Combination of distributed search and multi-search in Peers-mcd.d

(System description)

Maria Paola Bonacina *

Department of Computer Science – The University of Iowa
Iowa City, IA 52242-1419, USA
E-mail: bonacina@cs.uiowa.edu

1 A Modified Clause-Diffusion prover with multi-search

Peers-mcd.d implements contraction-based strategies for equational logic, modulo associativity and commutativity, with paramodulation, simplification and functional subsumption. It is a new version of Peers-mcd [4], that parallelizes McCune's prover EQP (version 0.9d), according to the Modified Clause-Diffusion methodology (`http://www.cs.uiowa.edu/~bonacina/cd.html`).

In *parallel search* with peer processes (no master-slave hierarchy), multiple deductive processes search the space of the theorem-proving problem, each developing its own data base and derivation, and cooperate through communication of data, until one finds a proof and all halt. Within parallel search, we distinguish between *distributed search*, where the searches generated by the processes are differentiated by subdividing the inferences among them, and *multi-search*, where they are differentiated by assigning different search plans to the processes. Most approaches to parallel search in theorem proving adopted either one or the other: for instance, the systems based on Team-Work and combination of homogeneous provers emphasized multi-search, while the previous Clause-Diffusion provers emphasized distributed search (see [6] for a survey and references). A major difference between Peers-mcd.d and all its predecessors is that Peers-mcd.d implements both *distributed search* and *multi-search*, and their combination.

Peers-mcd.d can run in one of three modes:

- *Pure distributed-search mode:* the search space is subdivided among the processes; all processes execute the same search plan.
- *Pure multi-search mode:* the search space is not subdivided; every process executes a different search plan.
- *Hybrid mode:* the search space is subdivided, and the processes execute different search plans.

The basic structure of the search plan in a Peers-mcd.d process is to select premises for *expansion* (paramodulation), normalize the generated equations (*forward contraction*), and apply them to normalize pre-existing equations

* Supported in part by the National Science Foundation with grants CCR-97-01508 and EIA-97-29807.

(*backward contraction*), in such a way to keep the data base inter-reduced. Peers-mcd.d offers three ways of diversifying the search plan:

- *Different premise selection mechanism*,
- *Different ratio of breadth-first search and best-first search*, and
- *Different heuristic function to sort equations for premise selection.*

Peers-mcd.d inherits from EQP two mechanisms to select the premises for paramodulation, the *given-clause* algorithm and the *pair* algorithm. The first one is a best-first search with the weight of equations as heuristic function: at every selection extract an equation of smallest weight and generate all its paramodulants with the already selected equations. The second one is a best-first search on pairs: at every selection extract a pair of equations of smallest weight and generate all their paramodulants. The most basic way of introducing multi-search is to have some processes execute the given-clause and some the pair algorithm: in Peers-mcd.d, when the flag `diverse-sel` is set, the even-numbered processes execute the pair algorithm, and the odd-numbered processes execute the given-clause algorithm.

If the parameter `pick-given-ratio` has value x, the given-clause/pair algorithm picks the oldest, rather than lightest, equation/pair once every $x + 1$ choices. The second way of diversifying search plans in Peers-mcd.d is to let each process use a different value of `pick-given-ratio`: when the flag `diverse-pick` is set, process p_k resets its `pick-given-ratio` to $x + k$.

The third ingredient to obtain different search plans is to let the processes do best-first search with different heuristic functions. The heuristic functions of [1, 7] measure the syntactic similarity between an equation and the target theorem(s): the higher the similarity, the better the heuristic value, since an equation similar to the goal might reduce it. Peers-mcd.d implements the heuristic functions *occ-nest*, *CP-in-goal*[1] and *goal-in-CP* of [7], except that it uses the measure m_0 of [1] for the number of occurrences of a function symbol in a term, to take into account that AC operators are varyadic, since terms under AC operators are flattened. When the flag `heuristic-search` is set, process p_k executes the given-clause algorithm with heuristic function *occ-nest* if $k \ mod \ 3 = 0$, *CP-in-goal* if $k \ mod \ 3 = 1$, and *goal-in-CP* if $k \ mod \ 3 = 2$. The pair algorithm does not use these heuristic functions, because they are defined for equations, not pairs.

The search space is subdivided by subdividing the generated equations among the processes. This is achieved *without a top-level scheduler*: whenever a process generates and keeps an equation (i.e., the equation is not deleted by forward contraction), it gives it a process number, which becomes part of the equation's identifier (see [5] for details). This induces a subdivision of inferences, because each process skips the steps that it knows are done by others based on the identifiers of the premises. All inferences that generate new clauses, including backward-contraction, are thus subdivided, while deletions are not. Each process broadcasts the equations it has generated and kept after normalization. In Peers-mcd.d, the parameter `decide-owner-strat`, that controls the choice of

[1] CP stands for critical pair, hence equation.

subdivision criterion, may also have the value `no-subdivide`, meaning that no subdivision occurs, and a process broadcasts an equation only if its weight (its heuristic value if `heuristic-search` is set) is lower than a given parameter.

In summary, if `decide-owner-strat` = `no-subdivide`, and at least one of `diverse-sel`, `diverse-pick` and `heuristic-search` is set, Peers-mcd.d runs in pure multi-search mode; if `decide-owner-strat` ≠ `no-subdivide`, and none of `diverse-sel`, `diverse-pick` and `heuristic-search` is set, Peers-mcd.d runs in pure distributed-search mode; if `decide-owner-strat` ≠ `no-subdivide`, and at least one of `diverse-sel`, `diverse-pick` and `heuristic-search` is set, Peers-mcd.d runs in hybrid mode.

2 Proofs of the Moufang identities without cancellation

The first automated proofs of the Moufang identities in alternative (i.e., non-associative) rings by a general-purpose prover were presented in [2]. They used AC-UKB, the *inequality ordered-saturation* inference rule (i.e., superposition of an un-orientable equation into a goal to generate a new goal which is kept only if its normal form is not greater or equal than an already existing inequality), inference rules that *build the cancellation laws in* [8], and the heuristic measures of [1] to sort equations and delete those whose heuristic value is worse than a given threshold. These problems are still used as benchmarks (e.g., [3]) and in competitions (e.g., [9]). The TPTP library presents them in different formulations: some differ from [2] in choice of axioms and/or conjecture; those that follow [2] include the cancellation laws as implications, so that they are not equational. In the experiments reported here, the problems were formulated as in [2], but *without cancellation laws*, since EQP and Peers-mcd.d do not implement the rules of [8], and they are purely equational provers which cannot handle implications.

In the following tables, the first column tells the mode: D for pure distributed-search mode and H for hybrid mode. The second column tells the search plan, given-clause, or pair, or *diverse*, if `diverse-sel` was set. The h means that `heuristic-search` was used. The number at the front is the `pick-given-ratio`: x if it was x for all processes, xd, if `diverse-pick` was set and process p_k used $x + k$, nothing, if `pick-given-ratio` was not used. The number in parenthesis is the value of `max-weight`, if deletion by weight was used. The times (expressed in sec) are *average CPU times*. For each search plan, five subdivision criteria were tried, and the best result (among the averages) was retained. "T" means time-out after 3600 sec. The workstations were HP B2000 or C360, with 1G or 512M of memory, with EQP0.9d running on a B2000 with 1G, and *N-Peers* (Peers-mcd.d with N processes) on N workstations, one per process.

The first two problems, **moufang1** (*Middle Alternative Law*) and **moufang2** (*Skew-Symmetry Relation of the Associator*), are too easy for parallelization: EQP0.9d proved them in 4 and 1 sec, respectively, using the *pair* algorithm with `pick-given-ratio` = 4. However, with the default search plan, namely *given-clause* algorithm and no `pick-given-ratio`, EQP0.9d terminated abnormally[2],

[2] Some constant in the AC-matching or AC-unification code of EQP was exceeded.

whereas 1-Peer did both problems in 1 sec, due to the heuristic function used by the given-clause algorithm.

For the *Left Moufang Identity* (**moufang3**), EQP0.9d could not find a proof with the default search plan, while Peers-mcd.d did, thanks to distributed search:

Mode	*Search plan*	*EQP0.9d*	*1-Peer*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
D	given(32)	T	T	598	91	187	40
H	given-h(32)	T	415	230	57	42	**9**
D	pair(32)	3,215	3,277	551	109	51	83
D	4-pair(32)	956	1,068	126	**38**	56	58
D	2-pair(32)	88	130	66	39	109	25
H	2d-diverse-h(32)	88	147	84	75	41	25

With `heuristic-search` on (second row), also 1-Peer found a proof, which shows the merit of the heuristic function, and all other times improved, up to a proof in only 9 sec with 8-Peers. EQP found a proof with the pair algorithm (third row), and the sequential time was reduced with `pick-given-ratio` = 4 (fourth row), but Peers-mcd.d with more than one node sped-up with these search plans also, finding a proof in 38 sec with 4 processes. The best sequential time was obtained with `pick-given-ratio` = 2 (last two rows): with this value, the parallel prover behaved more smoothly in hybrid mode.

For the *Right Moufang Identity* (**moufang4**), EQP found a proof only with the pair algorithm and `pick-given-ratio` = 4:

Mode	*Search plan*	*EQP0.9d*	*1-Peer*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
H	given-h(32)	T	437	268	162	100	**28**
D	pair(32)	T	T	865	356	161	105
H	4d-diverse-h(32)	1,558	1,638	75	**32**	27	47

The problem proved to be elusive for the default search plan, but with `heuristic-search` on (first row), Peers-mcd.d solved it, with run-time decreasing down to 28 sec for 8-Peers. With the pair algorithm and `pick-given-ratio` not set (second row), 1-Peer did like EQP, since the pair algorithm does not use the heuristic function, but the parallel prover succeeded. With the hybrid search plan *4d-diverse-h*, Peers-mcd.d exhibited super-linear speed-up for all numbers of processes, with the best result for 4-Peers: the speed-up was $1,558/32 = 48.68$ and the efficiency $48.68/4 = 12.17$.

For the *Middle Moufang Identity* (**moufang5**), EQP could not find a proof within 3,600 sec with the default search plan, and took 572 sec with the pair algorithm, while Peers-mcd.d was much faster: using the default search plan, but with `heuristic-search` on, hence in hybrid mode, 1-Peer found a proof in 16 sec, 2-Peers took 9 sec and 4-Peers only 5 sec.

Problems *moufang3*, *moufang4* and *moufang5* were tried also in pure multi-search mode, with the same search plans tried in hybrid mode, but no subdivision and equation broadcasting limited by heuristic value. Almost no speed-up was observed. Thus, distributed search did much better than multi-search on these problems, and the combination of the two did even better. This suggests that a key factor in parallel search, possibly even more basic than limiting communication, is to differentiate the processes, so that they *do not overlap* and explore different parts of the search space. The statistics showed that a speed-up is typically accompanied by a strong reduction in number of equations generated (for Peers-mcd.d, the sum of the equations generated by all peers), hinting that the subdivision was effective, and led the processes to generate different searches and different from the sequential one.

Directions for future work include the development of a Modified Clause-Diffusion prover for first-order logic with equality, to allow application to a larger class of problems.

Acknowledgements Thanks to Bill McCune for EQP0.9d, to my former student Javeed Chida, for implementing the heuristic functions in his master thesis, and to Gigina Carlucci Aiello of the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," where part of this work was done.

References

1. Siva Anantharaman and Nirina Andrianarivelo. Heuristical criteria in refutational theorem proving. In Alfonso Miola, editor, *Proceedings of the 1st DISCO*, volume 429 of *LNCS*, pages 184–193. Springer Verlag, 1990.
2. Siva Anantharaman and Jieh Hsiang. Automated proofs of the Moufang identities in alternative rings. *Journal of Automated Reasoning*, 6(1):76–109, 1990.
3. Jürgen Avenhaus, Thomas Hillenbrand, and Bernd Löchner. On using ground joinable equations in equational theorem proving. In Peter Baumgartner and Hantao Zhang (ed.), Proceedings of FTP 2000, Technical Report 5/2000, Institut für Informatik, Universität Koblenz-Landau, 33–43, 2000.
4. Maria Paola Bonacina. The Clause-Diffusion theorem prover Peers-mcd. In William W. McCune, editor, *Proceedings of CADE-14*, volume 1249 of *LNAI*, pages 53–56. Springer, 1997.
5. Maria Paola Bonacina. Experiments with subdivision of search in distributed theorem proving. In Markus Hitz and Erich Kaltofen, editors, *Proceedings of PASCO97*, pages 88–100. ACM Press, 1997.
6. Maria Paola Bonacina. A taxonomy of parallel strategies for deduction. *Ann. of Math. and AI*, in press, 2000. Available as Tech. Rep., Dept. of Computer Science, Univ. of Iowa from `http://www.cs.uiowa.edu/~bonacina/distributed.html`.
7. Jörg Denzinger and Matthias Fuchs. Goal-oriented equational theorem proving using Team-Work. In *Proceedings of the 18th KI*, volume 861 of *LNAI*, pages 343–354. Springer, 1994.
8. Jieh Hsiang, Michaël Rusinowitch, and Ko Sakai. Complete inference rules for the cancellation laws. In *Proceedings of the 10th IJCAI*, pages 990–992, 1987.
9. Geoff Sutcliffe. The CADE-16 ATP system competition. *Journal of Automated Reasoning*, 24:371–396, 2000.