Maintaining Soft Arc Consistencies in BnB-ADOPT$^+$ During Search

Ka Man Lei
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
kmlei@cse.cuhk.edu.hk

1 Introduction

Distributed Constraint Optimization Problems (DCOPs) have been applied in modeling and solving many multiagent coordination problems, such as meeting scheduling, sensor networks and traffic control. Several distributed algorithms for optimal DCOP solving have been proposed: ADOPT [Modi *et al.*, 2005], DPOP [Petcu and Faltings, 2005], BnB-ADOPT [Yeoh *et al.*, 2010]. BnB-ADOPT$^+$-AC and BnB-ADOPT$^+$-FDAC [Gutierrez and Meseguer, 2010a] incorporate consistency enforcement during search into BnB-ADOPT$^+$ [Gutierrez and Meseguer, 2010b], obtaining efficiency improvements. Enforcing consistency allows to prune some suboptimal values, making the search space smaller. This previous work considers unconditional deletions only, which avoids overhead in handling assignments and backtracking. However, values that could be deleted conditioned to some assignments will not be pruned with this strategy, so search space reduction opportunities are missed.

A search-based constraint solving algorithm essentially forms subproblems of the original problem by variable assignments. Our goal is to maintain soft arc consistencies in each subproblem, so that variable assignments during search are also considered in consistency enforcement. As a result, we can explore more value pruning opportunities and thus further reduce the search space. An essential contribution in Gutierrez and Meseguer's work is the introduction of an extra copy of cost functions in each agent, so that search and consistency enforcement are done asynchronously. Our contribution goes further maintaining soft arc consistencies in each subproblem during search, so that (i) search and consistency enforcement are done asynchronously, introducing some extra but finite and small number of variable domains and cost functions copies; (ii) the induced overhead caused by backtracking and undoing assignments and deletions is minimized. The asynchronicity requirement and different cost measurement (number of messages and NCCCs) require us to introduce novel techniques over those used in centralized CP. We show the benefits of our proposal on benchmarks usually unamenable to solvers without consistency.

2 Maintaining Soft Arc Consistencies

The domain deletions caused by consistency enforcement in subproblems are conditional and need to be undone upon changes in execution context. We propose to enforce AC and FDAC asynchronously in all subproblems during search by utilizing additional copies of variable domains and cost functions in each agent. We use an agent classing scheme based on the position of an agent in the problem structure. The scheme governs the required number of copies of variable domains and cost functions in an agent. With extra copies of variable domains and cost functions, the consistency enforcement in different subproblems can be performed asynchronously. However, it is unnecessary to maintain a copy for each subproblem for space efficiency. For example, Figure 1 Right illustrates the search tree, subproblems and classes of subproblems of a DCOP with 5 agents and there are four classes of subproblems. Each agent holds a variable domains and cost functions copy for each class of subproblem (Figure 1 Left). Therefore, the consistency enforcement in the four classes of subproblems can be performed asynchronously. In the following, we outline the issues and solutions used in maintaining AC and FDAC in BnB-ADOPT$^+$.

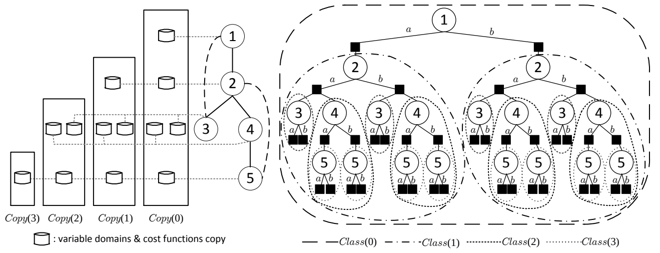

Figure 1: Left: The pseudo-tree of a DCOP. Right: Search tree (*a*/*b* domains), subproblems and classes of subproblems.

1. **Reinitialization**
   Since we do not have a variable domains and cost functions copy for each subproblem for space efficiency, the copy for a class of subproblems may have to be reinitialized when the variable assignments among ancestor agents change. The reason is that the deletion posted in a subproblem (except the original problem) is conditioned to the variable assignments the subproblem depends on and so these deletions may not occur in other subproblems when the variable assignments have changed. Therefore, these conditionally deleted values may have to be recovered when values of ancestor agents change. Otherwise, the search algorithm will search for solution based on obsolete value pruning information and may result in suboptimal solution.

The variable assignments information of ancestor agents is needed for reinitialization. This information is conveyed using VALUE, COST, DEL and UCO messages. We avoid introducing new messages since the overhead could be tremendous and we find that using the existing messages to do reinitialization is complete and safe.

2. **Backtracking**
Enforcing consistencies in a subproblem can possibly lead to empty domain in some agents involved in the subproblem, which means that the variable assignments of some ancestor agents cannot lead to an optimal solution. Therefore, at least one of the ancestor agents should change and prune its current value. A new message BTK is added to notify backtracking. BTK is sent up the pseudo-tree by the agent that obtains empty domain, until the target ancestor agent receives the message and perform backtracking. With backtracking, we can prune the search space at higher nodes in the search tree by maintaining consistencies in lower subproblems.

3. **Keeping Cost Function Copies Identical**
Two agents constrained by a cost function each holds a separate copy of the cost function for consistency enforcement. It is thus of paramount importance to ensure the two copies being identical but this task is made difficult by the asynchronous nature of the search algorithm. Simultaneous deletions in two constrained agents i and j cause projections from C_{ij} to C_i in agent j and C_j in agent i respectively. The asynchronous nature of message exchanges can result in the projections/extensions performed in different order and thus different C_{ij} copies in agents i and j respectively. Gutierrez and Meseguer [2012] propose to include two new messages to synchronize deletions but the overhead is high. We propose to solve this issue by allowing one of the two agents to undo and reorder the operations. With this *Undo Mechanism* we keep the asynchronicity and avoid extra messages. We give preference to one of the two agents. The operations will be done in the order of the preferred agent, while the non-preferred one must undo the operations that do not follow that order. The details of the Undo Mechanism are skipped due to space limitation.

4. **Transferring Deletions to Subproblems**
Consistency enforcement on subproblems is asynchronous, so redundant deletions may appear in different subproblems. If P' is a subproblem of P, the deletions effected in P must also occur in P'. That means, the value deletions effected in a problem P will also occur in all the subproblems of P. Therefore, we do not have to send out multiple DEL messages for these value deletions. When an agent receives a DEL message that include the value deletions for problem P, these deletions will also be applied to the subproblems of P in this agent.

3 Experiments and Conclusion

We evaluate our methods MAC/MFDAC by comparing to AC/FDAC on three sets of benchmarks: binary random DCOPs, Soft Graph Coloring Problems and Radio Link Frequency Assignment Problem. We have run 50 instances with a two-hours timeout for each parameter setting. Three measures of performance are used: (1) the number of messages to evaluate the communication cost, (2) the number of non-concurrent constraint checks (NCCCs) to evaluate the computation effort, and (3) the number of instances that can be solved in two hours time to evaluate the general efficiency of each algorithm. The experiments show that MAC/MFDAC substantially further reduce the total number of messages sent and NCCCs and be able to solve same or more amount of instances in two hours time. Therefore, we can conclude that including conditional deletions by enforcing consistencies in every subproblem during search is very beneficial.

To summarize, we propose methods to maintaining soft arc consistencies in every subproblem during search. In order to preserve the asynchronicities of search and consistency enforcement, we propose to include extra but small number of copies of variable domains and cost functions. We minimize the induced overhead caused by backtracking and undoing assignments and deletions by attaching information in the existing messages rather than creating new ones. We present the issues and solutions for maintaining consistencies in subproblems and ensure their correctness: (i) reinitializing variables' domains and cost functions after context changes in subproblems to ensure the search algorithm would not search on values using obsolete value pruning information, (ii) backtracking when an agent arrives at the empty domain within a subproblem so as to prune the value in upper agents which could not lead to an optimal solution, (iii) transferring deletions from subproblems to further subproblems to avoid redundant messages, and (iv) asynchronous methods to ensure identical cost functions copies in different agents by ensuring the ordering of consistency operations between every two agents. The experimental results allow us to consider the proposed methods as important steps to maintain consistencies in every subproblems asynchronously during search and improve the efficiency of optimal DCOP solving.

References

[Gutierrez and Meseguer, 2010a] Patricia Gutierrez and Pedro Meseguer. BnB-ADOPT$^+$ with several soft arc consistency levels. In *Proc. ECAI'10*, pages 67–72, 2010.

[Gutierrez and Meseguer, 2010b] Patricia Gutierrez and Pedro Meseguer. Saving redundant messages in BnB-ADOPT. In *Proc. AAAI'10*, pages 1259–1260, 2010.

[Gutierrez and Meseguer, 2012] Patricia Gutierrez and Pedro Meseguer. Improving BnB-ADOPT$^+$-AC. In *Proc. AAMAS '12*, pages 273–280, 2012.

[Modi *et al.*, 2005] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

[Petcu and Faltings, 2005] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *Proc. IJCAI'05*, pages 266–271, 2005.

[Yeoh *et al.*, 2010] William Yeoh, Ariel Felner, and Sven Koenig. BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm. *JAIR*, 38:85–133, 2010.