Efficient Handling of Complex Local Problems in Distributed Constraint Optimization [1]

David A. Burke and Kenneth N. Brown[2]

1 INTRODUCTION

Many distributed constraint optimisation algorithms require each agent to have a single variable. For agents with multiple variables, a standard approach is to compile the local problem down to a new variable whose domain is the set of all local solutions. We present two modifications to this method, which (i) reduce problem size by removing interchangeable and dominated local solutions, and (ii) speed up search by identifying values that are interchangeable with respect to specific agents. We show that the modifications give orders of magnitude improvement over the basic compilation.

2 BACKGROUND

A Distributed Constraint Optimisation Problem consists of a set of *agents*, $A=\{a_1, a_2, ..., a_n\}$, and for each agent a_i, a set $X_i=\{x_{i1}, x_{i2}, \dots, x_{im_i}\}$ of *variables* it controls, such that $\forall i{\neq}j\ X_i \cap X_j = \phi$. Each variable x_{ij} has a corresponding domain D_{ij}. $X = \bigcup X_i$ is the set of all variables in the problem. $C=\{c_1, c_2, \dots, c_t\}$ is a set of *constraints*. Each c_k has a *scope* $s(c_k) \subseteq X$, and is a function $c_k : \prod_{ij:x_{ij} \in s(c_k)} D_{ij} \rightarrow I\!N$. The *agent scope*, $a(c_k)$, of c_k is the set of agents that c_k acts upon: $a(c_k) = \{a_i : X_i \cap s(c_k) \neq \phi\}$. An agent a_i is a *neighbour* of an agent a_j if $\exists c_k : a_i, a_j \in a(c_k)$. A *global assignment*, g, is the selection of one value for each variable in the problem: $g \in \prod_{ij} D_{ij}$. A *local assignment*, l_i, to an agent a_i, is an element of $\prod_j D_{ij}$. Let t be any assignment, and let Y be a set of variables, then $t_{\downarrow Y}$ is the projection of t over the variables in Y. The global objective function, F, assigns a cost to each global assignment: $F : \prod_{ij} D_{ij} \rightarrow I\!N :: g \mapsto \sum_k c_k(g_{\downarrow s(c_k)})$. An optimal solution is one which minimises F. The solution process, however, is restricted: each agent is responsible for the assignment of its own variables, and thus agents must communicate with each other, describing assignments and costs, in order to find a globally optimal solution.

Most DCOP algorithms assume that each agent controls only a single variable. This assumption is justified by two standard reformulations [2], by which any DCOP problem with complex local problems (i.e. multiple variables in each agent) can be transformed to give exactly one variable per agent: (i) *Compilation*: for each agent, define a variable whose domain is the set of solutions to the original local problem; (ii) *Decomposition*: for each variable in each local problem, create a unique agent to manage it.

3 IMPROVING COMPILATION

To apply the *basic compilation* method to a DCOP: (i) for each agent a_i, create a new variable z_i, whose domain $D_i = \prod_j D_{ij}$ is the set of all solutions to the agent's internal problem; (ii) for each agent a_i, add a unary constraint function f_i, where $\forall l \in D_i,\ f_i(l) = \sum_{j:s(c_j) \subseteq X_i} c_j(l_{\downarrow s(c_j)})$ (i.e. the cost is the sum of the costs from all constraints which act on a_i only); (iii) for each set of agents $A_j = \{a_{j1}, a_{j2}, ..., a_{jp_j}\}$, let $R_j = \{c : a(c) = A_j\}$ be the set of constraints whose agent scope is A_j, and for each $R_j \neq \phi$, define a new constraint $C_j : D_{j1} \times D_{j2} \times \ldots \times D_{jp_j} \rightarrow I\!N :: l \mapsto \sum_{c \in R_j} c(l_{\downarrow s(c)})$, equal to the sum of the constraints in R_j (i.e. construct constraints between the agents' new variables, that are defined by referring back to the original variables in the problem). In an optimisation problem, every set of assignments of values to variables is a valid solution giving a domain of size $|D_i| = \prod_j |D_{ij}|$ for each agent a_i. The solution space for the problem is $\prod_{i=1}^n |D_i|$.

For an agent with a complex local problem, only external variables (those that have constraints to other agents) can have a direct impact on other agents. We produce an *improved compilation* by recognising that any local solutions that have identical assignments to those external variables are equivalent with respect to the distributed problem. If there is more than one optimal local solution with the same assignments to external variables, the solutions are *fully interchangeable*, and so only one is required. Also, solutions with identically assigned external variables but with sub-optimally assigned other variables are *strictly dominated* and can be ignored. We refine the compilation so that we only find *one* optimal local solution for each combination of external variables. For each agent a_i, let $p_i = \{x_{ij} : \forall c\ x_{ij} \in s(c) \rightarrow s(c) \subseteq X_i\}$ be its *private* variables – variables which are not directly constrained by other agents' variables – and let $e_i = X_i \setminus p_i$ be its *external* variables – variables that do have direct constraints with other agents. For each a_i, create a new variable z'_i with domain $D'_i = \prod_{j:x_{ij} \in e_i} D_{ij}$, and add a function f'_i, where $\forall l \in D'_i,\ f'_i(l) = \min\{f_i(t) : t \in D_i, t_{\downarrow e_i} = l\}$. That is, D'_i contains all assignments to the external variables, and their cost is the minimum cost obtained when they are extended to a full local assignment for a_i. The new constraints are defined as in the basic compilation (for each $R_j \neq \phi$,

[1] This work is supported by SFI under Grant No. 03/CE3/I405. We thank the developers of ADOPT for making their code available, and BCRI at UCC for their computing resources.

[2] Centre for Telecommunications Value-chain Research and Cork Constraint Computation Centre, Dept. Comp. Sc., UCC, Ireland

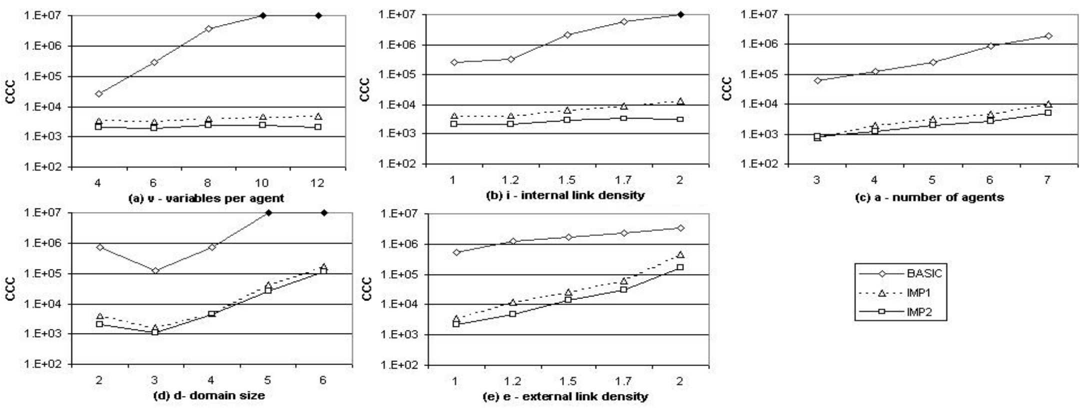

Figure 1. Concurrent Constraint Checks (cutoff = 10,000,000).

$C'_j : D'_{j1} \times D'_{j2} \times \ldots \times D'_{jp_j} \rightarrow I\!N :: l \mapsto \sum_{c \in R_j} c(l_{\downarrow s(c)}))$, but will act on smaller sets of tuples. In our reduced compilation, the size of a domain is $|D'_i| = \prod_{j:x_{ij} \in e_i} |D_{ij}|$, which is a reduction over the basic compilation by a factor of $\prod_{j:x_{ij} \in p_i} |D_{ij}|$.

4 INTERCHANGEABILITY IN DCOP

We introduce the concept of sub-neighbourhood interchangeability and apply it to distributed search using compiled values. For each agent a_i and for a set of agents S, let $h^s_i = \{x_{ij} : \exists c : x_{ij} \in s(c) \wedge a(c) \cap S \neq \phi\}$ be the set of original variables of a_i that are adjacent to original variables of the agents in S. Two compiled values, $l_a, l_b \in D_i$ are *sub-neighbourhood interchangeable* (SNI) with respect to the agents in S ($l_a \equiv^s_i l_b$), if both values represent identical assignments to the variables of h^s_i: $l_a \equiv^s_i l_b \leftrightarrow l_{a \downarrow h^s_i} = l_{b \downarrow h^s_i}$.

We apply SNI sets to the ADOPT algorithm [1]. In ADOPT, agents are prioritised into a tree. Agents send their values to all neighbours lower in the priority tree, and receive costs only from direct children. Considering each subtree separately, let S be the set of lower priority neighbours of a_i, lying in the subtree rooted by a_s. For each subtree, partition a_i's values into SNI sets, such that $\Phi^s_i(x)$ is a function returning the SNI set to which x belongs: $\forall l_a, l_b \in D_i, l_a \equiv^s_i l_b \leftrightarrow \Phi^s_i(l_a) = \Phi^s_i(l_b)$. Then, ADOPT can be modified such that if a_i receives a cost ε from a_s with a compatible context, the costs of all values interchangeable with the current value x are updated: $\forall l \in \Phi^s_i(x), cost(l, a_s) \leftarrow \varepsilon$.

5 EXPERIMENTS

We compare the effect of the two modifications to the basic compilation on random binary graph colouring problems. The problems are characterised by a 5-tuple $\langle a, e, v, i, d \rangle$: a is the number of agents, e is the external link density (# inter-agent constraints $= ea$), v is the number of variables per agent, i is the internal link density (# intra-agent constraints $= iv$) and d is the domain size of each variable. Our base problem setting is $\langle 5, 1, 6, 1, 2 \rangle$. We then compare the algorithms varying v, i, a, d and e in turn, averaging over 20 test instances. Both the basic and reduced compilations are generated using Ilog JSolver. We implement the SNI sets in ADOPT and run it in a simulated distributed environment: we use one machine but each agent runs asynchronously on its own thread.

The time required to compile agents into a single variable is always faster in the improved compilation – up to 2 orders of magnitude faster for some parameter settings. In the distributed search, we use the Concurrent Constraint Checks metric to measure performance. In Figure 1, plotted on a log scale, we compare the original compilation method *BASIC*, with our improved method *IMP1* and also *IMP2*, which uses both the improved compilation and SNI sets. The reduced domains resulting from our improved compilation, *IMP1*, give orders of magnitude improvements over the basic compilation for most parameter settings. In particular, as the number of variables in the local problem increases, the impact on our improved reformulation is minimal, while the basic reformulation quickly becomes very difficult to compute. Tests varying the internal link density, number of agents and domain size all demonstrate improvements of at least 2 orders of magnitude over *BASIC* (runs of *BASIC* that exceeded 10^7 CCC were cut off, and are shown shaded). *IMP2* also outperforms *IMP1* by on average 45%: using the SNI sets that occur in the compilation speeds up the distributed search.

6 CONCLUSION

A standard technique for handling complex local problems in DCOP is to convert multiple local variables to a single variable whose domain is the set of all local solutions. We have presented two advances on this method: we represent only one optimal local solution for each combination of external variables, discarding interchangeable and dominated solutions; and we identify values that are interchangeable with respect to neighbouring agents. We have evaluated the methods using the ADOPT algorithm, and they give orders of magnitude improvements over the basic compilation.

REFERENCES

[1] P. Modi, W. Shen, M. Tambe, and M. Yokoo, 'Adopt: Asynchronous distributed constraint optimization with quality guarantees', *Artificial Intelligence*, **161**(1–2), 149–180, (2005).

[2] M. Yokoo and K. Hirayama, 'Algorithms for distributed constraint satisfaction: A review', *Autonomous Agents and Multi-Agent Systems*, **3**(2), 185–207, (2000).