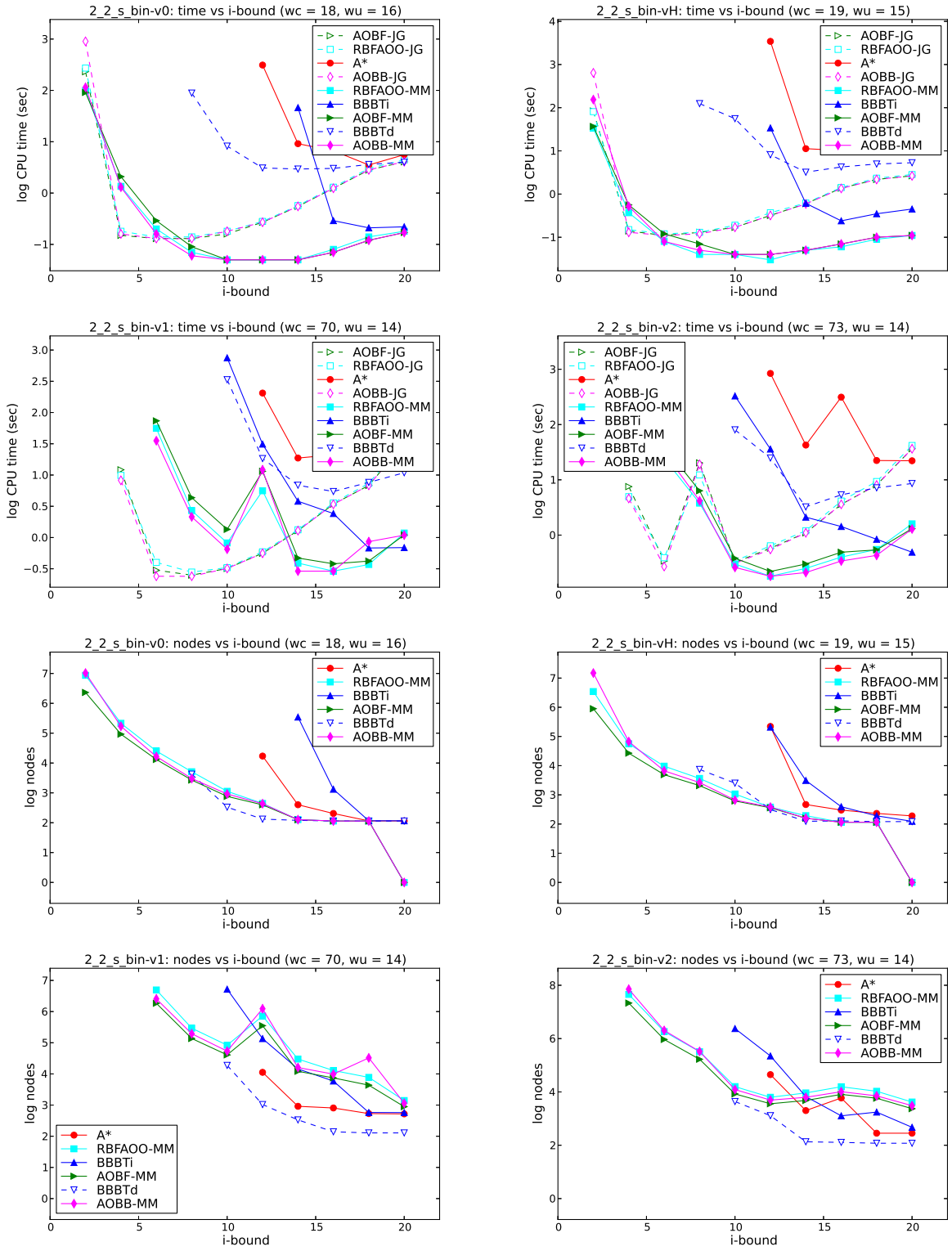

Figure 38: 2-2-s-bin instance (time and nodes)

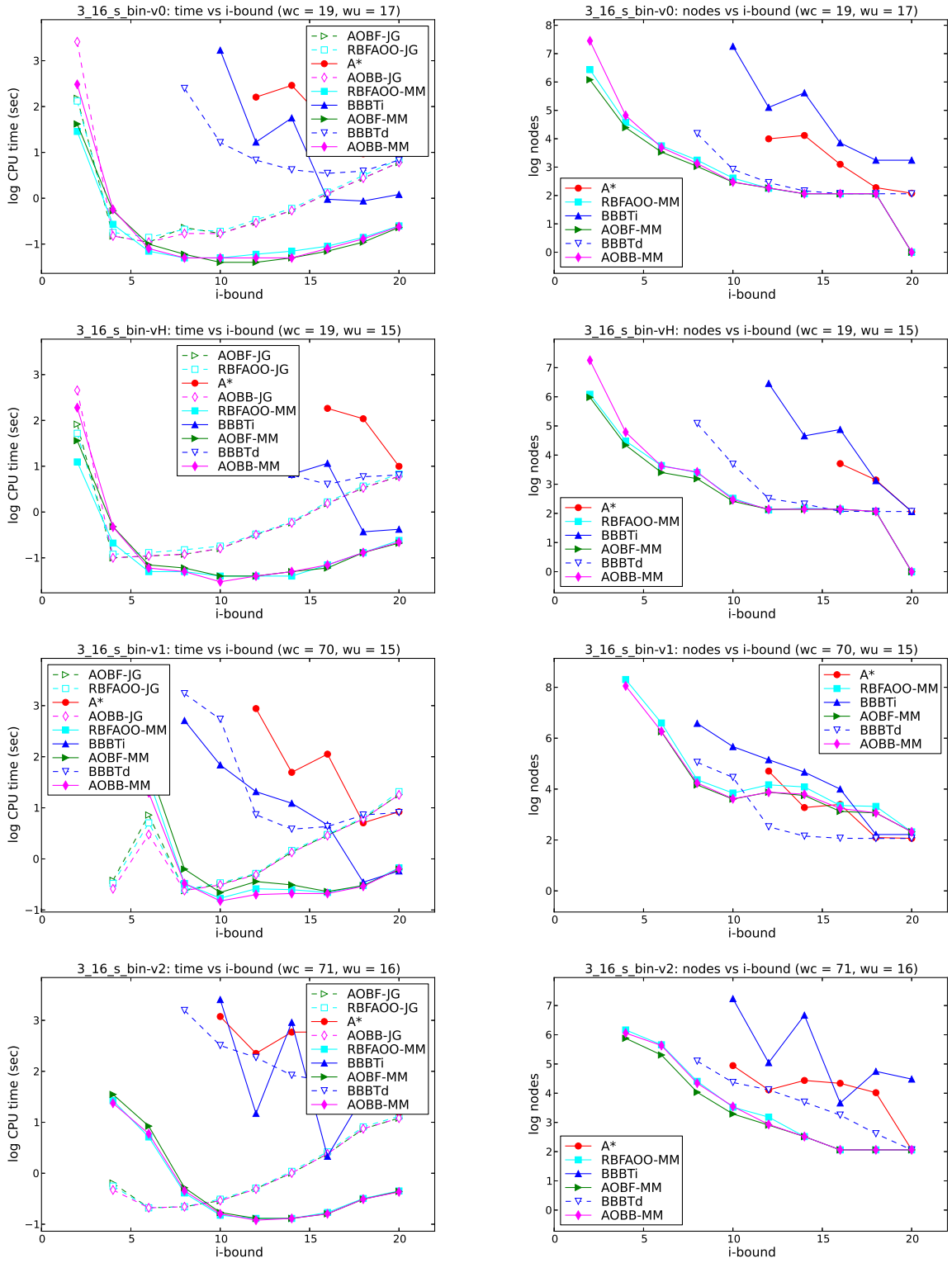

Figure 39: 3-16-s-bin instance (time and nodes)

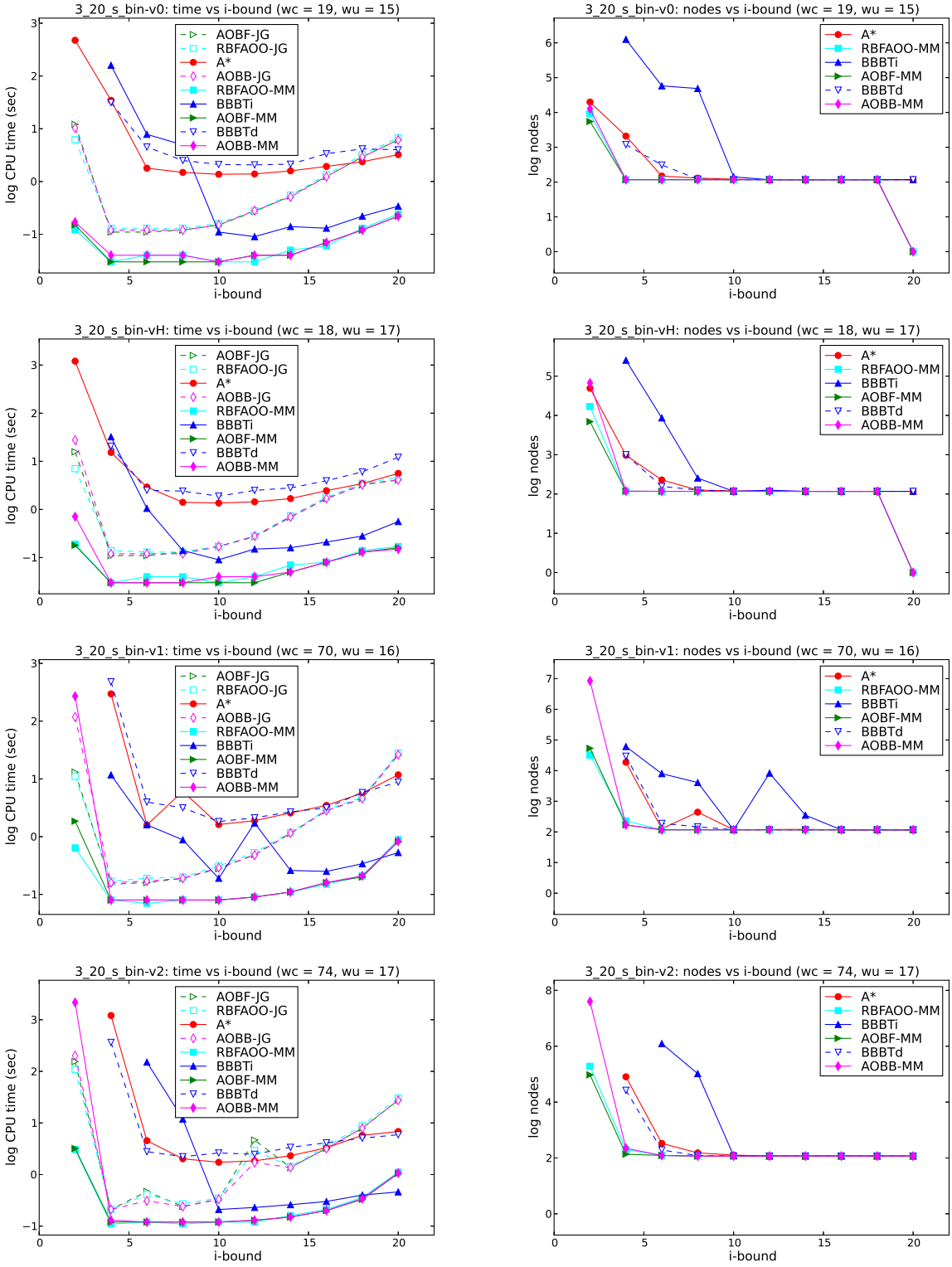

Figure 40: 3-20-s-bin instance (time and nodes)

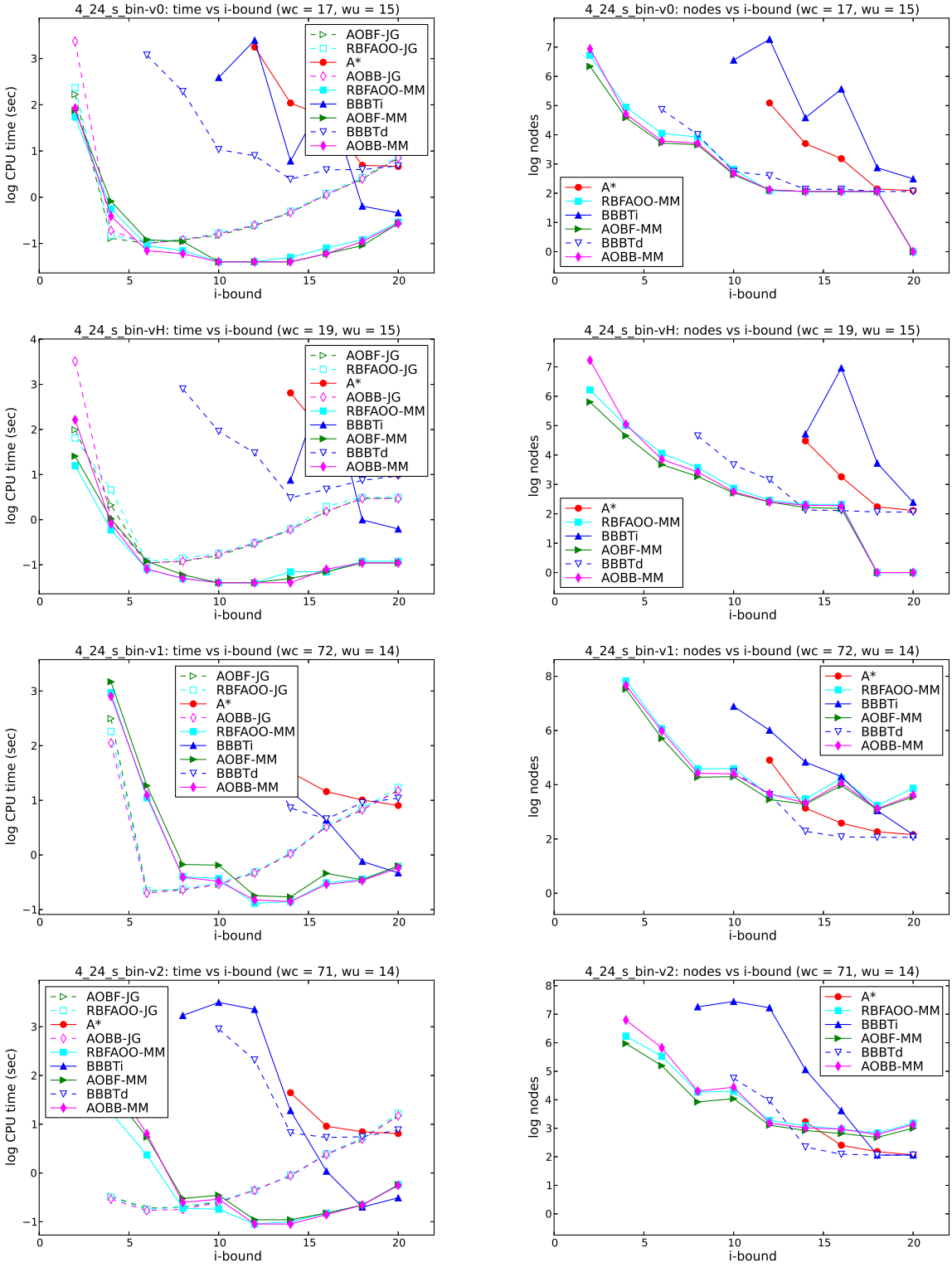

Figure 41: 4-24-s-bin instance (time and nodes)

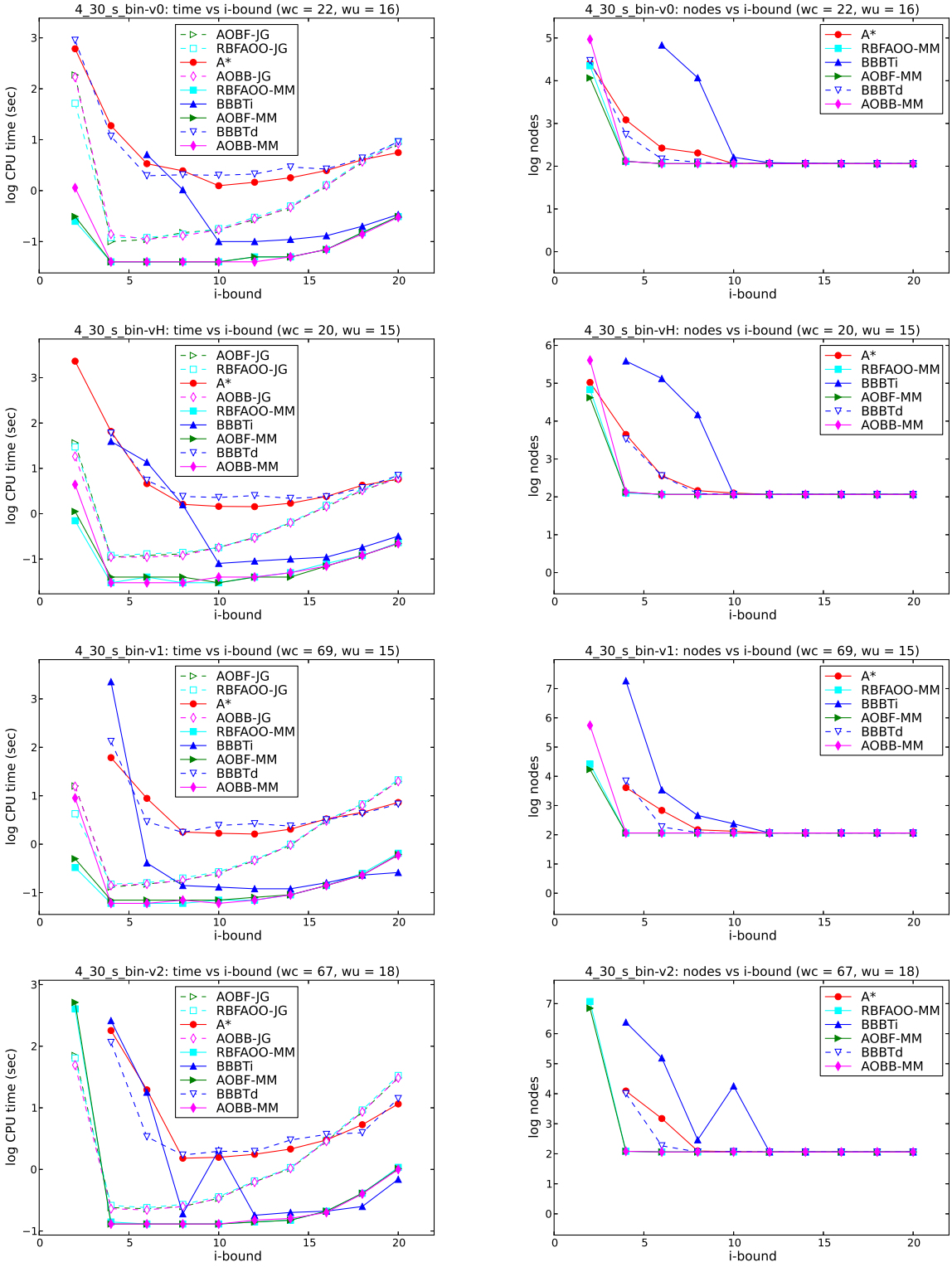

Figure 42: 4-30-s-bin instance (time and nodes)

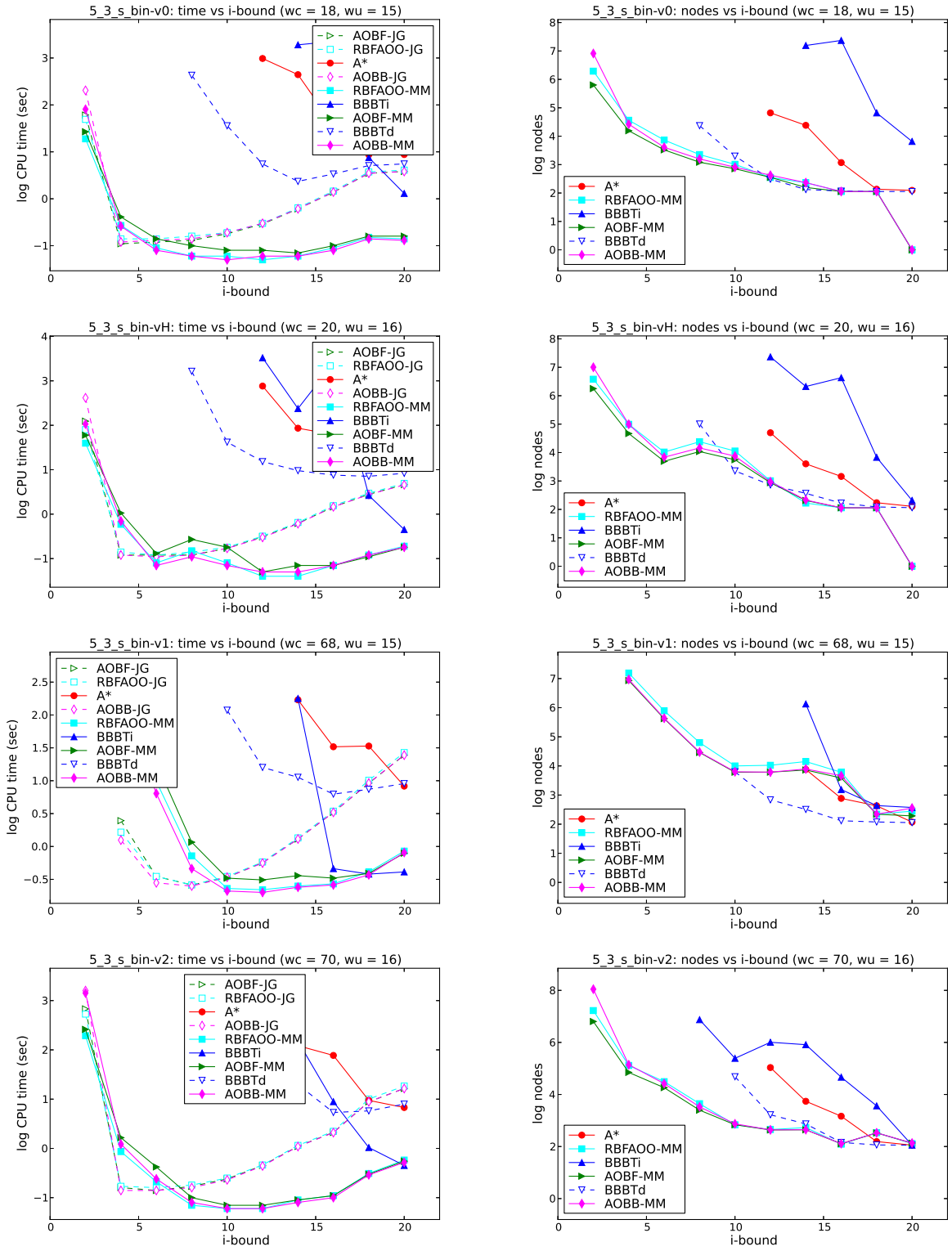

Figure 43: 5-3-s-bin instance (time and nodes)

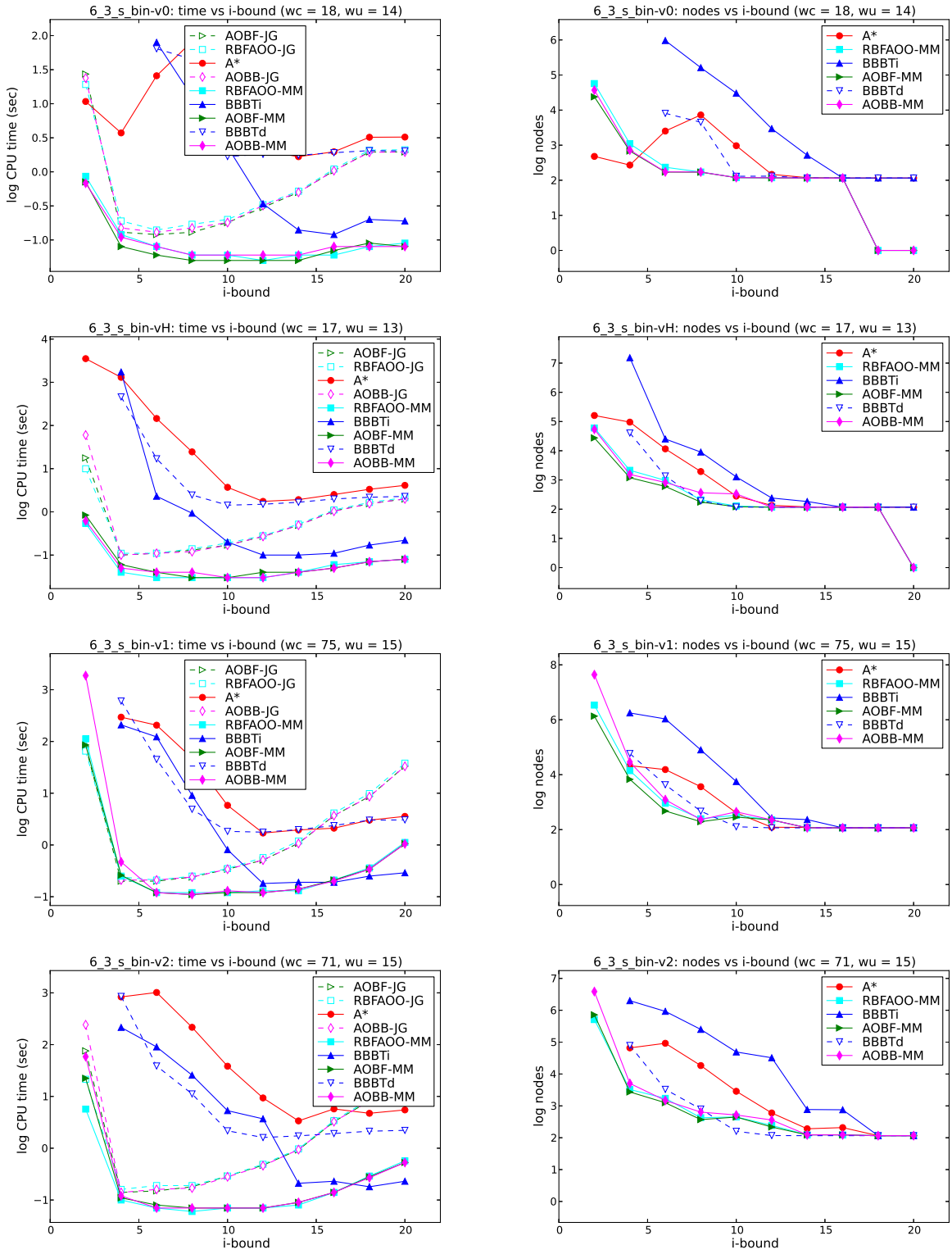

Figure 44: 6-3-s-bin instance (time)

2.5 Results for `protein` networks

Figure 45 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the grid benchmark (includes all instances).

Figure 46 shows the median CPU time (log-scale) and number of solved instances as a function of the i-bound for the `easy` as well as the `hard` instances of the grid benchmark. The rest of the figures in this section show detailed plots for each of the instances from this benchmark.

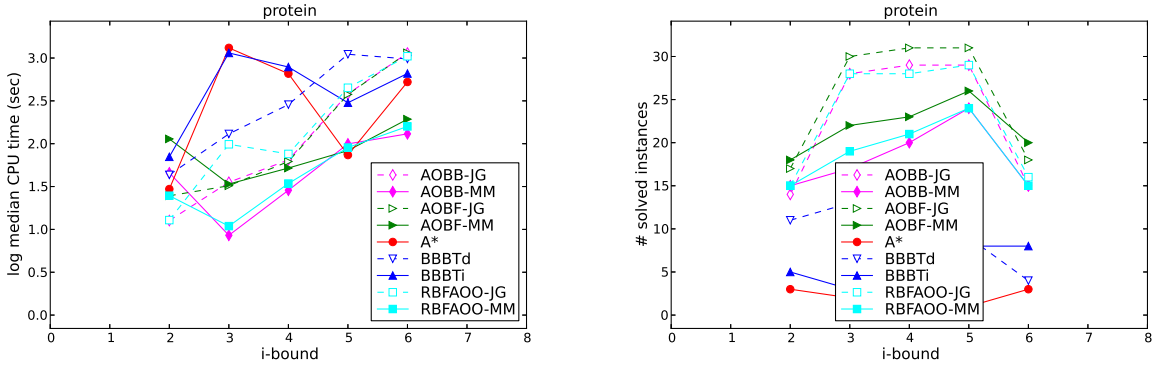

Figure 45: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `protein` benchmark.

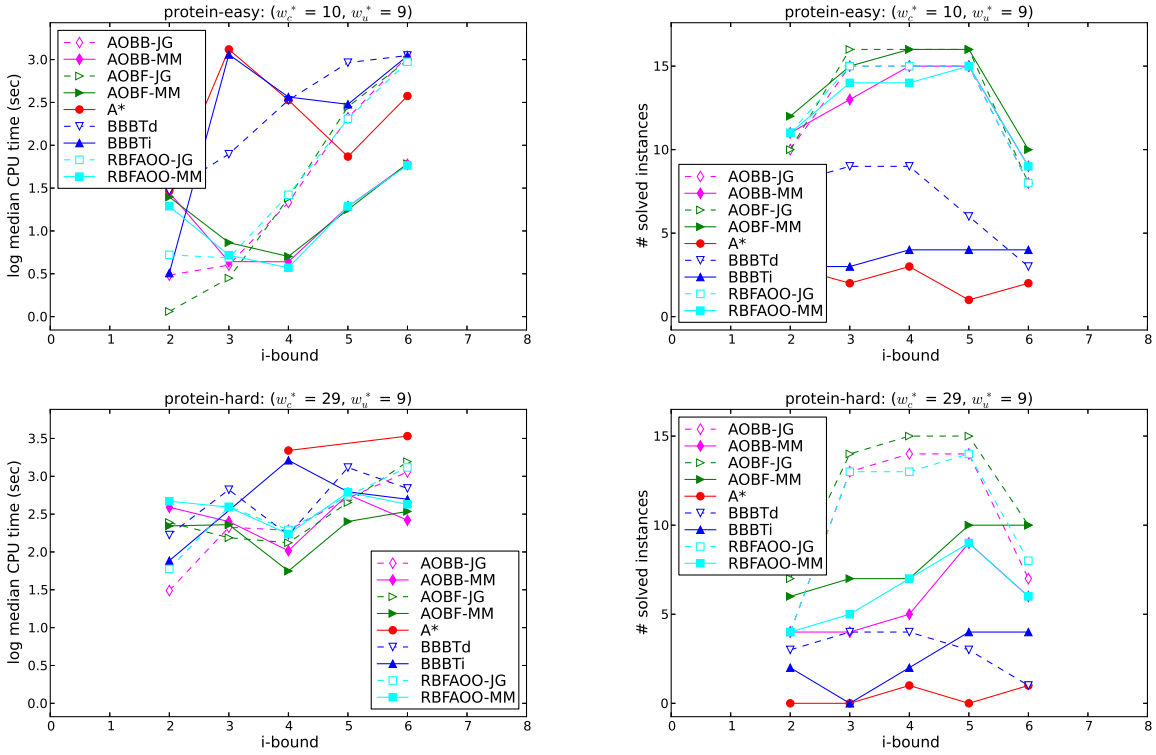

Figure 46: Log median CPU time (sec) and number of instances solved as a function of the i-bound for the `protein-easy` and `protein-hard` benchmark.

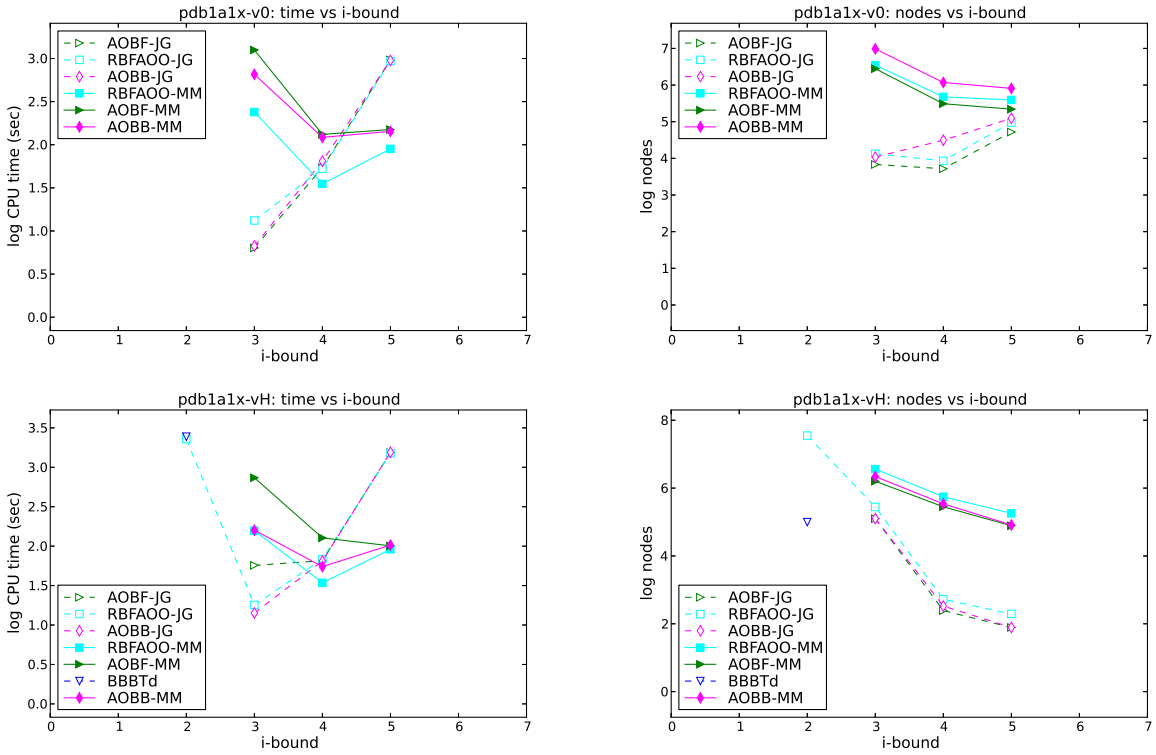

Figure 47: pdb1a1x instance (time and nodes)

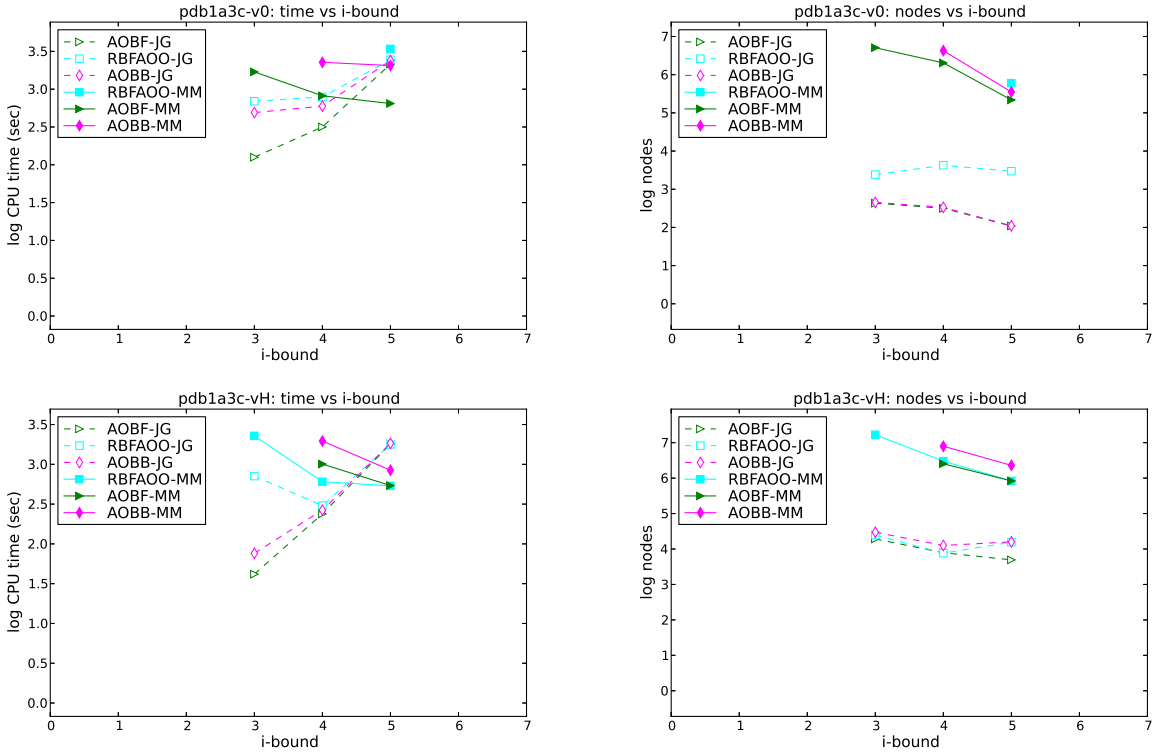

Figure 48: pdb1a3c instance (time and nodes)

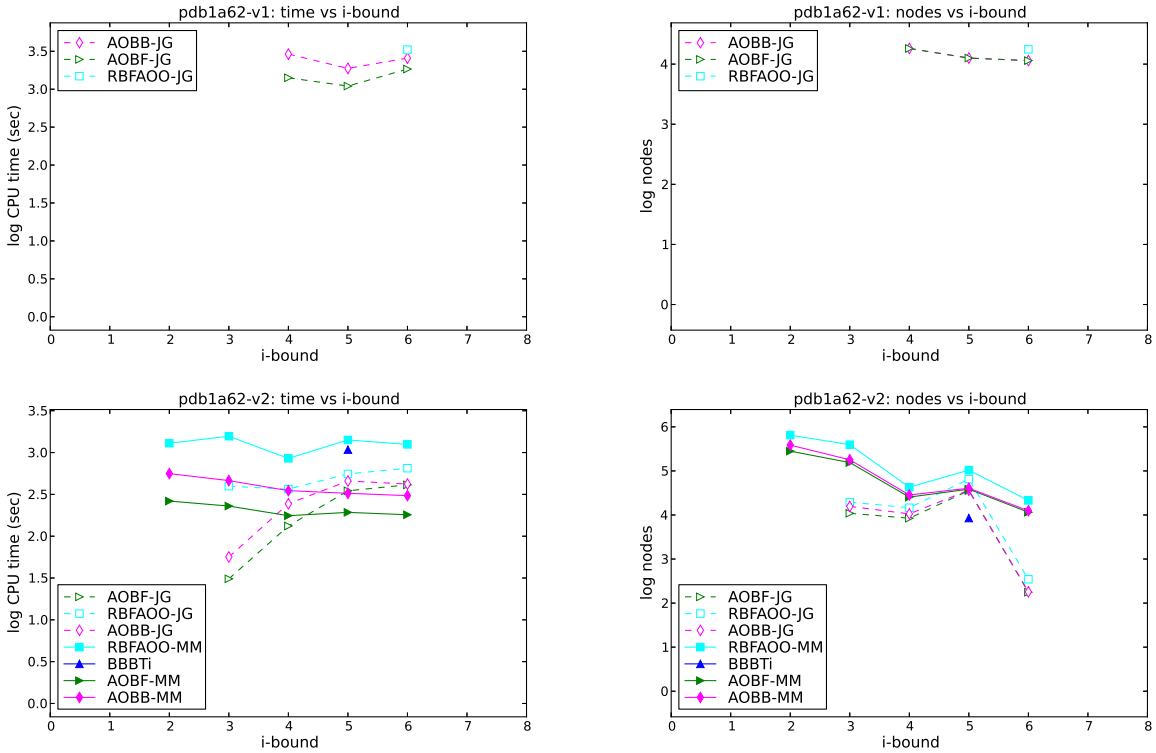

Figure 49: pdb1a62 instance (time and nodes)

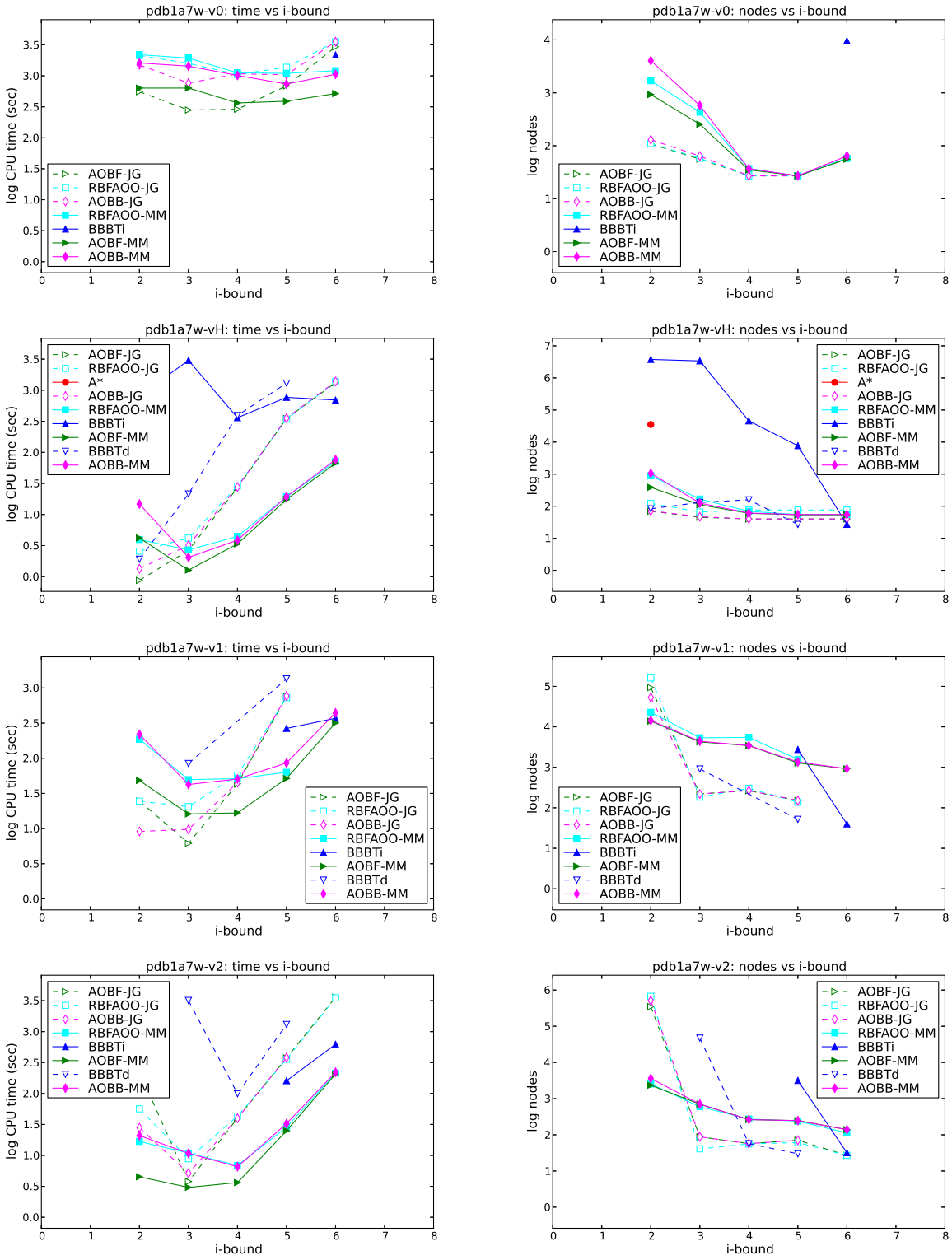

Figure 50: pdb1a7w instance (time and nodes)

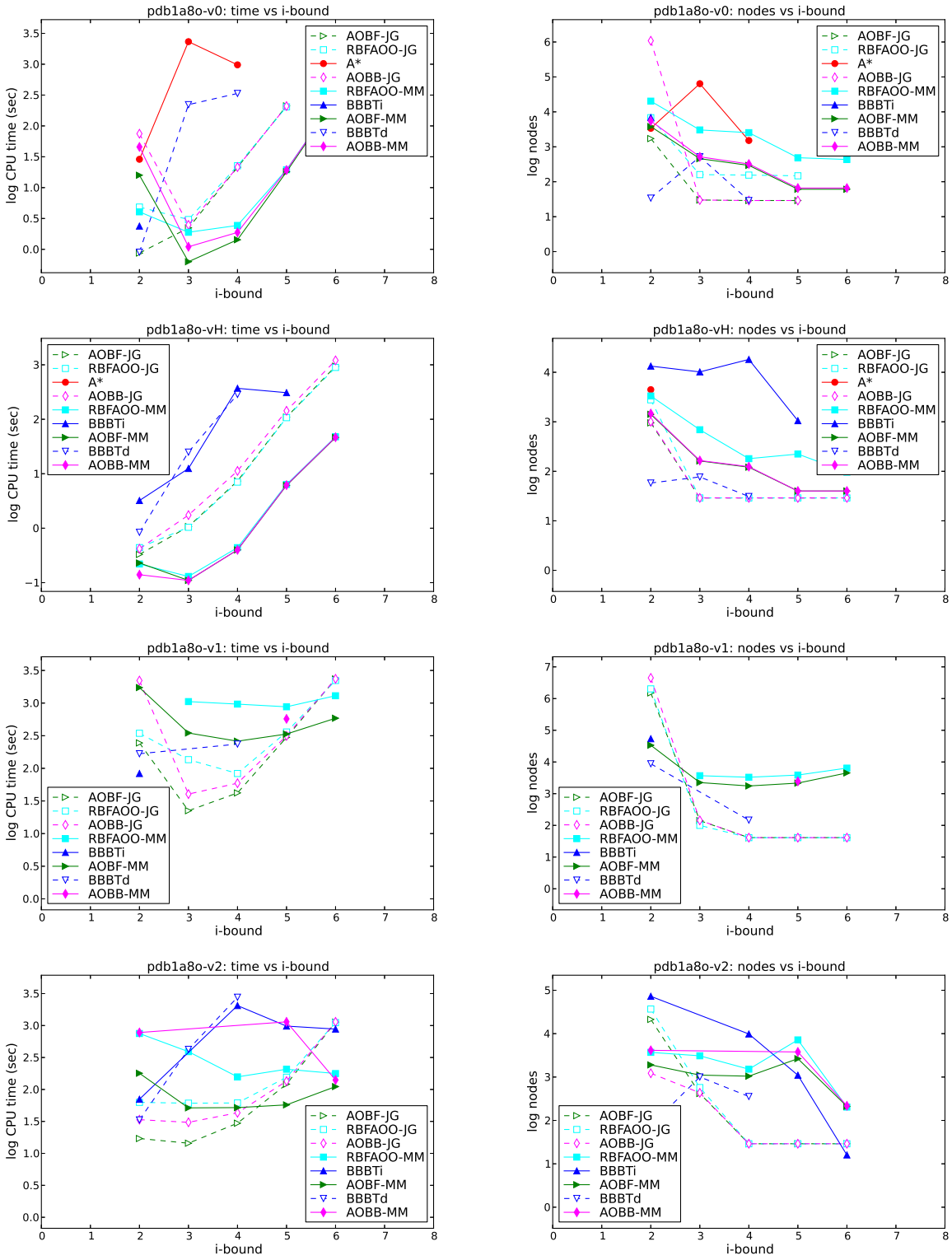

Figure 51: pdb1a8o instance (time and nodes)

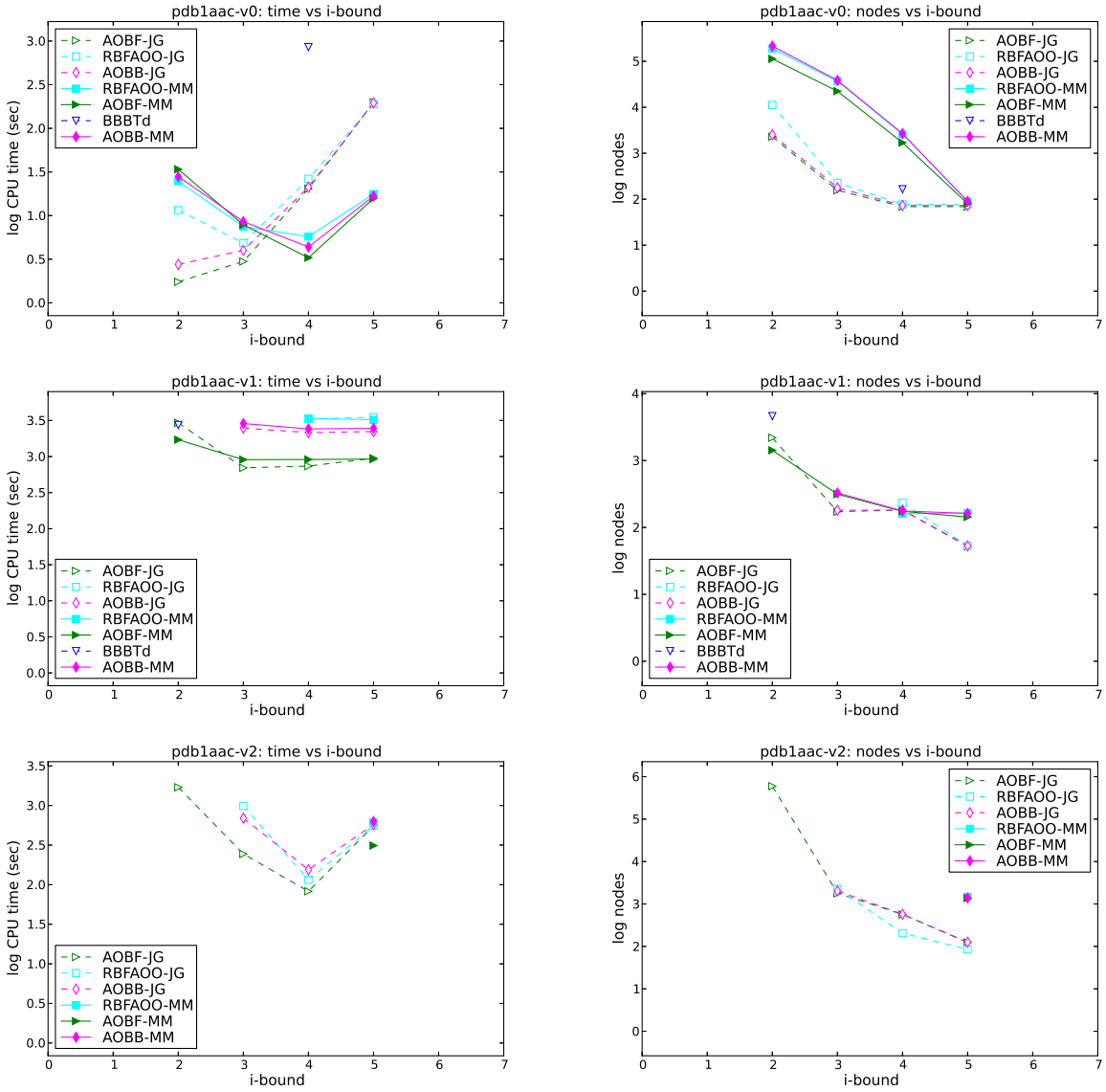

Figure 52: pdb1aac instance (time and nodes)

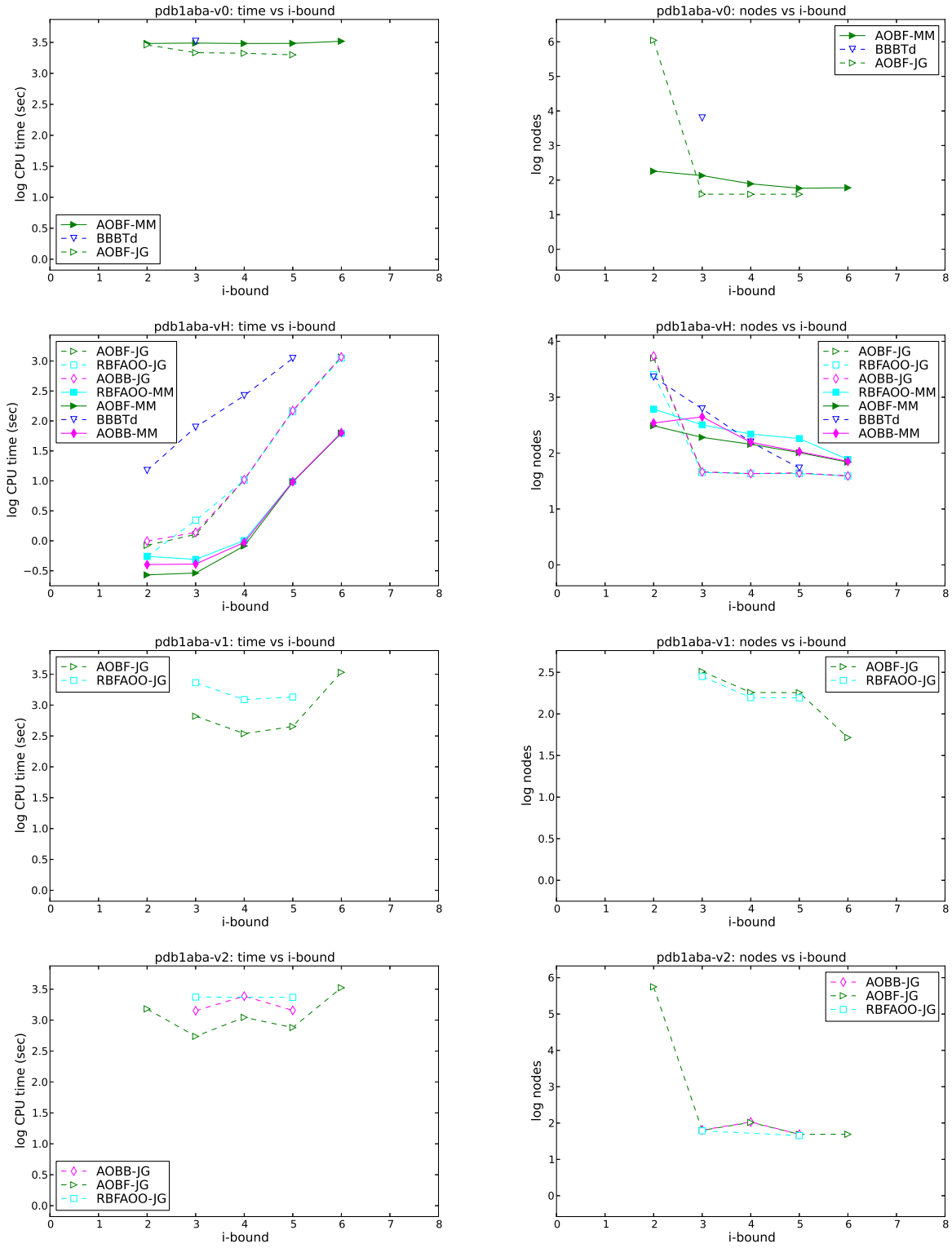

Figure 53: pdb1aba instance (time and nodes)

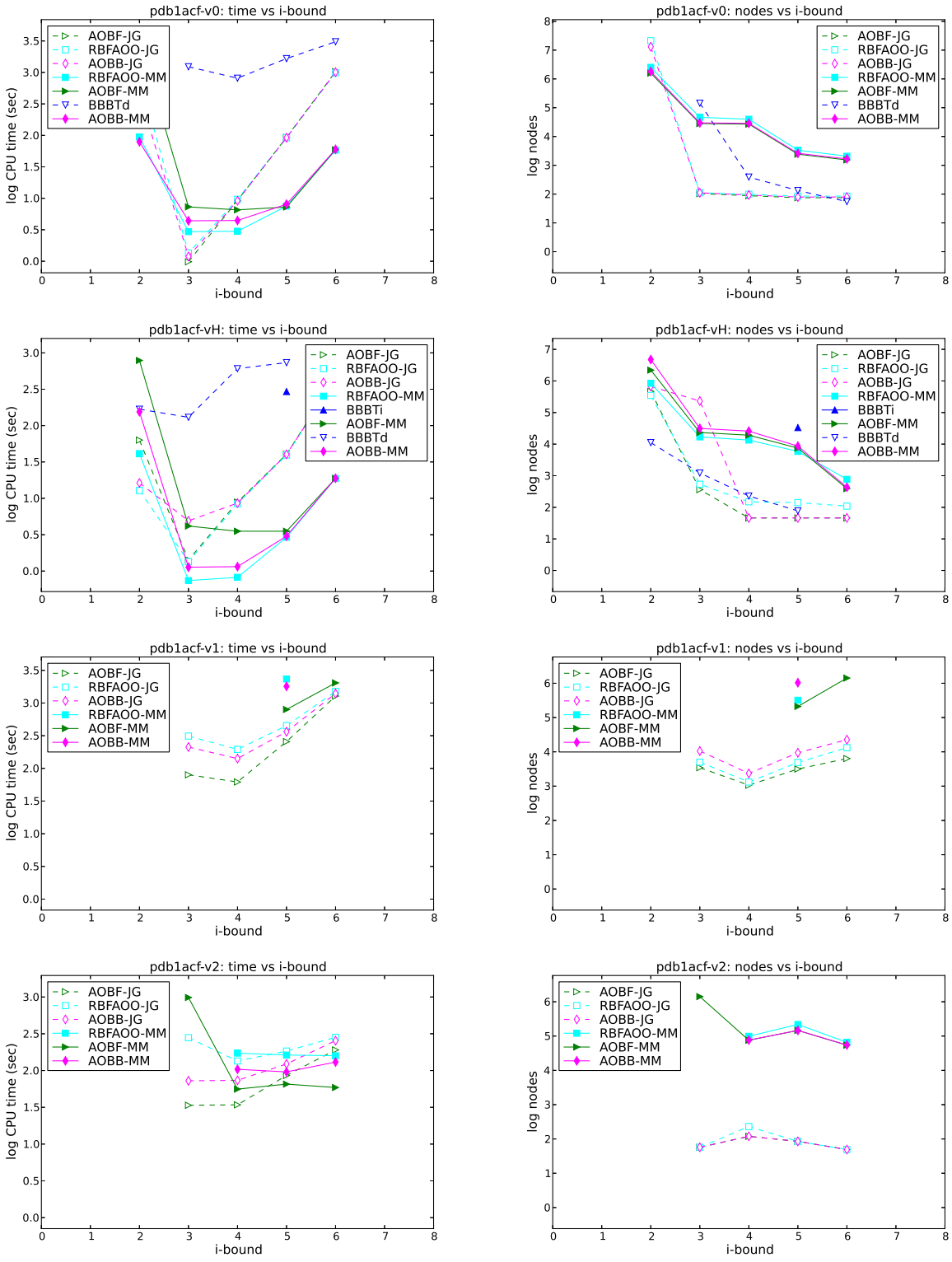

Figure 54: pdb1acf instance (time and nodes)

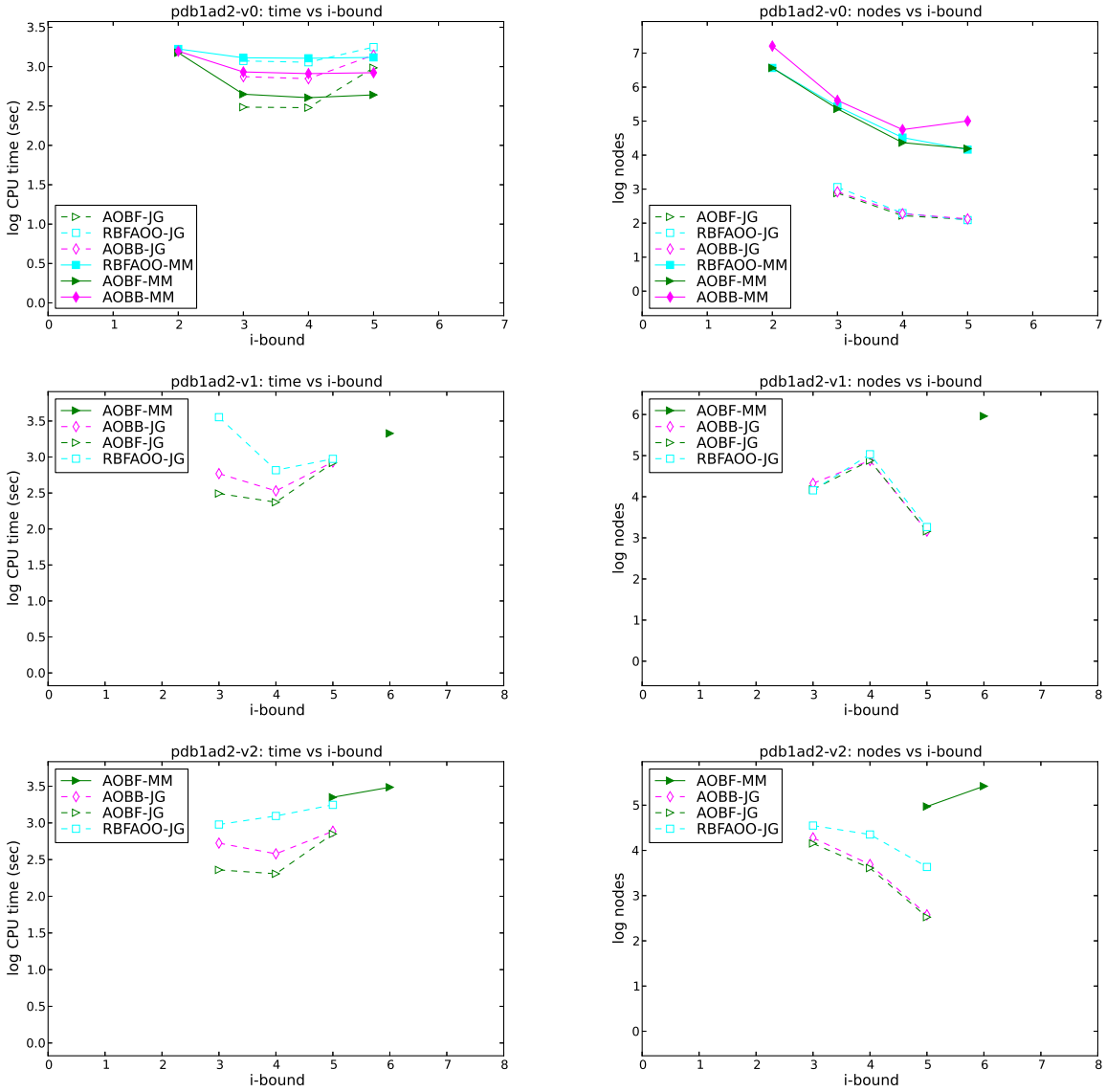

Figure 55: pdb1ad2 instance (time and nodes)

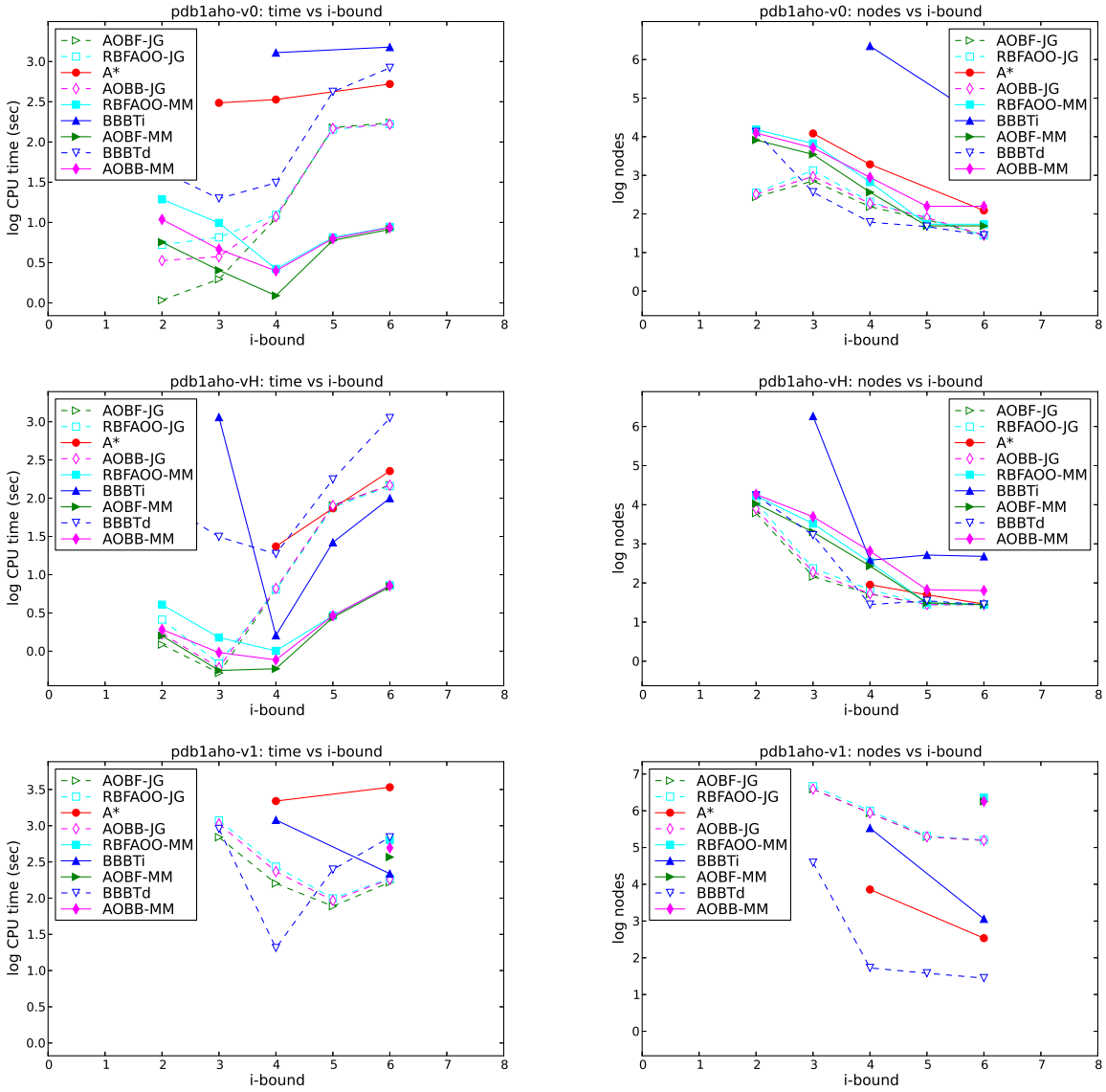

Figure 56: pdb1aho instance (time and nodes)