Vol.2011-AL-134 No.23
2011/3/7

Regular Paper

Symmetry Breaking by Dominance Detection in Distributed Environments

Xavier Olive[†1] and Hiroshi Nakashima[†1]

This paper considers the approach of symmetry breaking by dominance detection and extends it to the scope of distributed constraint programming. It exploits previous results of symmetry propagation in distributed environments and proves that propagating partial symmetry information together with backtrack messages of a distributed Branch & Bound is sufficient to prune the search tree. It finally demonstrates the efficiency of such pruning by significantly cutting down the number of non-concurrent constraint checks and by improving execution time.

1. Introduction

Distributed constraint optimisation is a framework designed for modelling naturally distributed problems. Those consist of agents cooperating to solve an optimisation problem and include problems like meeting scheduling or the allocation of targets to sensors in sensor networks. In such problems, only a subset of agents knows about each constraint, leading to the idea of natural distribution. Keeping the definition of a problem distributed rather than centralising it is relevant when computational and communication power is limited.

Distribution and symmetry detection may sound incompatible by nature;[8),10)] first introduced the idea of symmetry propagation in a distributed context for the DPOP algorithm[11)], which is a distributed version of the bucket elimination algorithm. Agents propagate and exploit symmetry information together with constraints to be joined and projected. The ambition of symmetry detection in distributed environments is to ensure that a symmetry suspected by an agent is: a) valid for the global problem; b) exploited to improve performance of the resolution process.

†1 Kyoto University

This paper adapts this idea of symmetry propagation among agents for SynchBB[4)], the distributed synchronised Branch & Bound. It proves that each agent propagating symmetry information together with backtrack messages; and combining its current symmetry information with other pieces of symmetry propagated from other agents is sufficient to apply symmetry breaking by dominance detection techniques[3)], already widely known in centralised constraint programming. The result is particularly worth of attention as it does not require symmetry groups held by each agent to be valid for the whole problem in order for this symmetry breaking process to be correct.

Eventually, we validate our proposal by comparing performance between the original SynchBB and SynchBB implementing our adaptation of symmetry breaking by dominance detection to distributed environments in order to find a significant improvement on the number of non-concurrent constraint checks (#NCCCS) and on execution time, particularly on highly connected problems.

2. Distributed Constraint Programming

2.1 Definition

Definition 1 A distributed constraint optimisation problem (DCOP) is a constraint optimisation problem where variables are distributed over different agents. It consists of a finite set of variables $x_1 \cdots x_n$, a set of domains $d_1 \cdots d_n$, a set of agents $a_1 \cdots a_n$ not necessarily all different, and a set of constraints $c_1 \cdots c_t$. Each constraint c_i is defined on a scope of variables $(x_{i_1}, \ldots x_{i_k})$ and takes a positive value, or cost, on $\mathbb{R}^+ \cup \{+\infty\}$.

The objective is to find a global solution, i.e., an assignation of values to all variables so that the sum of the costs of all constraints is minimised. Constraints fall into two categories: local (private) and global (distributed) constraints. Each agent owns a local subproblem, which is a partial view of the global problem. This global problem is the union of all the local subproblems.

2.2 SynchBB

Whilst many different algorithms have been developed for distributed constraint programming like DPOP[11)] or ADOPT[7)], we focus here on a pioneer algorithm implementing a basic distributed synchronised version of Branch & Bound: SynchBB[4)].

Variable and value ordering are fixed, and a partial assignation for variables, called *path*, is exchanged among agents to be extended in a complete path, being a solution. The first agent in the ordering initiates the algorithm by sending a path containing its first value to the second agent. When an agent receives a path from its predecessor in the ordering, it appends its first value to the path and sends this new path to its successor, provided the partial cost of the new path, or lower bound (LB), is less than the current upper bound (UB); otherwise, it continues to try next values until the cost does not exceed the bound. If values are exhausted, it backtracks to its predecessor.

When an agent is notified of a backtrack from its successor agent in the ordering, it changes its assignation to next values, until the cost does not exceed the bound. If values are exhausted, another backtrack takes place.

3. Symmetry Breaking and Propagation

We present in this section the basics of symmetry in constraint programming[3], symmetry breaking by dominance detection[2], the constructive orbit problem[1] and how it can be related to propagation of symmetry information as presented in 10).

3.1 Definition

Definition 2 (Symmetry) A symmetry σ on a DCOP is an automorphism on the set of assignations of values to variables, such that for any assignation a, $f(a) = f(\sigma(a))$, where f is the global cost function of the DCOP.

The set of all symmetries in a DCOP form a group. We chose the following convenient notation to explicit symmetries which can be defined as permutations of variables and/or values: we write $\{\{x_1, x_3\}, \{x_2\}\}$ for a problem with 3 variables x_1, x_2, x_3 where x_1 and x_3 are permutable. Similarly, $\{\{1, 2, 3\}, \{4, 5\}\}$ denotes a domain where values $1, 2, 3$ are all permutable, and $4, 5$ as well; this notation implies that 1 and 4 are not permutable. We consider for simplification reasons, but without any restriction on the general case, that all variables are defined on the same domain.

3.2 Symmetry Breaking by Dominance Detection

In the Branch & Bound algorithm, symmetry breaking by dominance detection (SBDD)[2] is an operation answering the question "is this node symmetrically equivalent to a previously visited one?" If the answer is yes, then the algorithm can backtrack.

In practice, we keep a record in a list of *fail sets* S of roots of completed subtrees. Each record contains information about the decision made to reach the root of the subtree. When we reach a node, we consider P, the set of decisions corresponding to the current search node. We say that our current node is dominated by a completed subtree if there exists a σ in G, symmetry group of the problem, and an s in S such that $\sigma \cdot s \subseteq P$.

For example, if three variables x_1, x_2, x_3 all defined on $\{1, 2, 3, 4\}$ are subject to an `alldifferent` constraint. The algorithm find two solutions $(1, 2, 3)$ and $(1, 2, 4)$, then backtracks to variable x_2 and assigns $x_2 = 3$: all the assignations located in the subtree rooted $(1, 3)$ are symmetrical to assignations in the subtree rooted in $(1, 2)$: it is thus not necessary to explore it and we can already backtrack.

This method, efficient for dealing with problems with large symmetry groups[3], considers we know about the symmetry group of the problem, which is a parameter impossible to know a priori in a distributed environment.

3.3 The constructive orbit problem

Definition 3 (Orbit) The orbit of an element α is the set of elements which are images of α through any symmetry σ in the group G, and is noted $G \cdot \alpha = \{\sigma \cdot \alpha \,|\, \sigma \in G\}$.

In the SBDD context, pruning occurs when the current node is equivalent to a state in the list of fail sets: finding that two nodes are equivalent, i.e., in the same orbit, is referred to as the *orbit problem*. Another way to solve the orbit problem is to find a representative state in an orbit, e.g., the smallest in lexicographical order. Finding this canonical form for any assignation is referred to as the *constructive orbit problem*.[1] describe a way to compute the constructive orbit problem, albeit focusing on the context of model checking.

Definition 4 (Canonical form) Let α be an assignation of values to variables and G a group of symmetries over those assignations. We call canonical form of α and write $\gamma^G(\alpha)$ the smallest element in lexicographical order in $G \cdot \alpha$.

9) details an efficient way to compute the γ function when G is a combination of variable and value permutations. The simplest case is when G is a symmetry

group where all variables and all values are permutable. For example, computing $\gamma(3,2,3)$ when the three variables $\{\{x_1,x_2,x_3\}\}$ and the three values $\{\{1,2,3\}\}$ are permutable consists of counting the occurrences of each value (here, one occurrence of 2 and two occurrences of 3); then, we consider the number of occurrences in descending order and associate each of them with each value in lexicographical order, which leads us to two occurrences of 1 followed by one occurrence of 2: $\gamma(3,2,3) = (1,1,2)$.

3.4 Symmetry Propagation

10) first presented the idea of propagating and combining symmetry information between agents in distributed constraint optimisation. This scheme relies on the idea of *partial symmetry*, i.e., a symmetry on a partial definition of a problem and on two operations on symmetry groups: junction, consisting of combining symmetry information between two scopes; and projection, consisting of reducing the scope of the symmetry group.

Definition 5 A subproblem of a problem p is the restriction of p to a subset of variables V, their neighbour variables, and the constraints involving any variable in V.

Each agent naturally owns a subproblem of the global problem, restricted to its variables.

Definition 6 (Partial symmetry) A partial symmetry of a problem p is a symmetry over a subproblem of p.

Each agent can detect partial symmetries over the subproblem of its own. There is no inclusion rule between the group of global symmetries of a problem p and the group of partial symmetries seen by an agent: this means that it is impossible to find *all* symmetries of a global problem from the partial symmetries of all agents. However, if all agents agree on a partial symmetry σ, σ is also a global symmetry. More specifically, we can state the following proposition.

Proposition 1 If σ is a partial symmetry over p_1 and p_2, two subproblem of the same CSP, then σ is a partial symmetry over their reunion $p_1 \cup p_2$.

This leads us to the general definition of the junction of two symmetry groups.

Definition 7 (Junction) Let G_1 (resp. G_2) be a group of symmetries on the constraint c_1 (resp. c_2) defined on the scope of variables S_1 (resp. S_1), we define the junction $G_1 \oplus G_2$ by considering the elements of G_1 (resp. G_2):

- if $\sigma_1 \in G_1$ permutates variables in $S_1 \cap \overline{S_2}$, then $\sigma_1 \in G_1 \oplus G_2$,
- if $\sigma_2 \in G_1$ permutates $x \in S_1 \cap \overline{S_2}$ with $y \in S_1 \cap S_2$, then $\sigma_2 \notin G_1 \oplus G_2$,
- if $\sigma_3 \in G_1$ permutates variables $y \in S_1 \cap S_2$, then $\sigma_3 \in G_1 \oplus G_2$ iff. $\sigma_3 \in G_2$.

For example, with the notation defined for variable permutations, let $G_1 = \{\{x,y,z,t\}\}$ and $G_2 = \{\{x,y,u\}\}$ be two groups of symmetry of subproblems restricted respectively to $S_1 = \{x,y,z,t\}$ and $S_2 = \{x,y,u\}$. The junction $G_1 \oplus G_2$ is defined on $S_1 \cup S_2 = \{x,y,z,t,u\}$ and can be expressed as $\{\{x,y\},\{z,t\},\{u\}\}$.

With this definition of junction, we can reformulate Proposition 1 into the following:

Proposition 2 If G_1 (resp. G_2) is a group of symmetry on subproblem p_1 (resp. p_2), then $G_1 \oplus G_2$ is a group of symmetry on $p_1 \cup p_2$.

This implies that the junction of all symmetry groups as seen by each agent is a symmetry group for the global problem. However, rather than computing complete groups of symmetries for each agent, we prefer to focus on restrictions of those groups on the scopes of variables owned by each agent. This leads us to the definition of the projection of a variable out of a symmetry group.

Definition 8 (Projection) Let G be a symmetry group on the scope S and x a variable in S. $G|_x$ is the restriction of G on $S - \{x\}$. By definition, all symmetries of $G|_x$ are symmetries of the original group G.

For example, if a symmetry group can be represented as the following combination of variable and value permutations $G = \{\{x,z\},\{y\}\},\{\{1,2\},\{3\}\}$, then $G|_z = \{\{x\},\{y\}\},\{\{1,2\},\{3\}\}$.

4. Distributed SBDD

Rather than trying to find all symmetries of a DCOP, which is not reasonable when we want to keep the definition of a problem distributed; or to find symmetries of a DCOP through the junction of all agents' symmetry groups, we present an algorithm which dynamically builds partial symmetry groups.

4.1 Algorithm

We consider a list of variables $(x_1, x_2, \cdots x_n)$. Those variables are defined on domains that we will consider identical for clarity reasons and name $(u_1, u_2, \cdots u_p)$. For each constraint whose scope is $(x_{i_1}, x_{i_2}, \cdots, x_{i_k})$, the variable the least in lexicographic order, i.e., x_{i_k}, is responsible for evaluating its cost. Each variable x_i

is aware of a group of symmetry G_i. The symmetries intrinsic to each constraint owned by x_{i_k} are joined into G_{i_k}.[⋆1]

An assignation initiated to $\alpha_0 = \varnothing$ is sent by the root of the search tree to x_1. When a variable x_{i+1} receives an assignation α_i, it creates α_{i+1} by appending $x_{i+1} \leftarrow u_1$: if α_{i+1} does not break any of the constraints held by x_{i+1}, it passes α_{i+1} to x_{i+2}; otherwise, it tries with next value u_2. When the domain is exhausted, x_{i+1} backtracks and projects x_{i+1} out of G_{i+1} and sends the result $G_{i+1}|_{x_{i+1}}$ back to x_i together with the backtrack message.

When x_i receives a backtrack message from x_{i+1}, it updates its symmetry information $G_i \leftarrow G_i \oplus G_{i+1}|_{x_{i+1}}$ and, if G_i is not empty, will record the current couple $\left(\gamma^{G_i}(\alpha_i), G_i\right)$, i.e., the lowest element $\gamma^{G_i}(\alpha_i)$ of α_i's orbit with respect to G_i. Each variable x_i keeps track of a list of couples $\left(t_i^1, t_i^2, \cdots\right)$, where $t_i^k = \left(\gamma^{G_i^k}\left(\alpha_i^k\right), G_i^k\right)$.

The symmetry breaking strategy is then based on the following proposition.

Proposition 3 Let β_i be an assignation made by variable x_i. If there exists a $t_i^k = \left(\gamma^{G_i^k}\left(\alpha_i^k\right), G_i^k\right)$ such that $\gamma^{G_i^k}(\beta_i) = \gamma^{G_i^k}\left(\alpha_i^k\right)$, then the subtree rooted in β_i can be pruned.

Proof For any partial assignation α made by variable x, we name $\mathsf{LB}(\alpha)$ the *lower bound* calculated with α; $\mathsf{UB}^{\downarrow}(\alpha)$ and $\mathsf{UB}^{\uparrow}(\alpha)$ the *upper bound* respectively before and after the traversal of the subtree rooted in α. The process of updating upper bounds as defined in Branch & Bound ensures that for any assignation ε in the subtree rooted in α, if ε triggered a backtrack, the inequality $\mathsf{LB}(\varepsilon) \geq \mathsf{UB}^{\uparrow}(\alpha)$ stands.

If we know in advance, i.e., before traversing the subtree rooted in β_i, that for any leaf assignation δ in the subtree rooted in β_i, $\mathsf{LB}(\delta) \geq \mathsf{UB}^{\downarrow}(\beta_i)$, then we can prune the subtree rooted in β_i, knowing that $\mathsf{UB}^{\uparrow}(\beta_i) = \mathsf{UB}^{\downarrow}(\beta_i)$.

Let $j > i$ be the largest index of the variable accessed in the tree parse before reaching α_i^k. Then, G_i^k is the symmetry group obtained by joining $G_i, G_{i+1}, \ldots, G_j$ and projecting $x_{i+1}, \ldots x_j$ according to the process described hereabove, that is

$$G_i^k = G_i \oplus \left(G_{i+1} \oplus \left(G_{i+2} \oplus \left(\cdots G_j\right)|_{\ldots}\right)|_{x_{i+2}}\right)|_{x_{i+1}}$$

Let δ be a leaf assignation in the subtree rooted in β_i and δ_j its leading part for variables $x_1, \ldots, x_j$; δ_j can therefore be expressed as an assignation to variables $x_1, \ldots, x_j$ whose leading part for $x_1, \ldots, x_i$ is β_i. The positivity of constraints ensures that $\mathsf{LB}(\delta) \geq \mathsf{LB}(\delta_j)$.

We introduce the following lemma in order to proceed further this proof.

Lemma 1 There exists a σ in G_i^k such that $\alpha_i^k = \sigma(\beta_i)$ and $\mathsf{LB}(\delta_j) = \mathsf{LB}(\sigma(\delta_j))$, where $\sigma(\delta_j)$ is in the subtree rooted in α_i^k.

Proof of Lemma 1 Let us consider first the case where $j = i + 1$. The lower bound evaluated for δ_j is by definition obtained by adding the lower bound for β_i and the sum of the constraints owned by x_j, namely $f_j(\delta_j)$:

$$\mathsf{LB}(\delta_j) = \mathsf{LB}(\beta_i) + f_j(\delta_j)$$

We have $\mathsf{LB}(\beta_i) = \mathsf{LB}\left(\alpha_i^k\right)$ by definition; on the other hand, $f_j(\delta_j) = f_j(\sigma(\delta_j))$ is ensured by the inclusion $G_i \oplus G_j|_{x_j} \subseteq G_i \oplus G_j$. We can therefore easily unfold:

$$\begin{aligned}\mathsf{LB}(\sigma(\delta_j)) &= \mathsf{LB}\left(\alpha_i^k\right) + f_j(\sigma(\delta_j)) \\ &= \mathsf{LB}(\beta_i) + f_j(\delta_j) \\ &= \mathsf{LB}(\delta_j)\end{aligned}$$

The general case for any j arises from a simple induction. □

Let us come back to the proof of Proposition 3 and distinguish the two following cases.

(1) Let us assume first that β_i is the first target of the pruning by dominance detection.
Since $\sigma(\delta_j)$ is in the subtree rooted in α_i^k and j is the deepest level in the traverse of the subtree, there exists a leading part ε_l of $\sigma(\delta_j)$ (with $i < l \leq j$) which triggered a backtrack in the search process of the subtree. Therefore, Branch & Bound ensures that $\mathsf{LB}(\varepsilon_l) \geq \mathsf{UB}^{\uparrow}(\alpha_i^k)$ and the fact that the chronological series of evaluations of (UB) by all variables x_i is non increasing lets us assert that $\mathsf{UB}^{\uparrow}\left(\alpha_i^k\right) \geq \mathsf{UB}^{\downarrow}(\beta_i)$.

⋆1 Without loss of generalisation, each agent can apply a symmetry detection algorithm on its subproblem in order to initialise its group of partial symmetries G_{i_k}. However, in our examples, we reasonably assume the definition of one constraint includes its intrinsic symmetry definition, e.g., as the definition $x \neq y$ is stated, the property that all variables and all values are permutable is naturally deduced. Then, the junction of the symmetry groups of all constraints owned by the same agent is a valid symmetry group for the subproblem owned by this agent.

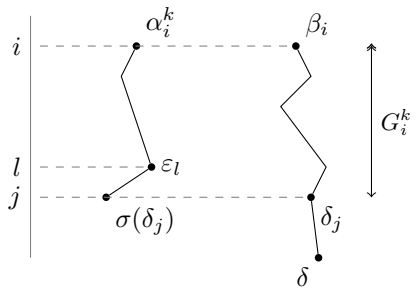

Fig. 1 Piece of the search tree

Therefore as we unroll the following sequence of equalities and inequalities (refer to Figure 1):

$$\mathsf{LB}(\delta) \geq \mathsf{LB}\left(\delta_j\right) = \mathsf{LB}\left(\sigma\left(\delta_j\right)\right) \geq \mathsf{LB}\left(\varepsilon_l\right) \geq \mathsf{UB}^{\uparrow}\left(\alpha_i^k\right) \geq \mathsf{UB}^{\downarrow}(\beta_i)$$

we get the proof that $\mathsf{LB}(\delta) \geq \mathsf{UB}^{\uparrow}(\beta_i)$ to allow us to prune the subtree rooted in β_i and to have $\mathsf{UB}^{\downarrow}(\beta_i) = \mathsf{UB}^{\uparrow}(\beta_i)$.

(2) In the general case, there exists a leading part ε_l of $\sigma(\delta_j)$ (with $i < l \leq j$) which either triggered a backtrack, or which was pruned by symmetry detection. If ε_l triggered a backtrack, we can refer to the preceding case. If ε_l was pruned by symmetry detection, let us consider two cases:

(a) $l = j$ is impossible, because pruning at level j by symmetry detection requires that a backtrack from level $j+1$ was triggered at least once;

(b) $l > j$. There exists a $t_l^{k'} = \left(\gamma^{G_l^{k'}}\left(\tau\left(\varepsilon_l\right)\right), G_l^{k'}\right)$ such that $\gamma^{G_l^{k'}}\left(\tau\left(\varepsilon_l\right)\right) = \gamma^{G_l^{k'}}\left(\varepsilon_l\right)$. We can apply the same reasoning in order to find

$$\mathsf{LB}\left(\sigma\left(\delta_j\right)\right) \geq \mathsf{UB}^{\uparrow}\left(\varepsilon_l\right) \geq \mathsf{UB}^{\downarrow}\left(\beta_i\right)$$

□

As a consequence, when computing the cost of a optimisation problem, each agent checks whether the canonical form of the current assignation is equivalent to any recorded assignation with respect to the group of symmetry known at the time of the record. If so, we may prune the subtree rooted in the current assignation; if not, we can compute the lower bound of the current assignation.

4.2 Example Problem

Let us consider the following simple problem. We have 4 variables x, y, z, t, each owned by a different agent. They are all defined on $\{0, 1, 2\}$ and constrained by the following rules we describe here intensively:

$$(x, y, z) : \begin{cases} \texttt{alldifferent} & \mapsto & 0 \\ x = y = z & \mapsto & +\infty \\ & \text{otherwise} & 20 \end{cases}$$

$$(z, t) : \begin{cases} z = t & \mapsto & +\infty \\ z \neq t & \mapsto & 5 \end{cases}$$

We simulate in Table 1 the sequence of assignations leading to find a solution while taking symmetries into account. Figure 2 illustrates with a red colour the nodes that are dominated by others. The symmetry information are registered in each agent/variable. The search tree is significantly pruned.

This problem displays an example where not all t_z^k correspond to the same symmetry group: symmetry information known by agent/variable z before and after the backtrack to $(0, 0, 1)$ is different: $\{\{x, y, z\}\}$ becomes $\{\{x\}, \{y, z\}\}$. Even though x and z are not permutable in the global problem, the orbit of assignation $(0, 0, 0)$ with respect to the symmetry group $\{\{x, y, z\}, \{0, 1, 2\}\}$ can be considered as part of the list of fail sets.

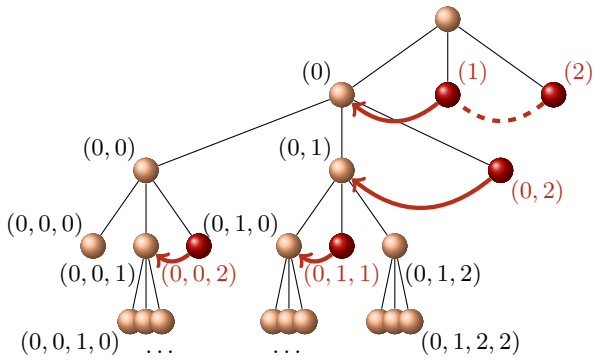

Fig. 2 Distributed SBDD with symmetry propagation — red nodes are symmetrically equivalent to preceding states and have been pruned in the tree walk.

	assignation	LB	UB	G_i
x	(0)	0	$+\infty$	
y	$(0,0)$	0	$+\infty$	
z	$(0,0,0)$	$+\infty$	$+\infty$	$\{\{x,y,z\}\},\{\{0,1,2\}\}$
z	$(0,0,0)$	BT	$+\infty$	$\{\{x,y,z\}\},\{\{0,1,2\}\}$
z				$t_z^1 = ((0,0,0),\{\{x,y,z\}\},\{\{0,1,2\}\})$
z	$(0,0,1)$	20	$+\infty$	$\{\{x,y,z\}\},\{\{0,1,2\}\}$
t	$(0,0,1,0)$	25	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,0,1,1)$	$+\infty$	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,0,1,2)$	25	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
z	$(0,0,1)$	BT	25	$\{\{x,y\},\{z\}\},\{\{0,1,2\}\}$
z				$t_z^2 = ((0,0,1),\{\{x,y\},\{z\}\},\{\{0,1,2\}\})$
z	$(0,0,2)$	BT		$\gamma_z^2(0,0,2) = (0,0,1)$
y	$(0,0)$	BT	25	$\{\{x,y\}\},\{\{0,1,2\}\}$
y				$t_y^1 = ((0,0),\{\{x,y\}\},\{\{0,1,2\}\})$
y	$(0,1)$	0	25	$\{\{x,y\}\},\{\{0,1,2\}\}$
z	$(0,1,0)$	20	25	$\{\{x,y\},\{z\}\},\{\{0,1,2\}\}$
t	$(0,1,0,0)$	$+\infty$	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,1,0,1)$	25	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,1,0,2)$	25	25	$\{\{z,t\}\},\{\{0,1,2\}\}$
z	$(0,1,0)$	BT	25	$\{\{x,y\},\{z\}\},\{\{0,1,2\}\}$
z				$t_z^3 = ((0,1,0),\{\{x,y\},\{z\}\},\{\{0,1,2\}\})$
z	$(0,1,1)$	BT		$\gamma_z^3(0,1,1) = (0,1,0)$
z	$(0,1,2)$	0	25	$\{\{x,y\},\{z\}\},\{\{0,1,2\}\}$
t	$(0,1,2,0)$	5	5	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,1,2,1)$	5	5	$\{\{z,t\}\},\{\{0,1,2\}\}$
t	$(0,1,2,2)$	$+\infty$	5	$\{\{z,t\}\},\{\{0,1,2\}\}$
z	$(0,1,2)$	BT	5	$\{\{x,y\},\{z\}\},\{\{0,1,2\}\}$
z				$t_z^4 = ((0,1,2),\{\{x,y\},\{z\}\},\{\{0,1,2\}\})$
y	$(0,1)$	BT	5	$\{\{x,y\}\},\{\{0,1,2\}\}$
y				$t_y^2 = ((0,1),\{\{x,y\}\},\{\{0,1,2\}\})$
y	$(0,2)$	BT		$\gamma_y^2(0,2) = (0,1)$
x	(0)	BT	5	$\{\{x\}\},\{\{0,1,2\}\}$
x				$t_x^1 = ((0),\{\{x\}\},\{\{0,1,2\}\})$
x	(1)	BT		$\gamma_x^1(1) = (0)$
x	(2)	BT		$\gamma_x^1(2) = (0)$

Table 1 Tree walk for distributed SBDD with symmetry propagation (BT stands for "backtrack")

5. Performance

We chose to evaluate this way of propagating symmetry groups and pruning search trees by executing various instances of the distributed graph colouring problem, equivalent to many real-life problems, such as job scheduling or register allocation. Although much optimisation has been produced for the centralised graph colouring problem, we compare performance between strictly distributed methods, namely the original SynchBB with and without our adaptation of SBDD to distributed environments. We based our comparison on the number of non-concurrent constraint checks (#NCCCS) introduced in 6) and on the execution time.

Definition 9 (#NCCCS) Non-concurrent constraint checks are counted in a similar way Lamport's logical time is achieved: instead of steps of computation, the number of non-concurrent constraint checks is counted to measure the local computational effort.

In practice, with SynchBB, the #NCCCS is equal to the total number of constraint checks if all variables are connected, i.e., if the constraint graph is formed of only one component. In order to take the symmetry detection overhead into account, we counted one constraint check for each evaluation of the γ function.

We modelled our distributed graph colouring problem with n variables $x_1 \cdots x_n$. Each variable represents a node x_p, which has a constraint stating the impossibility of having identical colours for this node and each of its neighbours (x_{p_i}). We named that constraint `graphcolouring`$(x_p, x_{p_0}, \cdots x_{p_k})$, k being the number of neighbours of x_p. The cost function assigns $+\infty$ if there exists an i such as $x_p = x_{p_i}$, and 0 otherwise. This constraint naturally induces a group of value symmetries (all colours are permutable) and of variable symmetries (all x_{p_i} are permutable). We generated several distributed graph colouring problems with 6 colours, 4 agents, each owning 5 variables. The symmetries present in this problem being combinations of variable and value symmetries, we used the implementation of the γ function as introduced in Section 3.3 and described in depth in 9).

On Figure 3, we compared the number of non-concurrent constraint checks on 200 instances of the problem: one resolution was executed on a regular SynchBB, the other with a SynchBB with dominance detection. Dominance detection was efficient with respect to the #NCCCS on 90 % of the executions. This means that on these executions, computing the γ function was compensated by the pruning of the search tree.

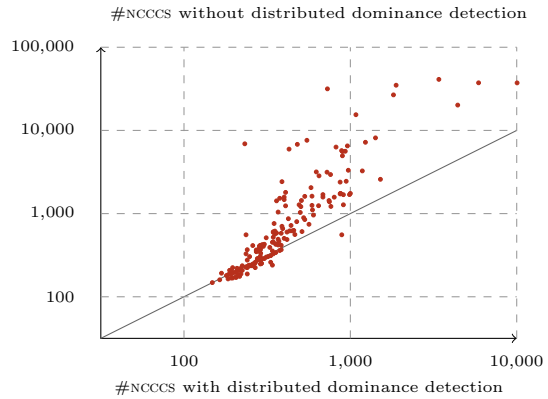

Fig. 3 Number of non-concurrent constraint checks (#NCCCS) for 200 distributed graph colouring problem instances. Each dot represents one instance; the coordinates depicts the #NCCCS for a resolution with SynchBB with (x-axis) and without distributed dominance detection (y-axis).

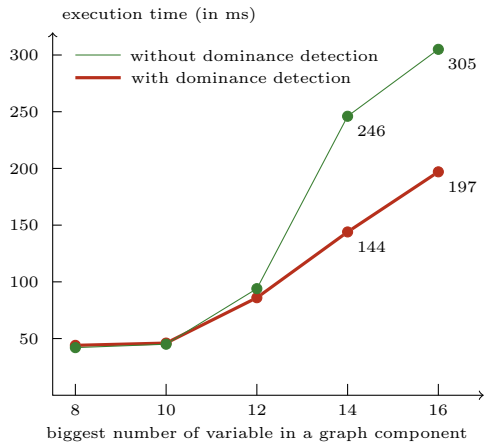

Fig. 4 Execution time in ms for distributed graph colouring problem instances solved with SynchBB algorithm with and without distributed dominance detection

Figure 4 draws an average of execution times[⋆1] according to the biggest number of variables in a graph component, i.e., the depth of the deepest search tree in the problem. Dominance detection brought an average speedup of 1.71 (resp. 1.54) on the total amount of executions for trees of depth 14 (resp. 16): our method yielded better performance in deeper trees in general.

6. Conclusion

This paper presented a distributed version of symmetry breaking by dominance detection for the distributed Branch & Bound (SynchBB), where symmetry information are sent together with backtracking messages, and adequately joined at that time with neighbouring agents' symmetry information. It proved that it is sufficient to rely on the list of fail sets created with respect to symmetry information known at the time of their creation in order to prune the search tree. Indeed, updating an agent's symmetry information does not question the validity of previously recorded fail sets.

The proposed method showed a significant execution speed-up (around 1.7) on problems with deep search trees and a serious cut on the number of non-concurrent constraint checks on 90 % of generated instances of the distributed graph colouring problem.

References

1) Donaldson, A.F. and Miller, A.: On the constructive orbit problem, *Annals of mathematics and artificial intelligence*, Vol.57, pp.1–35 (2009).
2) Fahle, T., Schamberger, S. and Sellmann, M.: Symmetry Breaking, *Principles and Practice of Constraint Programming* (Walsh, T., ed.), LNCS 2239, Springer, pp. 93–107 (2001).
3) Gent, I., Petrie, K. and Puget, J.-F.: Symmetry in Constraint Programming, *Handbook of constraint programming* (Rossi, F., van Beek, P. and Walsh, T., eds.), Elsevier Science Ltd, chapter10, pp.329–376 (2006).
4) Hirayama, K. and Yokoo, M.: Distributed Partial Constraint Satisfaction Problem, *Principles and Practice of Constraint Programming*, pp.222–236 (1997).
5) Léauté, T., Ottens, B. and Szymanek, R.: FRODO2.0: An Open-Source Framework for Distributed Constraint Optimization, *Proceedings of the 11th International Workshop on Distributed Constraint Programming* (Hirayama, K., Yeoh, W. and

⋆1 Frodo framework[5], Core2Duo based Linux, 2GB RAM, `java-6-sun-1.6`

Zivan, R., eds.), pp.160–164 (2009).
6) Meisels, A., Kaplansky, E., Razgon, I. and Zivan, R.: Comparing performance of distributed constraints processing algorithms, *Proceedings of the 3rd International Workshop on Distributed Constraint Reasoning*, pp.86–93 (2002).
7) Modi, P., Shen, W., Tambe, M. and Yokoo, M.: ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees, *Artificial Intelligence*, Vol.161, No.1-2, pp.149–180 (2005).
8) Olive, X. and Nakashima, H.: SymDPOP: Adapting DPOP to exploit partial symmetries, *Proceedings of the 12th International Workshop on Distributed Constraint Programming* (van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M. and Sen, S., eds.), pp.38–52 (2010).
9) Olive, X. and Nakashima, H.: Around the Constructive Orbit Problem in Distributed Constraint Programming, *IPSJ SIG Notes*, Vol.2011-AL-134 (2011).
10) Olive, X. and Nakashima, H.: Efficient Representation of Constraints and Propagation of Variable–Value Symmetries in Distributed Constraint Reasoning, *Journal of Information Processing*, Vol.19 (2011).
11) Petcu, A. and Faltings, B.: A Scalable Method for Multiagent Constraint Optimization, *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Vol.19, p.266 (2005).