Smart Selective Competition Parallelism ATP

Geoff Sutcliffe and Darryl Seyfang

James Cook University
Townsville, Queensland, Australia 4811
geoff@cs.jcu.edu.au

Abstract

This paper describes SSCPA, an uncooperative multiple calculus competition parallelism ATP system, that multitasks on a single CPU. SSCPA runs multiple sequential ATP systems in parallel, using performance data from the ATP systems to select those that are best suited to the problem.

Introduction

Automated Theorem Proving (ATP) is concerned with the use and development of systems which automate sound reasoning: the derivation of conclusions that follow inevitably from facts. This capability lies at the heart of many important computational tasks. One type of ATP system is fully automatic 1st order ATP systems, such as Bliksem (De Nivelle 1998), Gandalf (Tammet 1997), Otter (McCune 1994), SETHEO (Letz et al. 1992), and SPASS (Weidenbach, 1997). Such systems are capable of solving non-trivial problems, e.g., EQP solved the Robbins problem (McCune 1997). However, in practice, the search complexity of most interesting problems is enormous, and many problems cannot currently be solved within realistic resource limits. Therefore a key concern of ATP research is the development of more powerful systems, capable of solving more difficult problems within the same resource limits.

One approach to developing more powerful ATP systems is the use of parallelism, where multiple tasks work together towards solving a problem. The parallelism may be true parallelism on multiple CPUs, or multitasking on a single CPU. Suttner and Schumann (1994) provide an overview of parallel ATP, classifying parallel ATP systems on two axes. The first axis considers whether the problem is partitioned between tasks (partitioning parallelism), or whether the same problem is given to each task to attempt in a different way, with all systems being stopped when any one finds a solution (competition parallelism). Competition parallelism is divided into two types, dependant on whether a single or multiple proof calculi are used. The second axis in the classification scheme considers whether or not the tasks cooperate by sharing data. Example parallel ATP systems are METEOR (Astrachan 1994) - uncooperative partitioning parallelism, ROO (Lusk and McCune 1992) - cooperative partitioning parallelism, p-SETHEO (Wolf and Letz 1998) - uncooperative single calculus competition parallelism, and CPTHEO (Fuchs and Wolf 1998) - cooperative multiple calculus competition parallelism.

This paper describes the Smart Selective Competition Parallelism ATP system SSCPA[1], an uncooperative multiple calculus competition parallelism ATP system, that multitasks on a single CPU. SSCPA runs multiple sequential ATP systems in parallel. This approach is motivated by the observation that no individual ATP system performs well on all problems. SSCPA uses performance data from the ATP systems to select those that are best suited to the problem, to run in parallel. The following sections describe the underlying concepts, current implementation, and computed results for SSCPA.

Underlying Concepts

ATP system performance data, e.g., the results of the CADE ATP system competitions (Sutcliffe and Suttner 1997, Suttner and Sutcliffe 1998, Sutcliffe and Suttner 1998c), shows that no single sequential ATP system performs well on all problems. There is convincing evidence that the specialisation of ATP systems is due to the deduction techniques used in relation to the syntactic characteristics of the problems. This is well demonstrated with respect to the variety of deduction techniques used by Gandalf and Otter (Suttner and Sutcliffe 1998). This observation provides the motivation for SSCPA's design. SSCPA uses the syntactic characteristics of the given problem to classify it into one of 18 predefined problem classes. SSCPA then uses performance data from the available sequential ATP systems to determine which of the systems perform well for that problem class. The recommended systems are then run in parallel, in one of several user selectable modes.

The use of feature vectors to select search strategies to run in parallel has been used previously in ATP, e.g., p-SETHEO parallelizes selected search strategies for SETHEO, and (Fuchs 1997) describes an application to the DISCOUNT system. The idea has also been used in general AI search (Cook and Varnell 1998). SSCPA appears to be the first ATP system that selects multiple ATP systems, with potentially very different calculi, to run

[1] SSCPA is pronounced as "skipper", but with an Australian accent.

in parallel. Further, the features used by SSCPA seem to be stronger than those used by the systems mentioned

For SSCPA to succeed, it is necessary to identify appropriate problem classes, to perform careful evaluation of the ATP systems within the classes, and to provide appropriate forms of competition parallelism.

Specialist Problem Classes

Specialist Problem Classes (SPCs) are syntactically identifiable classes of problems for which certain ATP techniques or systems have been observed to be especially well suited. The choice of what syntactic characteristics are used to form the classes is based on community input and analysis of system performance data. For example, everyone agrees that special techniques are deserved for problems with equality, and the CASC-15 results (Sutcliffe and Suttner 1998c) show that problems with true functions, i.e., of arity greater than zero, should be treated differently from those with only constants. The range of characteristics that so far appear to be relevant are:

- Theoremhood: Theorems vs Non-theorems
- Order: Variables vs No variables (essentially 1st order vs Propositional)
- Equality: No equality vs Some equality vs Pure equality
- Functions: True functions vs Constants only
- Form: CNF vs FOF
- Horness: Horn vs Non-Horn
- Unit equality: Unit equality vs Non-unit pure equality

Based on these characteristics 18 SPCs have been defined, as indicated by the leaves of the tree in Figure 1.

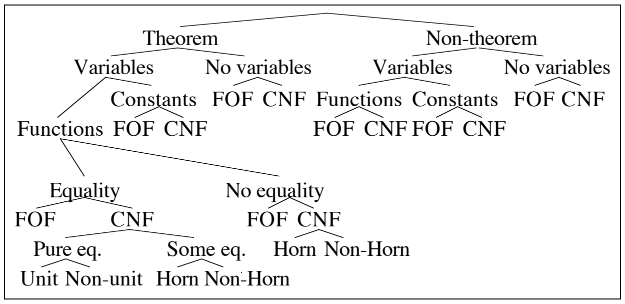

Figure 1: Specialist Problem Classes

ATP System Evaluation using the TPTP

The evaluation of ATP systems is done with respect to the SPCs. This allows the specialist capabilities of ATP systems to be identified, while overall capabilities can be inferred from the separate SPC capabilities. For SSCPA, the evaluation within SPCs provides the necessary information for recommending ATP systems.

The TPTP (Thousands of Problems for Theorem Provers) problem library is a library of test problems for 1st order ATP systems (Sutcliffe and Suttner 1998a). Since its first release in 1993, many researchers have used the TPTP as an appropriate and convenient basis for ATP system evaluation. Although other test problems do exist and are sometimes used, the TPTP is now the de facto standard for evaluating 1st order ATP systems. Some researchers, who have tested their ATP systems over the entire TPTP, have made their performance data available. The data has been collected (Sutcliffe and Suttner 1998b), and is used for evaluating the systems.

The collected performance data is provided by the individual system developers. This means that the systems have been run on a range of hardware, and using a range of CPU time limits on each solution attempt. This might suggest that the data cannot be used to compare the systems. However, analysis shows that differences in hardware and CPU time limits do not significantly affect which problems are solved by each ATP system. Figure 2 illustrates this point. Figure 2 plots the CPU times taken by each of the systems used in SSCPA, for each solution found, in increasing order of time taken. The relevant feature of these plots is that each has a point at which the time taken to find solutions starts to increase dramatically. Evidently a linear increase in the computational resources would not lead to the solution of significantly more problems. Thus the collected performance data allows a realistic evaluation of the systems, based on the problems that each system has solved.

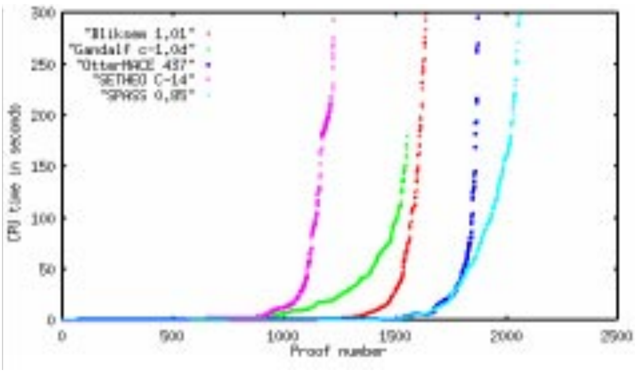

Figure 2: Proof number vs CPU time for ATP systems

To rate ATP systems, their collected TPTP performance data is analysed. Firstly, problems that are known to be designed specifically to be suited or ill-suited to any particular ATP system, calculus, or control strategy, are excluded. The remaining *unbiased* problems are divided into their SPCs. Within each such SPC, systems that solve the same problems are placed in equivalence classes, with each class being represented by an arbitrarily chosen member. A partial order between systems (and hence the equivalence classes) is determined according to whether or not a system solves a strict superset of the problems solved by another system. If a strict superset is solved, the first system is said to *subsume* the second system. The union of problems solved by the non-subsumed systems define the *state-of-the-art* - all the problems that are solved by any system. Any problems that are solved by all non-subsumed systems are considered to be *easy*, and any that are solved by just some of the non-subsumed systems are considered *difficult*. (In the TPTP, the fraction of non-subsumed class representatives that fail on a given problem, is the difficulty rating for that problem.) The fraction of the difficult unbiased problems that a given system solves, is the rating for that system. Note that a system rating of 1 indicates that the systems in that system's equivalence class subsume all other systems, and a system rating of 0 indicates that the system solves only easy problems.

Given an arbitrary problem it is then possible to recommend systems that seem more likely to succeed. Firstly, the SPC of the problem is established. Then, for that SPC, the representatives of classes of non-subsumed systems are recommended in order of system rating. Any systems that have a rating of 0 are excluded. SSCPA uses these system recommendations to decide which systems to run in parallel. (Possible improvements to this recommendation scheme are proposed in the conclusion.)

SSCPA Modes

There are several decisions that need to be made for a competition parallelism ATP system:

- Which ATP systems should be used?
- How much of the CPU time limit should be allocated to each system?
- What CPU priority should be given to each system?
- When should each system be started?

SSCPA defines three basic modes of operation. The *naive* mode runs all available systems in parallel, giving equal CPU allocations, equal CPU priorities, and starting all systems at the same time. The *smart selective* mode runs some of the recommended ATP systems in parallel, giving unequal CPU allocations, not necessarily equal CPU priorities, and not necessarily starting the systems at the same time. Details of the smart selective mode are given below. The *naive selective* mode runs the same ATP systems as the smart selective mode, giving equal CPU allocations, equal CPU priorities, and starting all systems at the same time.

The number of recommended ATP systems used in SSCPA's smart selective mode is determined by the CPU time limit and a *minimal time allocation*. The CPU time limit is repeatedly divided by two, and the result allocated to the next recommended ATP system. This continues until the allocation would be below the minimal time allocation value, at which stage all the remaining time is allocated to the next ATP system. For example, with a CPU time limit of 300 seconds and a minimal time allocation of 30 seconds, four systems would be used. The allocated times would be 150 seconds, 75 seconds, 37.5 seconds, and 37.5 seconds. If insufficient ATP systems are recommended the allocation process terminates early, with all the remaining time being allocated to the last recommended system. It is important to make the minimal time allocation large enough, such that the individual ATP systems are able to solve most of the problems that they can within the CPU time limit. Figure 2 suggests that about 50 seconds is appropriate.

There are three sub-modes to the smart selective mode. These specify when each of the ATP systems is started and what CPU priority is given to each ATP system. The *eager* smart selective mode starts all systems together, with equal priority. The systems with smaller time allocations progressively leave if they do not find a solution, leaving the CPU to the systems with larger allocations. This mode should work well if the ATP systems are equally likely to find a solution quickly, but the higher rated systems are more likely to find a proof after a longer time. The *fair* smart selective mode starts all systems together, with CPU priorities arranged so that the systems get time on the CPU in proportion to their allocations (so that if no system finds a solution, they all end together). This mode should work well if the time allocations reflect the likelihood of each system finding a solution. The *reluctant* smart selective mode starts the system with the largest allocation first, and then when half of its allocation is used, starts the next system, and so on (so that if no system finds a solution, they all end together). The reluctant mode should work well if the higher rated systems are more likely to find a solution quickly than the lower rated systems. Table 1 summarizes the SSCPA modes.

	Naive	Naive	Smart selective		
		selective	Eager	Fair	Reluct't
Systems	All	Recommended systems			
CPU allocat'n	CPU limit / # of systems		Based on rating		
CPU priority	Equal			Based on CPU all'n	Equal
Start	Together				Staggered

Table 1: SSCPA Modes

The SSCPA Implementation

The current SSCPA implementation uses the following ATP systems:

- Bliksem 1.01, the current release of Bliksem. Bliksem is available from `http://www.cwi.nl/~nivelle`.
- Gandalf c-1.8c, the release of Gandalf used in CASC-15. Gandalf is available from `http://www.cs.chalmers.se/~tammet/gandalf/`.
- OtterMACE 437, a combination of Otter 3.0.5 and MACE 1.3.2. Both are available from `http://www.mcs.anl.gov/home/mccune/ar/otter/`.
- SETHEO C-14, the release of SETHEO used in CASC-14. SETHEO is available from `http://wwwjessen.informatik.tu-muenchen.de/~setheo/`.
- SPASS 0.94. SPASS is available from `http://www.mpi-sb.mpg.de/guide/software/spass.html`.

These systems have been combined using `perl` scripts that extract system recommendations from the performance data and control the parallel execution of the systems. The SSCPA implementation accepts problems in TPTP format, and uses the `tptp2X` utility to convert the problem to the formats expected by the ATP systems. In practice this conversion often takes significantly more CPU time than to find a solution. This overhead could be greatly reduced, as `tptp2X` is written in Prolog and is not optimised. A WWW interface to some SSCPA modes is publicly available (Sutcliffe 1998).

Results

In order to evaluate SSCPA, it has not been necessary to physically run the implemented SSCPA. Rather, the collected performance data for the component systems has been used to compute SSCPA results. The computed results slightly underestimate the true capabilities of SSCPA, for two reasons. Firstly, the computation uses results from some slightly earlier system versions than used in the SSCPA implementation. The computation uses results from Bliksem 1.01, Gandalf c-1.0d, OtterMACE 437, SETHEO C-14, and SPASS 0.85. Secondly, the collected performance data was obtained on a range of hardware. By running SSCPA on the best of the hardware platforms (such porting appears to be easily possible), better results could be obtained. However, as was noted earlier, linear increases in the computational resources do not lead to the solution of significantly more problems.

SSCPA results have been computed for all 3622 problems in TPTP v2.1.0, with a 300 second time limit. The use of a 300 second time limit was the limit used in the CADE ATP system competitions, and seems to be accepted as reasonable by the ATP community. Table 2 shows the results for the individual systems. Table 3 shows the SSCPA results with minimal time allocations of 60 seconds (SSCPA uses up to three systems), 30 seconds (SSCPA uses up to four systems), and 15 seconds (SSCPA uses up to five systems). The 'Solved' columns show how many problems each system solves, and the 'Time' columns show the average CPU time taken over problems solved.

All the SSCPA modes out-perform the individual systems in terms of number of problems solved. This immediately justifies the use of parallelism. The slight improvement of the naive selective mode over the naive mode shows the benefit of using the recommended ATP systems. The improvement of the smart selective modes over the naive selective mode shows the benefit of allocating CPU time relative to the systems' ratings.

System	Solved	Time
Bliksem 1.01	1634	11.1
Gandalf c-1.0d	1551	15.6
OtterMACE 437	1892	12.4
SETHEO C-14	1220	18.0
SPASS 0.85	2054	16.6

Table 2: Individual System Results for TPTP v2.1.0

	60 seconds		30 seconds		15 seconds	
SSCPA	**Solved**	**Time**	**Solved**	**Time**	**Solved**	**Time**
Naive	2266	19.4	2301	17.6	2297	17.2
Naive sel.	2284	18.1	2304	16.8	2297	17.2
Eager	2309	20.6	2328	18.9	2324	18.9
Fair	2309	29.5	2328	17.8	2324	17.8
Reluctant	2309	24.1	2328	22.6	2324	19.2

Table 3: SSCPA Results for TPTP v2.1.0

Running SSCPA with a 240 second minimal time allocation, so that only the highest recommended ATP system is used, solves just 2145 problems with an average solution time of 13.4 seconds. This again shows the benefits of parallelism, but also shows the benefit of using a recommended ATP system.

All the SSCPA modes are slower than the individual systems, as would be expected. However, all the average CPU times are very low compared to the CPU time limit. Over the entire TPTP the average times taken do not vary much across the smart selective modes. The eager mode gives the most consistent performance, suggesting that the ATP systems are equally likely to find a solution quickly. This corresponds to the very short solutions times that all the systems achieve for many problems, as shown in Figure 2. Further discussion and analysis of times taken is given in (Seyfang 1998).

The results show an improved performance by SSCPA when four systems are used (30 second minimal time allocation) instead of three (60 second minimal time allocation), but no further improvement when five systems are used (15 second minimal time allocation). This is due to the very short time allocated to the fourth and fifth systems when they are recommended, and shows the importance of the minimal time allocation parameter.

In some SPCs less than the maximal number of systems are recommended, for all minimal time allocation values. In SPCs for FOF problems, Gandalf and SETHEO are never recommended, as their input format does not permit FOF. In some SPCs only one system is recommended, resulting in that system being allocated all 300 seconds.

SSCPA has also been evaluated on the 475 problems that were eligible for selection in the MIX division of CASC-15 (Sutcliffe and Suttner 1998c), using a 300 second time limit and a minimal time allocation of 60 seconds. The results are shown in Table 4.

System	Solved	Time
Bliksem 1.01	273	13.6
Gandalf c-1.0d	335	28.2
OtterMACE 437	292	20.2
SETHEO C-14	208	23.8
SPASS 0.85	367	39.4
Naive	418	39.7
Naive selective	433	37.9
Eager	442	43.3
Fair	442	39.5
Reluctant	442	44.6

Table 4: SSCPA Results for the CASC-15 MIX Problems

Conclusion

This paper has described SSCPA, an uncooperative multiple calculus competition parallelism ATP system, that multitasks on a single CPU. SSCPA runs multiple sequential ATP systems in parallel. This approach is motivated by the observation that no individual ATP system performs well on all problems. SSCPA uses performance data from the ATP systems to select those that are best suited to the problem, to run in parallel.

It is clear that SSCPA is a meta-system, and is reliant on the capabilities of the sequential ATP systems it employs. As a result, SSCPA is easily upgraded as new ATP systems, or new versions of ATP systems, become available. All that is required it to run the new system over the TPTP, collect the performance data, and make the new system available to SSCPA. Thus it is expected that SSCPA will always be able to out perform the available sequential ATP systems.

Currently SSCPA recommends the arbitarily chosen representatives from equivalence classes of unsubsumed systems. This could immediately be improved by choosing the representative based average solution time. Further, the motivation for recommending only the equivalence class representatives is that members of an equivalence class have very similar capabilities with respect to TPTP problems. For arbitrary (non-TPTP) problems it may be the case that using more equivalence class members would improve performance. This would be particularly appropriate if the number recommended equivalence class representatives is less than the number of systems that SSCPA could use. This is an issue for further investigation.

Acknowledgements

Serious thanks go to the developers of the sequential ATP systems, for making their systems and performance data available.

References

Astrachan O.L. (1994), METEOR: Exploring Model Elimination Theorem Proving, *Journal of Automated Reasoning* 13(3), pp.283-296.

Cook D., Varnell R. (1998), Adaptive Parallel Iterative Deepening Search, *Journal of Artificial Intelligence Research* 9, pp.139-166.

De Nivelle H. (1998), Bliksem 1.00, Sutcliffe G., Suttner C., *Proceedings of the CADE-15 ATP System Competition* (Lindau, Germany), pp.9.

Fuchs M. (1997), Automatic Selection of Search-guiding Heuristics, Dankel D., *Proceedings of the 10th Florida Artificial Intelligence Research Symposium* (Daytona Beach, USA), pp.1-5,

Florida AI Research Society.Fuchs M., Wolf A. (1998), System Description: Cooperation in Model Elimination: CPTHEO, Kirchner C., Kirchner H., *Proceedings of the 15th International Conference on Automated Deduction* (Lindau, Germany), pp.42-46, Lecture Notes in Artificial Intelligence 1421, Springer-Verlag.

Letz R., Schumann J., Bayerl S., Bibel W. (1992), SETHEO: A High-Performance Theorem Prover, *Journal of Automated Reasoning* 8(2), pp.183-212.

Lusk E.L., McCune W.W. (1992), Experiments with ROO, a Parallel Automated Deduction System, Fronhofer B., Wrightson G., *Parallelization in Inference Systems*, pp.139-162.

McCune W.W. (1994), Otter 3.0 Reference Manual and Guide, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, USA.

McCune W.W. (1997), Solution of the Robbins Problem, *Journal of Automated Reasoning* 19(3), pp.263-276.

Seyfang D. (1998), Smart Switched Automated Theorem Proving System, Honours Thesis, Department of Computer Science, James Cook University, Townsville, Australia.

Sutcliffe G., Suttner C.B. (1997), Special Issue: The CADE-13 ATP System Competition, *Journal of Automated Reasoning* 18(2).

Sutcliffe G., Suttner C.B. (1998a), The TPTP Problem Library: CNF Release v1.2.1, *Journal of Automated Reasoning* 21(2), pp.177-203.

Sutcliffe G., Suttner C.B. (1998b), ATP System Results for the TPTP Problem Library, `http://www.cs.jcu.edu.au/~tptp/TPTP/Results.html`.

Sutcliffe G., Suttner C.B. (1998c), The CADE-15 ATP System Competition, *Journal of Automated Reasoning*, To appear.

Sutcliffe G. (1998), System on TPTP, `http://www.cs.jcu.edu.au/cgi-bin/tptp/SystemOnTPTPFormMaker`.

Suttner C.B., Schumann J. (1994), Parallel Automated Theorem Proving, Kanal L., Kumar V., Kitano H., Suttner C., *Parallel Processing for Artificial Intelligence 1*, pp.209-257, Elsevier Science.

Suttner C.B., Sutcliffe G. (1998), The CADE-14 ATP System Competition, *Journal of Automated Reasoning* 21(1), pp.99-134.

Tammet T. (1997), Gandalf, *Journal of Automated Reasoning* 18(2), pp.199-204.

Weidenbach C. (1997), SPASS: Version 0.49, *Journal of Automated Reasoning* 18(2), pp.247-252.

Wolf A., Letz R. (1998), Strategy Parallelism in Automated Theorem Proving, Cook D., *Proceedings of the 11th Florida Artificial Intelligence Research Symposium* (Sanibel Island, USA), pp.1-5.