The Promise of Parallelism for AI

James Hendler
Computer Science Dept.
University of Maryland

To those of us playing with parallel models of AI, the question is often raised as to whether the parallelism offers new ways of doing things, or just lets us do the old ones fast. The implication is that if the latter, the work is just "engineering hacks" and not "scientifically interesting." We won't mention where this question typically arises from[1], but we do know that answering it is sometimes difficult.

In this talk, I will attempt to face this question head on. In short, my suggestion is that the answer to "Whether parallelism adds something new OR is it just faster" is "YES." That is, that these two are not crucially different, and that parallelism both lets us explore new approaches (for example Waltz' work in MBR) and to do traditional techniques faster. However, and more importantly, I will argue that the latter (faster) will provide some of the former (new).

My contention is simple. Certain well known kinds of inferences are "computationally ineffective" in serial systems. Examples include top-down inferencing and recognition queries in semantic networks, and certain approaches to case-based memory retrieval. In these cases, serial systems often go to great lengths to avoid such inferences, and a large amount of the work in the area ends up surrounding an "ephiphenomenal" problem – one that emerges from implementation details, not from deeper rationale. Thus whole cottage industries are created in developing new techniques, none of which are really necessary or interesting given parallel tools for overcoming the inefficiency.

In defense of this position, I will explore one example - the use of parallelism in support of a case-based planning system. I will show that SPMD parallelism provides an alternative to the use of highly structured memories in CBR. Using a massively parallel frame-based AI language (Parka), our system is able to do extremely fast retrieval of complex cases from a large, unindexed memory. Since the retrieval makes use of the massive parallelism inherent in Parka, the system has many advantages: indexing is not necessary, very large case-bases can be used, memory can be probed in numerous alternate ways, and queries can be made at several levels, allowing more specific retrieval and adaptation. I will describe experiments with our case-based planning system, CaPER, and show that it can rapidly produce efficient plans, even with a very large case memory. Thus, the system using the parallel memory can *effectively* solve problems that serial implementations cannot.

[1] although the "term subsumption" community might want to feel uncomfortable right about here.

Massively distributed constraint satisfaction

Walter Hower and Stephan Jacobi
Institut für Informatik, Fachbereich 4, Universität Koblenz-Landau
Rheinau 3–4, D–W-5400 Koblenz, F.R.G.
walter@uni-koblenz.de

Abstract

The present work would like to illustrate a distributed ("MIMD") approach solving the *constraint satisfaction problem* by massive parallelism.

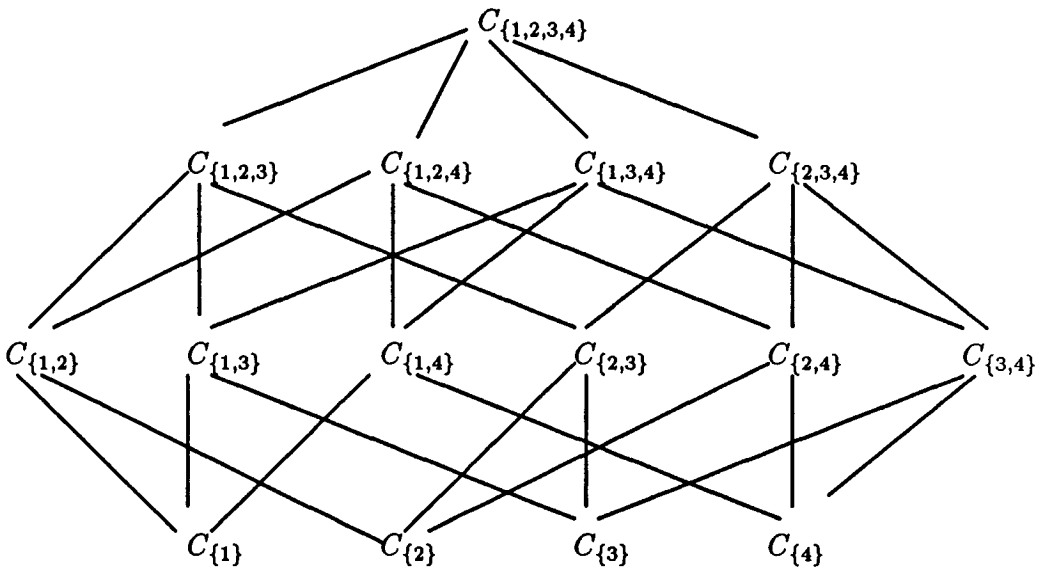
0 Introduction

To cope with ever increasing knowledge and its maintenance is one of the real challenges of the future. One idea approaching this problem is the identification and exploitation of parallelism. The situation is not hopeless; there already exist various architectures of parallel hardware, and also parallel software is available. The field of Artificial Intelligence (see [RK91], e.g.) normally has to deal with really hard problems where a large amount of knowledge must be processed. So, it is a predestined area regarding the deployment of parallel architectures. One specific problem is the topic of the current paper: the *constraint satisfaction problem* (=: CSP).
The following section briefly introduces this subject (cf. also [Dec92]) — along with the identification of sources for a parallelization. Section 2 considers some related work. In section 3 we present our approach realizing a distributed representation. Final remarks conclude the present paper.

1 The CSP

The problem deals with the assignment of values to variables according to existing "constraints". Given n variables with their finite domains along with the admissible combinations (=: constraints) of their values it is the goal to explicate all instantiations still possible resulting in a set of n-tuples which represents the globally consistent solution.

The following figure symbolizes the structure of a CSP with four variables:



The index set of one constraint enumerates the specific variable(s) involved in the constraint declaration.
For instance: $C_{\{1,2,4\}} := \{(0,1,9),(1,9,0),(3,5,7),(2,4,8)\}$ indicates that just these four 3-tuples are allowed concerning the three variables V_1, V_2, V_4. (Assuming that also the domains of the variables (the values — 1-tuples) are given, and maybe also some other constraints (of arbitrary arity (≤ 4)), the problem would be to compute $C_{\{1,2,3,4\}}$, the set of all admissible 4-tuples obeying all the prevalent constraints.)
The representation figured above may illustrate the exponential complexity of the CSP: computing all $\binom{n}{i}$ i-ary constraints implies a complexity of $O(2^n)$. (Currently, [How92a] seems to reflect one of the best algorithms for *general* global constraint satisfaction.)
The bad (serial) time behaviour (which is by the nature of the problem) of the CSP has motivated the idea to try to parallelize the computation — due to the huge inherent parallelism (cf. also [How91]).
Initially, we recognize two sources for a parallelization: first, we could in parallel process along the edges of a specific vertex, and second, we could in parallel consider the vertices (of a specific stage).

2 Related work

[HM90] employs a parallel ATMS model to realize a parallelization. [KD91] discusses "local consistency" considering "2-consistency" (also called "arc consistency" — the kind of consistency just dealing with unary and binary constraints) and briefly "3-consistency" (also called "path consistency" — dealing with at most ternary constraints). [YIK91] does the same; there, the number of "agents" is less than or equal to the number of the variables. [ZM91] presents a parallel shared-memory approach as well as a distributed representation achieving arc consistency. [CDK91] just deals with binary constraints. [CBB91] reports on experiments with parallelism concerning arc consistency. [HH91] may point to a connectionistic approach; although it especially deals with local (arc) consistency, its neural network architecture ensures global consistency. [LHB92] proposes a distributed solution considering binary constraints. [CS92] exploits massive parallelism in order to obtain arc consistency.

3 Our Distributed Approach

The (linear parallel time) algorithm has been published in [How92b]; the work reported here actually exploits its most rigorous realization in a distributed architecture. (For instance, it really performs a pre-processing (descending the lattice figured above in a specific way) prior to the ascending phase needed by the algorithm to compute the globally consistent solution.) Roughly speaking, the algorithm works as follows: Among the constraints given in an individual CSP it takes the ones with the highest arity, i.e., it first considers those restrictions which concern the largest number of variables. Then it propagates the relevant tuples to those lower-ary constraints which are influenced directly by these specific (higher-ary) constraints. In order to exemplify the interrelations please recall the figure presented earlier. There, the constraint $C_{\{1,2,4\}}$ directly influences the constraints $C_{\{1,2\}}$, $C_{\{1,4\}}$, and $C_{\{2,4\}}$. (The 3-tuples (of $C_{\{1,2,4\}}$) would get projected (in three different ways — depending on the (three) target constraint sets) to according 2-tuples belonging to the appropriate variables.) After having reached the bottom of the lattice an upward phase has to follow; ascending towards the top node finally computes the globally consistent solution ("n-consistency") after the arrival at $C_{\{1,\ldots,n\}}$.

Please note that each constraint has just to wait for the completion of the parallel propagation along its incoming links; the computation of constraints which do not directly depend on each other — for instance the ones with the same arity (placed at a common stage in the lattice) — is performed in parallel, too.

The architecture we use in our approach is a distributed one realized by T800 transputers via message passing (MIMD philosophy) subject to a "master-worker" design. We would like to propose a concept consisting of several masters each with several workers. Let t be the number of "useful" transputers (see below) we employ about $t/5$ "master" and $4t/5$ "worker" transputers in order to keep small the communication time. (Each transputer just supports four links.) The exact number of transputers useful for a good speed-up actually depends on the individual problem; however, the upper bound is clearly determined by the so-called "maximum number of effective processors", here $\binom{n}{\lceil n/2 \rceil}$ — cf. [How92b]. (Please note that in [Swa88] an exponential number of processors is employed to compute (in constant time) binary ("local") consistency.) Also, the transputer network topology depends on the application. Internally, a master (transputer) is organized asynchronously. So-called "threads" perform the communication routines; for each worker (transputer) such a thread is established. No synchronization is needed among the different workers, and optimal load balancing is possible. The master maintains the status of the constraints, for instance, whether a constraint has just been pre-processed (via a downward step) or already been computed completely (after an upward step). Additionally, it supplies the constraints to the workers which themselves (re)compute a single constraint (depending on the ones directly connected to).

Unfortunately, we were just able to use one master during our experimentation. However, linear speed-up could be obtained when considering just the computation time of the different workers. (When we have only one master some communication overhead is measurable.)

In one of our experiments we model a crossing with eight traffic lights — four for the vehicles and four for the pedestrians; formulating the constraints results in four 4-ary constraints. The traffic lights for the vehicles have four possible values, and the ones for the pedestrians have two different values. We are looking for all admissible colour assignments to the eight traffic lights according to the constraints — i.e., the set of admissible 8-tuples. (Please note that out of the 2^{12} possibilities concerning the domain values just four combinations (2^2) actually form the globally consistent solution.)

Our realization really produces an optimal efficiency (speed-up devided by the number of processors) of about 1.

4 Final Remarks

Due to existing limitations our realization should just be considered as an initial attempt; future architectures (cf. [KHH+91]) would exploit more thoroughly the large inherent parallelism of the distributed algorithm underlying the work reported here.
Currently, we investigate the incorporation of the ideas into a CAD system where the intelligent and quick processing of constraints is indispensable.

Acknowledgement
Many thanks to Manfred Rosendahl for his helpful support.

References

[CBB91] James M. Conrad, Dennis Bahler, and James Bowen. Static parallel arc consistency in constraint satisfaction. In Zbigniew W. Ras and Maria Zemankova, editors, *Methodologies for Intelligent Systems, 6th International Symposium, ISMIS '91*, pages 500–509, Charlotte, NC, U.S.A., October 16–19, 1991. Proceedings, Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, Volume 542, Springer-Verlag, Berlin/Heidelberg.

[CDK91] Zeev Collin, Rina Dechter, and Shmuel Katz. On the Feasibility of Distributed Constraint Satisfaction. In John Mylopoulos and Ray Reiter, editors, *12th International Joint Conference on Artificial Intelligence*, pages 318–324, Darling Harbour, Sydney, Australia, 24 – 30 August 1991. IJCAII, Proceedings, Volume 1; distributed by Morgan Kaufmann Publishers, San Mateo, California, USA.

[CS92] Paul R. Cooper and Michael J. Swain. Arc consistency: parallelism and domain dependence. *Artificial Intelligence*, 58(1–3):207–235, December 1992. Elsevier Science Publishers B.V., Amsterdam, The Netherlands.

[Dec92] Rina Dechter. Constraint Networks. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276–285. John Wiley & Sons, 1992. Volume 1, second edition.

[HH91] Steffen Hölldobler and Walter Hower. Constraint Satisfaction in a Connectionist Inference System. In Francisco J. Cantú-Ortiz and Hugo Terashima-Marín, editors, *IV International Symposium on Artificial Intelligence*, pages 215–221, Cancún, November 13–15, 1991. Proceedings, Editorial Limusa, S.A. de C.V., México.

[HM90] Taku Harada and Fumio Mizoguchi. Parallel Constraint Satisfaction by Paralleling ATMS. In Hozumi Tanaka, editor, *Artificial Intelligence in the Pacific RIM*, pages 462–467. IOS Press, Amsterdam, The Netherlands, 1990. PRICAI '90, proceedings.

[How91] Walter Hower. Parallel global constraint satisfaction. In *Informal Proceedings of the IJCAI-91 Workshop on Parallel Processing for Artificial Intelligence*, pages 80–85, PPAI-91, Darling Harbour, Sydney, New South Wales, Australia, August 24/25, 1991.

[How92a] Walter Hower. A general framework for global constraint satisfaction. (accepted for) FGCS'92 Workshop on Constraint Logic Programming, Institute for New Generation Computer Technology, Tokyo, Japan, June 6/7, 1992.

[How92b] Walter Hower. Constraint satisfaction via partially parallel propagation steps. In Bertram Fronhöfer and Graham Wrightson, editors, *Parallelization in Inference Systems*, pages 234–242. Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, Volume 590, Springer-Verlag, Berlin/Heidelberg, 1992.

[KD91] Simon Kasif and Arthur L. Delcher. Analysis of Local Consistency in Parallel Constraint Satisfaction Networks. In *1991*

Spring Symposium "Constraint-Based Reasoning", pages 154–163, Stanford University, CA, U.S.A., March 26–28, 1991. AAAI. Working Notes.

[KHH+91] Hiroaki Kitano, James Hendler, Tetsuya Higuchi, Dan Moldovan, and David Waltz. Massively Parallel Artificial Intelligence. In John Mylopoulos and Ray Reiter, editors, *12th International Joint Conference on Artificial Intelligence*, pages 557–562, Darling Harbour, Sydney, Australia, 24 - 30 August 1991. IJCAII, Proceedings, Volume 1; distributed by Morgan Kaufmann Publishers, San Mateo, California, USA.

[LHB92] Q. Y. Luo, P. G. Hendry, and J. T. Buchanan. A New Algorithm for Dynamic Distributed Constraint Satisfaction Problems. In *Proceedings of the 5th Florida AI Research Symposium*, pages 52–56, 1992. FLAIRS '92, Ft. Lauderdale, Florida, USA.

[RK91] Elaine Rich and Kevin Knight. *Artificial Intelligence.* McGraw-Hill, second edition, 1991.

[Swa88] Michael J. Swain. Comments on Samal and Henderson: "Parallel Consistent Labeling Algorithms". *International Journal of Parallel Programming*, 17(6):523–528, 1988. Plenum Press, New York / London.

[YIK91] Makoto Yokoo, Toru Ishida, and Kazuhiro Kuwabara. Distributed Constraint Satisfaction for DAI Problems. In *1991 Spring Symposium "Constraint-Based Reasoning"*, pages 191–199, Stanford University, CA, U.S.A., March 26–28, 1991. AAAI. Working Notes.

[ZM91] Ying Zhang and Alan K. Mackworth. Parallel and Distributed Algorithms for Constraint Networks. Technical Report 91-6, Department of Computer Science, The University of British Columbia, Vancouver, B.C., Canada, May 1991. cf. also: *Parallel and Distributed Constraint Satisfaction*, in: Informal Proceedings of the IJCAI-91 Workshop on Parallel Processing for Artificial Intelligence, pages 229–234, PPAI-91, Darling Harbour, Sydney, New South Wales, Australia, August 24/25.