Global Constraints in Distributed Constraint Satisfaction
Christian Bessiere, Ismel Brito, Patricia Gutierrez, Pedro Meseguer

To cite this version:

HAL Id: lirmm-00748192
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00748192
Submitted on 5 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Global Constraints in Distributed Constraint Satisfaction
(Extended Abstract)

Christian Bessiere
Université Montpellier 2, LIRMM-CNRS
Montpellier, France
bessiere@lirmm.fr

Ismel Brito, Patricia Gutierrez, Pedro Meseguer
IIIA, CSIC, Universitat Autonoma de Barcelona
Bellaterra, Spain
{ismel|patricia|pedro}@iiia.csic.es

ABSTRACT
Global constraints have been crucial for the success of centralized
constraint programming. Here, we propose the inclusion of global
constraints in distributed constraint satisfaction. We show how this
inclusion can be done, considering different decompositions for
global constraints. We provide experimental evidence of their bene-
fits on several benchmarks solved with the ABT algorithm.

Categories and Subject Descriptors
I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms
Algorithms

Keywords
Distributed constraint satisfaction, global constraints

1. INTRODUCTION
Global constraints have been crucial in the development of effi-
cient constraint solvers [5]. They allow to capture global properties
on an unbounded set of variables. In many cases, the exploitation
of the semantic associated with each global constraint allows to codify
propagators able to reach local consistency levels (typically gener-
alized arc consistency, GAC) with polynomial complexity. This
is a great advantage with respect to GAC propagators for generic
non-binary constraints, which have complexity exponential in the
constraint arity.

Often, it is implicitly assumed that distributed constraint reasoning
precludes the use of global constraints. With the usual assump-
tion that each agent contains a single variable (so agents and vari-
ables can be used interchangeably), an agent knows the constraint
with each one of its neighbors, and nothing else [6]. These con-
straints are obviously binary. But this interpretation is too restric-
tive because there are distributed applications for which it is natural
to use global constraints.

When adding global constraints in distributed reasoning we ob-
tain several benefits. First, the expressivity of distributed constraint
reasoning is enhanced since there are relations among several vari-
ablesto be expressed as a conjunction of binary relations
(most global constraints are not binary decomposable). Second, the
solving process can be done more efficiently. Local consistency can
be more efficiently achieved when global constraints are involved
[5]. Assuming a solving strategy maintaining some kind of local
consistency, using global constraints improves its efficiency.

Accepting the interest of global constraints in distributed con-
straint reasoning, another question naturally follows: since some
Global constraints have been crucial for the success of centralized
constraint solvers [5]. A global constraint
is a class of constraints defined by a Bool-
ean function \( f_c \) whose arity is not fixed. Constraints with different
arities can be defined by the same Boolean function. For instance,
\textit{alldifferent} \( (x_1, x_2, x_3) \) and \textit{alldifferent} \( (x_1, x_4, x_5, x_6) \) are two
instances of the \textit{alldifferent} global constraint, where \( f_{\text{alldifferent}}(T) \)
returns true iff \( x_i \neq x_j \forall x_i, x_j \in T \). A global constraint \( C \)
is \textit{contractible} iff for any tuple \( t \) on \( x_{i_1}, \ldots, x_{i_{p+1}} \), if \( t \) satisfies
\( C(x_{i_1}, \ldots, x_{i_{p+1}}) \) then the projection \( \{ x_{i_1}, \ldots, x_{i_p} \} \) of \( t \) on the
first \( p \) variables satisfies \( C(x_{i_1}, \ldots, x_{i_p}) \) [4]. A global constraint \( C \)
is \textit{binary decomposable without extra variables} iff for any instance
\( C(T) \) of \( C \), there exists a set \( S \) of binary constraints involving only
variables in \( T \) such that the solutions of \( S \) are the solutions of
\( C(T) \) [1]. \( S \) is a \textit{binary decomposition} of \( C(T) \). In the following,
we write \( C \) for a global constraint, while \( C(T) \) means a particular
instance of that global constraint on the set of variables \( T \).

We consider three different representations for a global constraint
instance: \textit{direct}, \textit{nested} and \textit{binary}. In the \textit{direct representation},
\( C(T) \) is posted as a single constraint that allows all tuples on \( T \)
satisfying \( C \). Each agent in \( T \) includes \( C(T) \) in its constraint set.
The \textit{nested representation} is applicable to all contractible global
constraints. The nested representation of \( C(T) \) with \( T = (x_{i_1},
\ldots, x_{i_p}) \) is the set of constraints \( \{ C(x_{i_1}, \ldots, x_{i_j}) \} \ j \leq 2 \ldots p \).
Each agent in \( T \) includes all constraints of the nested representation
of \( C(T) \) that involve its variable in its constraint set. The \textit{binary
representation} is applicable to all global constraints that are binary
decomposable. The binary representation of \( C(T) \) is the set
of constraints of its binary decomposition. Each agent in \( T \) includes
all constraints of the binary decomposition of \( C(T) \) that involve
its variable in its constraint set. The three representations for the
alldifferent($x_1, x_2, x_3, x_4$) global constraint appear in Figure 1.

Considering ABT as the solving algorithm, it is worth noting that ABT—originally proposed for binary constraints—can be easily generalized to handle constraints of any arity [2]. We assume that our ABT version contains such generalization.

In the direct representation, $C(T)$ is posted as a single constraint. Each agent in $T$ knows it. The lowest priority agent of $T$ in the ABT order is in charge of evaluating it. Other agents in $T$ put a link between themselves and that agent. In the nested representation, $C(T)$, $T = (x_{i_1}, \ldots, x_{i_p})$, is represented by the set of constraints $\{C(x_{i_j}, \ldots, x_{i_k}) \mid j \in 2 \ldots p\}$. Thanks to the extra constraints that are posted, the checking of $C(T)$ is not postponed to the last agent in $T$. In the binary representation, $C(T)$ is represented by the set of constraints of its binary decomposition. These three representations of a global constraint instance are equivalent from the semantic point of view (they produce the same solutions). But they cause different ABT executions, so they can be seen as different models with dissimilar efficiency.

3. PROPAGATING GLOBAL CONSTRAINTS

Independently of the way a global constraint is included into ABT, this algorithm can be enhanced maintaining some form of local consistency during search. This was already investigated in [3], where limited/full forms of arc consistency (AC) were maintained during ABT execution for binary DisCSPs. While in [3] a limited form of AC causing unconditional deletions and full AC causing conditional deletions were considered, in this paper we only maintain the limited form of GAC that causes unconditional deletions (GAC because constraints may have arity higher than 2). Clearly, this limited GAC, that from now on we call UGAC, is less powerful than full GAC. Maintaining full GAC in the distributed context would cause a substantial load of extra messages which could overcome the benefits of domain pruning. We enforce UGAC on each considered global constraint by adapting the methods achieving GAC on them—developed in the centralized case—to this distributed setting, making them work inside each agent.

Before search, a suitable preprocess makes the problem GAC (before search any value deletion is unconditional, so GAC is equivalent to UGAC). During search, UGAC is enforced as follows: in ABT execution, if agent $self$ receives a nogood message justifying the removal of its value $v$ where the nogood has an empty left-hand side (see [6, 3] for details), $v$ can be unconditionally deleted from its domain. A deletion in the domain of $x_{self}$ is propagated maintaining UGAC on the constraints connecting $x_{self}$ with other variables, which may cause further deletions. Since the initial deletion is unconditional, deletions caused by the propagation are also unconditional.

To maintain UGAC during ABT search, some modifications are needed over the ABT algorithm: (1) the domain of variables constrained with $self$ has to be represented in $self$; (2) only the agent owner of a variable can modify its domain; if agent $i$ deduces that a value could be deleted from the domain of $x_i$, it does nothing because that deduction will be done by agent $j$ at some point; (3) there is a new message DEL to notify of value deletions: DEL($self, k, v$)—informing that $self$ removes $v$ from the domain of $x_{self}$—is sent from $self$ to every agent $k$ constrained with it; (4) a suitable preprocess makes all constraints GAC before ABT starts. These changes do not modify ABT correctness and completeness.

4. EXPERIMENTS AND SUMMARY

We evaluated the impact of the addition of global constraints on random binary DisCSPs including instances of alldifferent and at-most global constraints. For ABT on DisCSPs with loose binary constraints, the most efficient representation is the binary one, followed by nested and finally direct. For ABT on DisCSPs with tight binary constraints, the most efficient representation is the direct one, followed by nested and finally binary. The same pattern appears considering both the number of exchanged messages and the number of non-concurrent constraint checks (NCCCs). For ABT-UGAC, enforcing UGAC propagation during search causes a drastic efficiency improvement from medium to high tightness, while the ranking of representations remains the same.

As summary, in this paper we propose the use of global constraints in distributed constraint reasoning, considering three different ways to represent global constraints. We evaluate the performance of ABT with or without UGAC maintenance on random DisCSPs containing some global constraints. We conclude that UGAC propagation of global constraints is never harmful in terms of messages, and in some cases it can significantly reduce the search space. Regarding the different representations of global constraints, the direct representation is the less efficient one. For DisCSPs with loose binary constraints, the binary representation wins but for DisCSPs without solution, this representation degrades quickly generating too many nogood messages. The nested representation seems to offer a good compromise: it is never worse than direct, and in some cases it is better than binary. This is good news: there are many more constraints that are contractible than constraints that are binary decomposable.

5. ACKNOWLEDGMENTS

Christian Bessiere is partially supported by the FP7-FET ICON project 284715. Christian Bessiere and Pedro Meseguer are partially supported by the CNRS-CSIC integrated action 2010FR0040. Ismel Brito, Patricia Gutierrez and Pedro Meseguer are partially supported by the project TIN2009-13591-C02-02. Patricia Gutierrez has an FPI scholarship BES-2008-006653.

6. REFERENCES