Applying interchangeability techniques to the distributed breakout algorithm

Adrian Petcu and Boi Faltings
{adrian.petcu, boi.faltings}@epfl.ch
Artificial Intelligence Laboratory
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne, Switzerland

Abstract

This paper presents two methods for improving the performance of the Distributed Breakout Algorithm using the notion of interchangeability. In particular, we use neighborhood *partial* and *full* interchangeability techniques to keep conflicts localized and avoid spreading them to neighboring areas.

Our experiments on distributed sensor networks show that such techniques can significantly reduce the number of cycles required to solve the problems (therefore also reduce communication and time requirements), especially on difficult problems. Moreover, the improved algorithms are able to solve a higher proportion of the test problems.

1 Introduction

Distributed Constraint Satisfaction (DisCSP) is a powerful paradigm applicable for a wide range of coordination and problem solving tasks in distributed artificial intelligence.

Among the distributed algorithms that were developed for this kind of problems ([Yokoo *et al.,* 1998]), the Distributed Breakout Algorithm (DBA) received quite some interest (e.g. [Zhang *et al.*, 2002]) because of a number of properties that this algorithm exhibits (simple, efficient, low overhead, linear memory requirements, good anytime characteristics).

DBA is an extension of the original centralized Breakout Algorithm ([Morris, 1993]). This algorithm is a local search method, with an innovative technique for escaping from local minima: the constraints have weights, which are dynamically increased to force the agents to adjust their values while in a local minimum. During the execution of the algorithm, each agent proposes improvements to the current state by changing it's variable value such that the cost of violated constraints is decreased as much as possible.

While having the interesting properties that we enumerated above, local search algorithms also have a common problem: choosing indiscriminately between the possible values of the local variable (only considering the cost of the immediate constraint violations) can lead to "chain-reactions" (one conflict originating in one part of the constraint graph needlessly propagates throughout the whole graph, only to (hopefully) be resolved in a completely different part of the graph).

We analyzed these phenomena, and drew the conclusion that using interchangeability techniques, one can determine what values from the local domain will not cause such conflict propagations, and use one of those values as the next variable assignment. In this way, we look for a "local resolution" to all conflicts, in the sense that we keep them contained as much as possible, and only involve "external parties" when there is no other way.

We discovered that techniques based on interchangeability [Freuder , 1991] (both *neighborhood partial* and/w// *interchangeability* [Choueiry *et al,* 1998]) can improve the performance of this algorithm.

2 Preamble

Problem description and formalization

The distributed sensor network problem ([Gomes *et al,* 2002]) consists of a sensor field composed of n sensors, and m targets to be tracked. Each sensor has its own visibility *range.* The sensors can communicate among themselves, but not necessarily every sensor with every other sensor.

Some restrictions apply: 3 sensors have to be assigned to each target, and they must be able to communicate among themselves; each sensor can only track one target at a time.

In our approach, one agent corresponds to a target; each agent has 3 local variables (the sensors to be assigned to each target), and the domain of each variable is the set of sensors that can track the respective target.

There are two types of constraints: *intra-agent constraints* (the variables belonging to one agent must be assigned to different sensors, and the sensors assigned to one agent must have a communication link between themselves), and *inter-agent constraints* (no 2 variables from any 2 agents can be assigned the same value - a sensor can track a single target)

Interchangeability background

The concept of interchangeability informally means equivalence between values of a CSP variable:

- *Neighborhood Interchangeability: 2* values a and b of a variable *Vi* are A7 if they are equivalent for every constraint involving V_i;
- *Neighborhood Partial Interchangeability:* a weaker form of NI, defined for a subset of values from the local domain with respect to a set of neighbors, where the

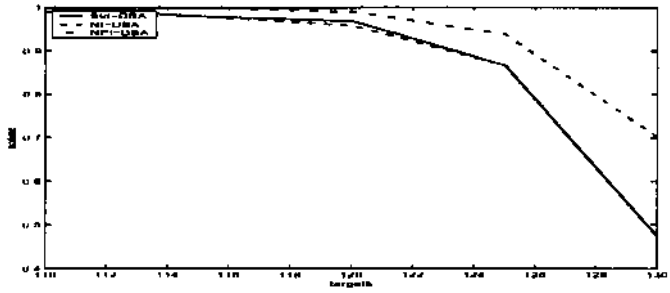

Figure 1: Percentage of solved problems

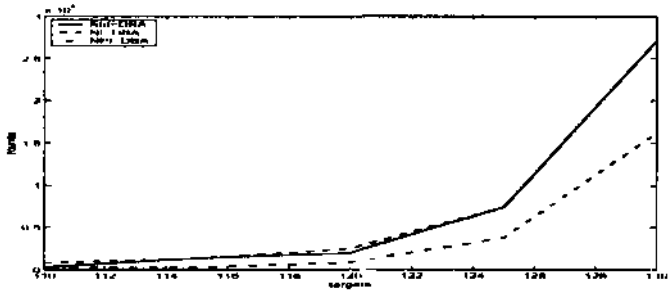

Figure 2: Average no. of rounds spent on each problem size

impact of the change of the local variable is limited to the reference set of neighbors.

3 Algorithms

Due to lack of space, we will present here only a high-level overview of the algorithms that we developed.

NI-DBA: the idea is that if we find the NI-sets for the local variables, we can safely assign values from those sets, being certain this won't cause any conflicts with the neighboring agents. The NI-sets are determined during the pre-processing phase, based on the domains of the neighbors, and are used at runtime like this: if an agent has a conflict with a neighbor, it will search for an improvement in it's local domain *giving preference to the values from the NI-set.* This avoids any future conflicts with any neighbor.

NPI-DBA: the NPI-sets are computed at runtime, w.r.t. the set of the neighbors that we already have conflicts with. When searching for a local improvement, we *give preference to the values from the NPI-sets,* thus not risking to cause future conflicts with neighbors that are not already involved, therefore keeping conflicts contained.

4 Evaluation

We made our evaluations in these settings: a sensor field with 400 sensors in total, and randomly generated *solvable* problems with 40, 60, 80, 100, 110,115, 120, 125 and 130 simultaneous targets(meaning three times as many variables).

We collected these results: problem solved/not solved (a problem is declared unsolved after the number of cycles reaches a threshold of 50000 cycles), and solving effort (time spent and number of cycles required).

For small numbers of targets, all the algorithms performed well; the differences increased with the problem difficulty, and peaked at 130 targets (most difficult problems), where NPI-DBA solved more than 70% of the problems, whereas Standard-DBA solved less than 50% (see Figure 1) Both the average number of rounds and the solving time for standard DBA are bigger than those for NPI-DBA, and close to the ones of NI-DBA. We see in Figure 2 that for difficult problems, the number of required rounds for NPI-DBA is about 40% smaller than the one of Standard DBA. A similar diagram for the time is available, but not included here.

We developed a visual interface that allows us to monitor the solving process, thus giving us clear indications that using the strategies based on NI/NPI greatly inhibits the propagation of conflicts around the constraint graph.

The initialization of the variables was random, as in Standard DBA, in order to keep the algorithms comparable, and see the improvements that the *search strategy* brings. Initialization with values from the NI-sets yields even larger improvements, leading us to believe that both the "informed" initialization of the variables and the subsequent search strategy play a role in the performance of the algorithm.

Overall, our results have shown that NPI-DBA is much better than NI-DBA. This is due to the fact that in dense problems, there is usually little or no NI at all, whereas NPI, being a weaker form of NI is still computable.

5 Conclusions and future work

The techniques presented here can be easily generalized beyond inequality constraints and resource allocation problems.

NPI-DBA clearly outperforms standard DBA for difficult problems, and NI-DBA shows comparable performance. Further speedups can be obtained with "informed" initializations, based on the NI data available after the preprocessing phase.

Further improvements could also be obtained by allowing multiple simultaneous changes of the local variables at each step, and by trying a hierarchical approach, where certain agents are delegated as "local authorities" for solving a particularly difficult local problem. It would be interesting to study in more detail the scalability of these algorithms.

References

[Freuder, 1991] Eugene C. Freuder *'Eliminating interchangeable values in CSPs'* In Proc. of AAAI 1991

[Zhang *et al,* 2002] Weixiong Zhang and Lars Wittenburg *Distributed Breakout Revisited.* In Proc. of AAAI 2002

[Gomes *et al,* 2002] Carla Gomes, Cesar Fernandez, Ramon Bejar *Communication and Computation in DisCSP Algorithms.* In Proc. of CP-2002, Ithaca, New York, USA

[Choueiry et al., 1998] Berthe Choueiry and G. Noubir: *On the Computation of Local Interchangeability in Discrete Constraint Satisfaction Problems.* In Proc. of AAAI-98.

[Morris , 1993] Morris, P. *The breakout method for escaping from local minima.* In Proceedings of the Eleventh National Conference on Artificial Intelligence, 40-45

[Yokoo et al., 1998] Makoto Yokoo *The Distributed Constraint Satisfaction Problem: Formalization and Algorithms.* IEEE Trans, on Knowledge and Data Engineering