Performance Analysis of Parallel Constraint-Based Local Search (Extended Abstract)

Salvador Abreu[4] Yves Caniou[1] Philippe Codognet[2]
Daniel Diaz[3] Florian Richoux[2]

[1] JFLI, CNRS / NII, Japan
yves.caniou@ens-lyon.fr
[2] JFLI, CNRS / UPMC / University of Tokyo, Japan
{codognet}{richoux}@jfli.itc.u-tokyo.ac.jp
[3] University of Paris 1-Sorbonne, France
daniel.diaz@univ-paris1.fr
[4] Universidade de Évora and CENTRIA FCT/UNL, Portugal
spa@di.uevora.pt

Abstract. We present a parallel implementation of a constraint-based local search algorithm and investigate its performance results on hardware with several hundreds of processors. We choose as basic constraint solving algorithm for these experiments the "adaptive search" method, an efficient sequential local search method for Constraint Satisfaction Problems. The implemented algorithm is a parallel version of adaptive search in a multiple independent-walk manner, that is, each process is an independent search engine and there is no communication between the simultaneous computations. Preliminary performance evaluation are very encouraging. On a variety of classical CSPs benchmarks from CSPLIB, speedups are very good for a few tens of cores, and good up to a few hundreds of processors. More challenging problems derived from real-life applications (Costas array) shows even better speedups, nearly optimal up to 256 cores.

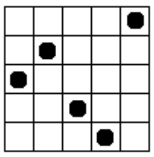
1 Introduction

In this short note we want to briefly address the issue of parallelizing constraint solvers for massively parallel architectures. A design principle implied by this goal is to abandon the classical model of propagation-based solver using shared data structures which have been developed for shared-memory architectures (e.g. multi-cores) or tightly controlled master-slave communication in cluster-based architectures and to consider either purely independent parallelism or very limited communication between parallel processes. One candidate for this purpose is the family of local search methods, especially when considering independent multiple-walks. Indeed, during the last decade, the family of Local Search methods and Metaheuristics has been quite successful in solving large real-life problems. Applying Local Search to Constraint Satisfaction Problems (CSP) has

also been attracting some interest as it can tackle CSPs instances far beyond the reach of classical propagation-based solvers.

A generic domain-independent Local Search method named Adaptive Search was proposed in [2,3]. It is a metaheuristic that takes advantage of the structure of the problem to guide the search and that can be applied to a large class of constraints (*e.g.,* linear and non-linear arithmetic constraints, symbolic constraints). Moreover, it intrinsically copes with over-constrained problems. Adaptive Search is a simple algorithm but it turns out to be quite efficient in practice, see [1] for a comparison with the Comet 2.1.1 system on a few classical benchmarks.

We performed our experiments with independent multiple-walks of the Adaptive Search constraint solver (i.e. launching in parallel several search engines starting from different initial configurations and performing the computation in a purely independent manner) and benchmarked it with classical CSP benchmarks from the CSPLIB and with a very difficult combinatorial problem, the Costas Array Problem (CAP).



Historically, Costas arrays have been developed in the 1960's to compute a set of sonar and radar frequencies avoiding noise. A Costas array is an $n \times n$ grid containing n marks such that there is exactly one mark per row and per column and the $n(n-1)/2$ vectors between the marks are all different. We give here an example of Costas array of size 5. It is convenient to see the Costas Array Problem (CAP) as a permutation problem by considering an array of n variables $(V_1, \ldots, V_n)$ which forms a permutation of $\{1, 2, \ldots, n\}$. The Costas array above can thus be represented by the array $[3, 4, 2, 1, 5]$.

We performed our experiments on two different hardware systems:

- the Hitachi HA8000 supercomputer of the University of Tokyo with a total number of 15232 cores. This machine is composed of 952 nodes, each of which is composed of 4 AMD Opteron 8356 (Quad core, 2.3 GHz) with 32 GB of memory. Users can only have a maximum of 64 nodes (1,024 cores) in normal service and we used up to 256 cores in our experiments.
- the Grid'5000 infrastructure, the French national Grid for the research, which contains a maximum of 5934 cores deployed on 9 sites distributed in France. We used two subsets of the computing resources of the Sophia-Antipolis node: Suno, composed of 45 Dell PowerEdge R410 with 8 cores each, thus a total of 360 cores, and Helios, composed of 56 Sun Fire X4100 with 4 cores each, thus a total of 224 cores.

2 Parallel Performance Analysis

We used the implementation of the Adaptive Search method consisting of a C-based framework library available as freeware at the URL:
`http://cri-dist.univ-paris1.fr/diaz/adaptive/`

In addition of the Costas array problem, we use a series of classical benchmarks from CSPLIB consisting of:

- `all-interval`: the All Interval Series problem (prob007 in CSPLib),
- `perfect-square`: the Perfect Square placement problem (prob009 in CSPLib),
- `magic-square`: the Magic Square problem (prob019 in CSPLib).

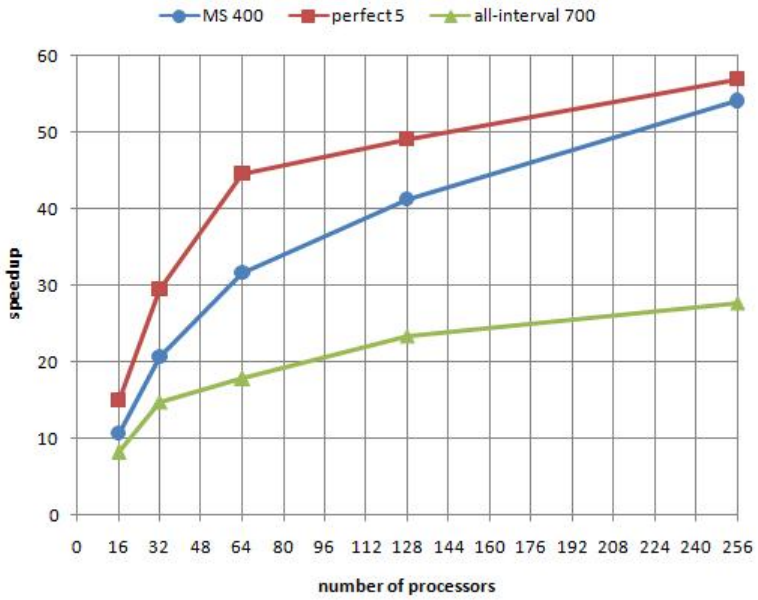


Fig. 1. speedups on HA8000

We can see in Figures 1 and 2 from [1] that the speedups are more or less equivalent on the HA8000 machine and on the Grid'5000 platform. As the speedups on the two Grid'5000 platforms (Helios and Suno nodes) are nearly identical, we only depicted the speedups with Suno, as we can experiment up to 256 cores on this platform. Only in the case of `perfect-square` are the results significantly different between the two platforms, for 128 and 256 cores. In those cases Grid'5000 has much better speedups than on HA8000. Maybe this is because execution time is getting too small (less than one second) and therefore some other mechanisms interfere. The stabilization point is not yet obtained for 256 cores, even if speedups do not increase as fast as the number of cores, *i.e.,* are getting further away from linear speedup.

Concerning the CAP, let us first note that finding big instances of Costas arrays, such as $n = 22$, takes many hours in sequential computation, and we will thus limit our experiments on all machines to executions on 32 cores and above. We can observe that on all platforms, execution times are halved when

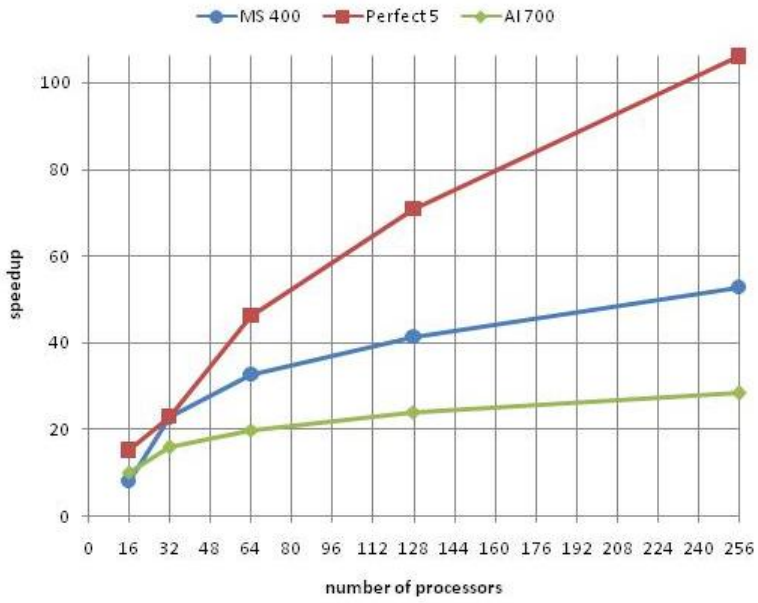


Fig. 2. speedups on Grid5000 (Suno)

the number of cores is doubled, thus achieving ideal speedup. This is graphically depicted in Figure 3 from [4] on a log-log scale. As a final result, we note that we can now solve $n = 22$ in about one minute on average with 256 cores on HA8000.

3 Conclusion and Future Work

We presented performances of a parallel implementation of a constraint-based local search algorithm, the "Adaptive Search" method in a multiple independent-walk manner. Each process is an independent search engine and there is no communication between the simultaneous computations except for completion. Performance evaluation on a variety of classical CSPs benchmarks over two different parallel architectures (a supercomputer and a Grid platform) shows that the method is achieving speedups of about 30 with 64 cores, 40 with 128 cores and more than 50 with 256 cores, and presents linear speedups on the Costas Array Problem. Of course speedups depend on the benchmarks and the bigger the benchmark, the better the speedup.

Current work focuses on more complex parallel methods with inter-processes communication, i.e., in the dependent multiple-walk scheme, in order to further improve performance. The communication mechanism will be designed with the goals of (1) minimizing data transfers as much as possible, as we aim at massively parallel machines with no hierarchical memory, (2) re-using some common

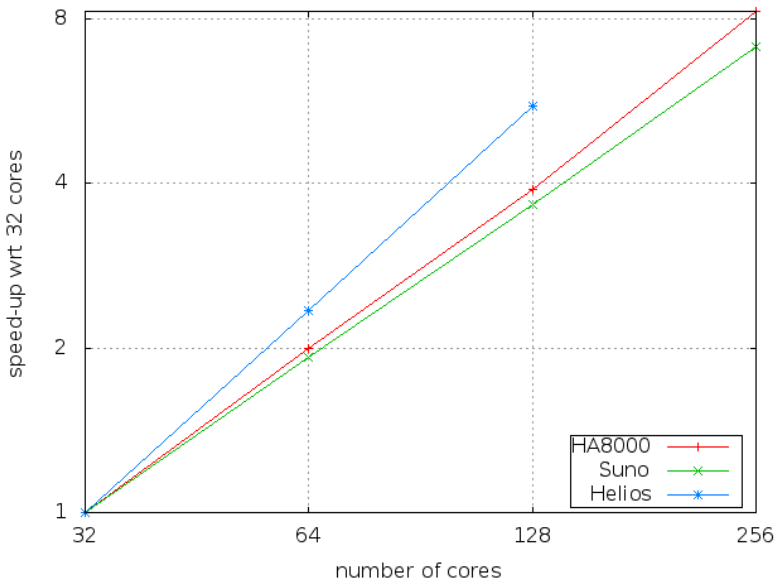


Fig. 3. Speedups for CAP 22 w.r.t. 32 cores

computations and/or recording previous interesting crossroads in the resolution, from which a restart can be operated. However, it is a challenge to design a scheme that could outperform the independent multiple-walk parallelization. One issue is that the global cost of a configuration is not a reliable information since given by heuristic error functions.

References

1. Y. Caniou, P. Codognet, D. Diaz, and S. Abreu. Experiments in parallel constraint-based local search. In *EvoCOP'11, 11th European Conference on Evolutionary Computation in Combinatorial Optimisation*, Lecture Notes in Computer Science, Torino, Italy, 2011. Springer Verlag.
2. P. Codognet and D. Diaz. Yet another local search method for constraint solving. In *proceedings of SAGA'01*, pages 73–90. Springer Verlag, 2001.
3. P. Codognet and D. Diaz. An efficient library for solving CSP with local search. In T. Ibaraki, editor, *MIC'03, 5th International Conference on Metaheuristics*, 2003.
4. D. Diaz, F. Richoux, P. Codognet, Y. Caniou, and S. Abreu. Constraint-based local search for costas array problem, draft.