

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.

Distributed Constraint Optimisation

- q A Distributed Constraint Optimisation Problem (DCOP) consists of a set of *agents*, where each agent has a set of *variables*.
- q Each variable has a corresponding *domain of values*.
- q There exists a set of *constraints*, where each constraint acts on a subset of the variables. Each constraint is a *cost function*, specifying a cost for each combination of values assigned to the variables in its scope.
- q Two agents are *neighbours* if they contain variables that share a constraint.
- q The problem also has an *objective function*, which is a function over the constraints (e.g. summation).
- q A solution to a DCOP is an assignment to each variable a value from its domain. An optimal solution is one which minimises the objective function.
- q The solution process is restricted. Each agent is responsible for the assignment of its own variables – agents must communicate to solve the problem.

Basic Compilation

1. Find **all solutions** to each agent's internal problem – each solution corresponds to one value in the domain of a new single-variable agent.
 2. Add a unary constraint at each agent, the cost of which is the corresponding local cost for each new value.
 3. Construct new constraints between agents, that are evaluated by referring back to the original variables in the problem.
- q Every set of assignments of values to variables is a valid solution – for n agents, each with m variables, with average domain size d , we require d^{nm} space for each agent and we have a total problem solution space of size d^{nm}
 - q Basic compilation is only feasible for very small problems

Improved Compilation

- q Find **one optimal solution** for each assignment to the external variables – if we assume n agents, each with m variables, with average domain size d , for which p variables are private, we require d^{mp} space for each agent and we have a total problem solution space of size $d^{p(m-p)}$, which is a **reduction** by a factor of d^{mp} .

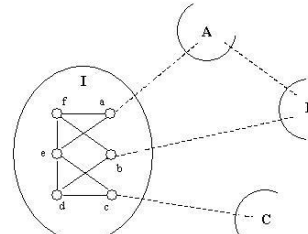
Interchangeability in DCOP

- q **Sub-neighbourhood interchangeability**: Two values x and y for a variable V are sub-neighbourhood interchangeable (**SNI**) with respect to a subset S of the neighbours of V , if and only if for every constraint C between V and variables in S , x and y satisfy C for identical sets of assignments to the other variables in C .
- q We can use SNI sets in the ADOPT DCOP algorithm to infer the costs associated with particular values.
- q In ADOPT, agents are prioritised into a tree. Each agent maintains costs for each of its subtrees, for each of its values.
- q For each subtree, calculate SNI sets for the set of neighbouring agents that appear in that subtree – all agents in a subtree affect the costs received from the child at the root of the subtree
- q When a cost is received from a child Z for a particular value x , consult SNI sets and update the costs for all values that are SNI with x for the subtree rooted by Z .

Experiments

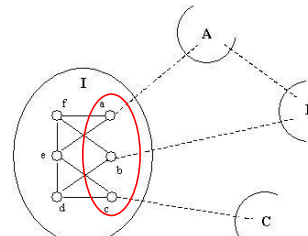
- q We compare the original compilation method *BASIC*, with our improved method *IMP1*, and also *IMP2*, which uses both the improved compilation and SNI sets.
- q We test using random binary graph colouring problems $\langle a, e, v, i, d \rangle$: a is the number of agents, e is the external link 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.
- q 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.
- q For many parameter settings (with complex local problems), the improved compilation offers **orders of magnitude** savings over the basic compilation.
- q Furthermore, using the SNI sets that occur in the compilation reduces the computation required in the distributed search by on average **45%**.

Consider an agent in a distributed 2-colouring problem with multiple local variables, three of which are external (have constraints with other agents).



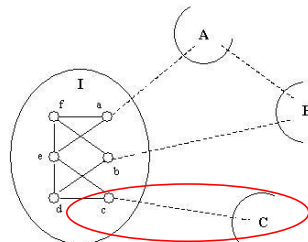
X	a	b	c	d	e	f
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	0	0	1	1
5	0	0	0	1	0	0
...
63	1	1	1	1	1	0
64	1	1	1	1	1	1

Basic compilation to new variable 'X' (64 solutions).



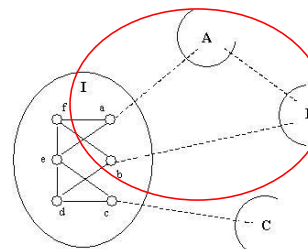
X	a	b	c	d	e	f
1	0	0	0	*	*	*
2	0	0	1	*	*	*
3	0	1	0	*	*	*
4	0	1	1	*	*	*
5	1	0	0	*	*	*
6	1	0	1	*	*	*
7	1	1	0	*	*	*
8	1	1	1	*	*	*

Improved compilation (8 solutions) – one optimal solution for each combination of external variables.



X	a	b	c	d	e	f
1	0	0	0	*	*	*
2	0	0	1	*	*	*
3	0	1	0	*	*	*
4	0	1	1	*	*	*
5	1	0	0	*	*	*
6	1	0	1	*	*	*
7	1	1	0	*	*	*
8	1	1	1	*	*	*

Consider variable c , which is an external variable linked to agent C . The original values 0 and 1 are represented 4 times each. Solutions {1,3,5,7} and {2,4,6,8} are sub-neighbourhood interchangeable (SNI) w.r.t. agent C .

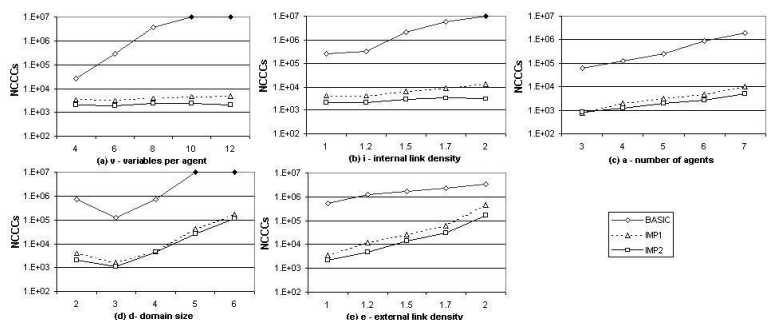


X	a	b	c	d	e	f
1	0	0	0	*	*	*
2	0	0	1	*	*	*
3	0	1	0	*	*	*
4	0	1	1	*	*	*
5	1	0	0	*	*	*
6	1	0	1	*	*	*
7	1	1	0	*	*	*
8	1	1	1	*	*	*

Variables a and b are linked to agents A and B , who have a constraint between them. Therefore an assignment to variable a indirectly affects variable b . So, we calculate SNI sets with respect to both agents. Solutions {1,2}, {3,4}, {5,6} and {7,8} are SNI w.r.t. agents A and B .

v	4	6	8	10	12
<i>BASIC</i>	0.871	1.366	2.420	5.706	17.951
<i>IMP1</i>	0.810	0.843	0.915	0.926	0.959
i	1	1.2	1.5	1.7	2
<i>BASIC</i>	1.441	1.449	1.490	1.505	1.513
<i>IMP1</i>	0.848	0.888	0.901	0.915	0.923
a	3	4	5	6	7
<i>BASIC</i>	1.117	1.325	1.424	1.543	1.667
<i>IMP1</i>	0.801	0.819	0.846	0.867	0.902
d	2	3	4	5	6
<i>BASIC</i>	1.441	3.980	13.218	41.350	105.107
<i>IMP1</i>	0.846	1.014	1.007	1.186	1.338
e	1	1.2	1.5	1.7	2
<i>BASIC</i>	1.450	1.434	1.427	1.440	1.415
<i>IMP1</i>	0.842	0.892	0.903	0.925	1.000

Time taken to compile agents.



Non Concurrent Constraint Checks (cutoff = 10,000,000).