Asynchronous Partitioning Framework (Extended Abstract)

Vitaliy Freidovich
Department of Mathematics and Computer Science
The Open University of Israel
Raanana, Israel
f_vitaliy@yahoo.com

Amnon Meisels
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
am@cs.bgu.ac.il

ABSTRACT
A new general framework for agent cooperation and coordination in solving distributed constraint satisfaction problems (DCSPs) is presented. The Asynchronous Partitioning Framework (APF) first partitions agents into groups of agents, based on some heuristic, prior to any search being conducted. During the partitioning phase one of the agents in each group is assigned the role of a group leader. Next, two distinct types of search processes among the agents are performed concurrently. The first type of search is conducted within each group, in parallel and asynchronously to all searches in other groups. The second type of search, the global search, is conducted between the groups, and treats each group as if it is a single agent represented by its group leader. The structure of the groups remains static throughout the search processes. Two distinct algorithms implementing APF are presented, and the advantages of APF are evaluated experimentally.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems, coherence and coordination.*

General Terms
Algorithms, Design, Measurement, Performance.

Keywords
Distributed Constraints Satisfaction; Distributed Search; Distributed Problem Solving; Distributed Partitioning.

1. INTRODUCTION
Distributed Constraint Satisfaction Problems (DCSPs) are composed of a set of agents. Each agent holds a set of variables, has a local representation of the constraint network, and is connected to other agents' variables by some constraints. The role of an agent is to assign values to its variables, attempting to generate a locally consistent assignment, exchanging messages for communication with constraining agents.

The present paper proposes a new approach for solving DCSPs, based on the actual structure of a given DCSP as represented by its constraints graph. The Asynchronous Partitioning Framework (APF) adheres to the fail first principle by concentrating computational efforts first on the most constrained regions of the constraints graph. This is achieved by partitioning the DCSP into groups, each representing such a region, and conducting local searches inside each group in parallel to other groups. The results of the local searches are combined into a complete solution by performing a global search between the groups.

Cite as: Asynchronous Partitioning Framework (Extended Abstract), Vitaliy Freidovich and Amnon Meisels, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1425-1426

2. PARTITIONING FRAMEWORK
The Asynchronous Partitioning Framework (APF) is composed of the following 4 distinct components:

1. The group partitioning algorithm, GroupPartition, partitions a given set of agents into disjoint groups, each group having a single group leader.
2. The local search algorithm, LocalSearch, conducts a search inside each group.
3. The global search algorithm, GlobalSearch, conducts a global search among all the groups.
4. The coordination engine is the heart of APF, and coordinates the LocalSearch and the GlobalSearch algorithms.

The search inside each group is synchronized with a token. While expanding a token, each agent tries to keep it consistent with its constraints inside its group and with constraints with agents in groups that have a higher priority than the current group, ignoring constraints with agents in lower priority groups. Each agent informs group leaders of agents in lower groups it is connected to when its value is changed, and assumes that they will adjust to its new assignment by changing their assignments if necessary. When receiving a value change notification for an agent a_i, its group leader must act upon the location of the token within the group, restarting the search at a_i, if necessary.

3. THE AGP ALGORITHM
The Asynchronous Group Partitioning (AGP) algorithm is an implementation of APF. It is composed of 4 phases: (1) Priority computation; (2) Group partitioning; (3) Group ordering, which together implement the GroupPartition algorithm, and (4) Search for a solution, which implements the coordination engine of APF.

3.1 Priority Computation Phase
The priorities of agents are calculated in a distributed manner using a simple heuristic, based on the density of the agent's neighborhood in the constraints graph, by employing the formula

```
find_group_leader:
1. agent ← first(sortedListOfNeighbors)
2. remove(agent, sortedListOfNeighbors)
3. if (agent = self)
4.     set_self_as_group_leader
5. else
6.     send(JOIN, agent)

set_self_as_group_leader:
1. groupLeader ← self
2. isGroupLeader ← true
3. add(self, group)
4. for each agent in waitingToJoinList do
5.     add(agent, group)
6.     send(JOINED, agent)
7. for each agent in listOfAllAgents do
8.     send(SET_MY_GROUP_LEADER, groupLeader, agent)

join(agent)
1. if (isGroupLeader)
2.     add(agent, group)
3.     send(JOINED, agent)
4. else if (isNotGroupLeader)
5.     send(NOT_JOINED, agent)
6. else
7.     add(agent, waitingToJoinList)

joined(agent)
1. groupLeader ← agent
2. isNotGroupLeader ← true
3. for each agent in waitingToJoinList do
4.     send(NOT_JOINED, agent)
5. for each agent in listOfAllAgents do
6.     send(SET_MY_GROUP_LEADER, groupLeader, agent)
```

Figure 1. Group partitioning phase

$Pa_i = k_i + \sum_{j=1}^{k_i}(CommonNeighbors(a_i,a_j))$, where Pa_i is a_i's priority, k_i is the number of a_i's neighbors, and $CommonNeighbors(a_i,a_j)$ returns the number of agents which are neighbors of both a_i and a_j.

3.2 Group Partitioning Phase

Figure 1 presents the group partitioning phase of agents into disjoint groups. The group partitioning phase starts by running find_group_leader, which tries to find the appropriate agent in each agent's neighborhood for being its group leader. The search is conducted according to the priorities of the different neighbors computed in the priority computation phase, equal priorities being resolved lexicographically. During this process each neighbor is fetched from sortedListOfNeighbors (ordered by priorities), and sent a JOIN message which requests it to add the current agent to its group. If it denies the request, by sending a NOT_JOINED message, find_group_leader is called again by AGP. Once the group leader of the current agent had been determined, it informs all other agents, with a SET_MY_GROUP_LEADER message.

3.3 Group Ordering & Search Phases

The group ordering phase orders the agents within each group, in accordance with the Inner Group Priority (IGP). IGP is calculated by using another simple heuristic, based on the size of groups which are connected to the current agent by some constraint.

AGP's search for solution phase is implemented by replacing

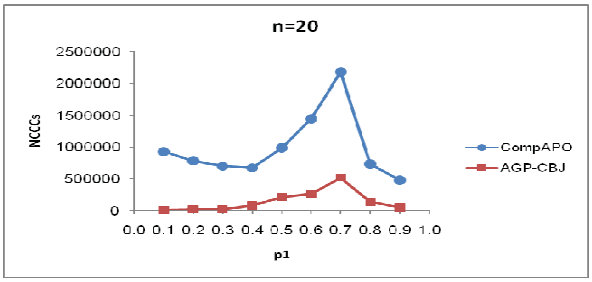

Figure 2. AGP-CBJ vs. CompAPO

APF's LocalSearch and GlobalSearch algorithms with SBT [3] and CBJ [2] algorithms, respectively.

4. THE AGP-CBJ ALGORITHM

APF is a general framework, as both the search inside each group and the search between the groups can be implemented by any algorithm preserving certain properties. To demonstrate this idea we have implemented the Asynchronous Group Partitioning Conflict Based Back-Jumping (AGP-CBJ) algorithm, as a second implementation of APF. AGP-CBJ is a natural evolution of the AGP algorithm. It is identical to the AGP algorithm, except that in AGP-CBJ the CBJ algorithm is used for the LocalSearch algorithm instead of the SBT algorithm.

5. EXPERIMENTAL EVALUATION

To evaluate the performance of APF, a number of experiments on randomly generated networks of constraints had been conducted. Figure 2 presents a comparison of the computational effort of AGP-CBJ and CompAPO [1] by Non-Concurrent Constraint Checks (NCCCs). The constraints density parameter - p_1, and the tightness parameter - p_2, were varied between 0.1 and 0.9. For each combination of p_1 and p_2, 10 different instances of problems had been generated, with 20 variables and domain of 10 values. NCCCs had been averaged for all problems with the given p_1, and all values of p_2. Clearly, AGP-CBJ outperforms CompAPO.

6. DISCUSSION

A new framework for agent cooperation and coordination in solving DCSPs was presented. APF is driven by the structure of the underlying constraints graph of a given DCSP, focusing the computational efforts on the more difficult regions of the problem first. Two distinct algorithms, AGP and AGP-CBJ which implement it were presented. AGP-CBJ was shown experimentally to outperform CompAPO.

7. REFERENCES

[1] Grinshpoun, T. and Meisels, A., "Completeness and performance of the apo algorithm", Journal of Artificial Intelligence Research, Vol.33, pp.223-258, 2008.

[2] Prosser, P., "Hybrid algorithms for the constraint satisfaction problem", Computational Intelligence, Vol.9, pp.268-299, 1993.

[3] Yokoo, M., Durfee, E. H., Ishida, T. and Kuwabara, K., "Distributed constraint satisfaction problem: Formalization and algorithms", IEEE Transactions on Data and Knowledge Engineering, Vol.10, pp.673-685, 1998.