On two perspectives in decomposing constraint systems

Equivalences and computational properties

Cees Witteveen
Department of Electrical Engineering,
Mathematics and Computer Science,
Delft University of Technology, The Netherlands

Wiebe van der Hoek and **Michael Wooldridge**
Department of Computer Science,
University of Liverpool, United Kingdom

Abstract

Decomposition is a technique to split a problem in a number of parts such that some global property of the problem can be obtained or preserved by concurrent processing of these parts. In artificial intelligence applications, as for example in scheduling, almost always the aim of decomposition is to find solutions to a (global) constraint system by distributed computations. Decomposition should enable the merging of local solutions to a global solution and, therefore, decomposition should aim at preserving solutions. Another aim of decomposition, often encountered in the database and sensor network communities, is to preserve consistency: as long as adding constraints to a local constraint store does not cause any inconsistencies, the consistency of the global constraint store should be preserved.

Although satisfying these preservation properties seem to require different decomposition modes, we show that in fact these properties are equivalent: whenever a decomposition is consistency preserving, it is also solution preserving and vice-versa. We then show that the complexity of finding such decompositions is polynomially related to finding solutions for the original constraint system, explaining the popularity of decomposition applied to tractable constraint systems. Finally, we address the problem of finding optimal decompositions and show that in general, even for tractable constraint systems, this problem is hard.

1 Introduction

Background

Decomposition is a technique to split a problem in a number of parts such that some global property of the problem can be obtained or preserved by independent, distributed, processing of the parts. There are at least two reasons for the attractiveness of this problem solving method. First of all, from a purely computational point of view, decomposition offers significant advantages, because a problem instance that could require a huge amount of computational resources when solved as a whole, often can be resolved rather easily by solving smaller subproblems and combining their outcomes. Secondly, since solving the subproblems by decomposition does not require any interaction during the problem solving process, they can be solved concurrently by independent problem solving processes. This latter aspect attracted some attention from the multi-agent community, because it allows a set of autonomous agents each to solve a part of a problem completely independently from each other: There is no need to provide complex interaction schemes to ensure the coordination of the agents in solving the problem. Hence, this method can also be used in cases where communication between the agents during problem solving is a major obstacle.

The key class of problems we focus upon in this paper are constraint problems, solvable by constraint processing. Constraint solving is a very general problem solving approach for dealing with combinatorial problems ranging from scheduling and time tabling to automated reasoning and database handling. It is a popular topic of research in such diverse areas as operations research, artificial intelligence research, databases and geometric modelling. The basic idea behind constraint solving is to represent combinatorial problems by a *constraint system*. The basic ingredients of a constraint system $\mathcal{S}$ are a set C of constraints over a set X of variables x_i each taking values in some domain of values D_i. A constraint system $\mathcal{S}$ is said to be solved if we have found values $d \in D_i$ for each of the variables $x_i \in X$ such that all the constraints $c \in C$ are satisfied. In artificial intelligence research, constraint systems have been used to represent such diverse problems as planning and scheduling, resource allocation, design and configuration problems (Dechter 2003). In the database community, constraints have also been used as *integrity constraints* to ensure the integrity of data stored and manipulated (Gupta and Widom 1993).

In both areas, *distributed constraint systems* have been a major focus of research. In a distributed constraint system, one distinguishes a set of agents A_i each controlling a disjoint subset of variables $X_i \subseteq X$. Each agent A_i is responsible for those constraints whose variables occur in its control set X_i (its set of *local constraints*). Agents have to interact with other agents for solving the set of *global constraints*[1], whose variables occur in different components X_i. If communication between the agents is difficult or agents do not

[1] Although the term "global constraints" in the CP-literature refers to constraints encapsulating sets of other constraints, in the context of distributed constraint processing we use this term only to distinguish them from local constraints. That is, global constraints are those constraints whose variables occur in more than one control set.

wish to communicate, enforcing these global constraints becomes an issue. Therefore, quite some research has been done on *decomposing* distributed constraint systems, that is, to replace each global constraint by a suitable set of local constraints, in such a way that the need for interaction during the problem solving process is eliminated (Hunsberger 2003; Gupta and Widom 1993; Brodsky, Kerschberg, and Varas 2004; Mazumbar and Chrysantis 2004).

Here, however, the focus of research within the artificial intelligence (AI) community has been quite different from the focus in the database community. In AI applications one typically assumes that the local constraints are controlled by agents whose (common) task is to assign suitable values to the variables such that all constraints are satisfied. Decomposition has to ensure that the local constraints can be solved completely independently from the others, after which the local solutions can always be *merged* to yield a solution to the complete system. Hence, in AI research the focus of decomposition has been on a *solution preserving* property : in obtaining a global solution, local solutions should always be preserved in order to ensure independent local problem solving. For example, in (Hunsberger 2002) decomposition[2] has been applied to ensure that independently chosen schedules for subnetworks of an STN can always be merged to a joint schedule of the total network. In (Karimadini and Lin 2009) a decomposition technique is presented to ensure decentralized cooperative control of multi-agent systems where satisfaction of all (distributed) subtasks of a joint task implies the fulfillment of the complete task as well.

On the other hand, the focus in the database and sensor network community has been on the use of integrity constraints for distributed databases that ensure that, whatever local information satisfying these constraints is added to the (distributed) database, the consistency of the total database will be preserved (Alwan, Ibrahim, and Udzir 2009; Gupta and Widom 1993; Brodsky, Kerschberg, and Varas 2004). Hence, in the database community, one focuses on the *preservation of consistency* in constraint systems: how to guarantee that when updating local databases and only ensuring their local consistency, the consistency of the resulting *global* constraint system (i.e. the union of all local databases + integrity constraints) will remain consistent, too. In the sensor network community, the decomposition problem is known as the *localization* problem, where a distributed constraint is reformulated into local constraints for mobile entities and is adjusted dynamically (Pietrzyk, Mazumdar, and Cline 1999; Mazumbar and Chrysantis 2004). The satisfaction of the distributed constraint is guaranteed whenever all the local constraints are satisfied.

Research Questions

A first question one would thus like to answer is: *what is the exact relationship between preservation of solutions versus preservation of consistency notions*: are these decomposition properties equivalent, does one imply the other, or are they independent? By analyzing the underlying notions and presenting a framework for decomposition, in this paper we will show that preserving consistency and preserving solutions are equivalent properties.

Having identified these two perspectives on decomposition, next, one would like to know how difficult is it to *find a suitable decomposition* (preserving solutions or preserving consistency), given a set of agents each controlling a part of the variables in a constraint system. Again, we will provide an answer by proving that finding a solution preserving decomposition for a constraint system is as hard as finding a solution for the constraint system as a whole.global constraint

Thirdly, often finding just a suitable solution preserving decomposition is not enough and we would like to find an *optimal solution preserving decomposition*. We show that, in general, finding an optimal solution preserving decomposition of a constraint system is intractable, even in cases where we can find a solution for the constraint system efficiently.

Remark 1 We note that there are other views on decomposition in constraint systems as expressed by *structural decomposition methods* (Gottlob, Leone, and Scarcello 1999; Wah and Chen 2006; Cohen, Gyssens, and Jeavons 2006; Naanaa 2009) and by the *distributed constraint optimization (DCOP) approach* (Yokoo and Hirayama 1996; Modi et al. 2003). In the structural decomposition view *(i)* the structure of the problem (i.e., the set of constraints) dictates the way in which the subproblems are generated and *(ii)* in general, the decomposition will not allow the subproblems to be *independently* solvable. In the DCOP approach, the partitioning of the variables is given, but, in general, the result of decomposition is not a set of independently solvable subproblems. Our approach differs from these approaches in the sense that, using the autonomous agent perspective, unlike the structural decomposition approach, we are interested in decomposition methods that take a *given* partitioning of the variables into account. Secondly, unlike the DCOP and structural decomposition approach, we require a *complete decomposition* of the original problem instance, that is, we would like to find a set of subproblems that can be solved *concurrently* and *independently* to obtain a complete solution to the original instance.

This paper is organised as follows. In Section 2 we discuss the necessary technical preliminaries. In Section 3 we discuss the equivalence between consistency and solution preserving decompositions. In Section 4 we show that solving a constraint system is –neglecting polynomial differences– as hard as finding a decomposition for it. Then, in Section 5, we discuss the notion of an optimal decomposition showing that finding an optimal decomposition should be considered as an intractable problem. Finally, in Section 6, we state some final conclusions to place this work into a broader perspective.

2 Preliminaries

In this section we briefly define constraint systems, distributed constraint systems, and decompositions of dis-

[2]Hunsberger has adopted the term *temporal decoupling* for decomposition in STNs.

tributed constraint systems.

Constraint Systems

A constraint system is a tuple $\mathcal{S} = \langle X, D, C\rangle$ where X is a (finite) set of variables, D is a set of (value) domains D_i for every variable $x_i \in X$, and C is a set of constraints over X. We assume constraints $c \in C$ to be specified as formulas over some language. To preserve generality, we don't feel the need to specify the set of allowable operators used in the constraints $c \in C$ and their interpretation. A solution σ of the system is an assignment $\sigma = \{x_i \leftarrow d_i\}_{i=1}^n$ to all variables in X such that each $c \in C$ is satisfied. The set of such solutions σ is denoted by $Sol(\mathcal{S})$. $\mathcal{S}$ is called *consistent* if $Sol(\mathcal{S}) \neq \emptyset$. We assume the set of solutions $Sol(\mathcal{S})$ to be anti-monotonic in the set of constraints; that is, if $\mathcal{S} = \langle X, D, C\rangle$ and $\mathcal{S}' = \langle X, D, C'\rangle$ are such that $C \subseteq C'$, then $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$. For every $c \in C$, let $Var(c)$ denote the set of variables mentioned in c. For a set of constraints C, we put $Var(C) = \bigcup_{c\in C} Var(c)$. Given $\mathcal{S} = \langle X, D, C\rangle$, we obviously require $Var(C) \subseteq X$. If D is a set of value domains D_i for variables $x_i \in X$ and $X' \subset X$ then $D_{X'}$ is the set of value domains D_i of the variables $x_i \in X'$. Likewise, given a set of constraints C and a set of variables X', we let $C_{X'}$ denote the subset $\{c \in C \mid Var(c) \subseteq X'\}$ of constraints over X'.

Distributed constraint systems and decompositions

In this paper we consider constraint systems $\mathcal{S}$ that are *distributed* (Yokoo and Hirayama 1996); that is, there is a set of N agents A_i, each being able to make assignments to or to add constraints over a subset X_i of the set X of variables. Here, we assume that agents do not share control over the variables, and that every variable is controlled by an agent. Hence, the collection $\{X_i\}_{i=1}^N$ constitutes a *partitioning* of X, i.e., $\bigcup_{i=1}^N X_i = X$, and, for $1 \leq i < j \leq N$, $X_i \cap X_j = \emptyset$.

Given a constraint system $\mathcal{S} = \langle X, D, C\rangle$ and a partitioning $\{X_i\}_{i=1}^N$ of X, we call the tuple $(\mathcal{S}, \{X_i\}_{i=1}^N)$ a *distributed constraint system*, derived from $\mathcal{S}$. We are particularly interested in those distributed systems $(\mathcal{S}, \{X_i\}_{i=1}^N)$ where each agent A_i, controlling the set X_i, only processes a set of constraints over X_i, and does not take into account other constraints. That effectively implies that in such a case, instead of one constraint system $\mathcal{S}$ and a partition $\{X_i\}_{i=1}^N$, we have a *set* of independent constraint systems $\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle$, where each C_i' is a set of constraints over X_i, i.e., $Var(C_i') \subseteq X_i$. We call the resulting set $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ of such subsystems a *decomposed constraint system*[3]. We say that such a decomposition $\mathcal{S}'$ is consistent if $\bigcup_i C_i'$ is consistent. In order to relate a distributed constraint system $(\mathcal{S}, \{X_i\}_{i=1}^N)$ to a decomposed constraint system $\mathcal{S}'$, we discuss two ways in which decomposed systems can be used to process the constraints in $\mathcal{S}$.

[3]For the moment, we do not specify any relationship between C_i' and C_{X_i}.

Solution preserving decompositions

A decomposed system can be used to *preserve solutions* of a global constraint system: the individual solutions σ_i of the subsystems $\mathcal{S}_i$ of a decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ can be merged to compose a global solution σ for a global constraint system $\mathcal{S}$. In that case the decomposed system is said to be solution preserving if *each collection* of local solutions σ_i can be used to compose a global solution σ:

Definition 1 *Let $(\mathcal{S}, \{X_i\}_{i=1}^N)$ be a distributed constraint system. Then the decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is said to be a solution-preserving decomposition w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$ if $\emptyset \subseteq Sol(\mathcal{S}_1) \times Sol(\mathcal{S}_2) \times \ldots \times Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$. $\mathcal{S}'$ is said to be* strictly solution preserving *if the first inclusion is strict whenever $Sol(\mathcal{S}) \neq \emptyset$.*[4]

Example 1 *Let $\mathcal{S} = \langle X, D, C\rangle$ be a constraint system where $C = \{x_1 \wedge x_2,\ x_1 \vee x_3,\ x_1 \vee x_4\}$ is a set of boolean constraints over $X = \{x_1, x_2, x_3, x_4\}$. If X is partitioned as $\{X_1 = \{x_1, x_2\}, X_2 = \{x_3, x_4\}\}$, the decomposition $\{\mathcal{S}_1, \mathcal{S}_2\}$ where $\mathcal{S}_1 = \langle\{x_1, x_2\}, D_1, \{x_1 \wedge x_2\}\rangle$ and $\mathcal{S}_2 = \langle\{x_3, x_4\}, D_2, \emptyset\rangle$ is a strictly solution preserving decomposition of $\mathcal{S}$: $\mathcal{S}_1$ has a unique solution $Sol(\mathcal{S}_1) = \{\{x_1 \leftarrow 1, x_2 \leftarrow 1\}\}$, while $\mathcal{S}_2$ has a "universal" solution set: $Sol(\mathcal{S}_2) = \{\{x_3 \leftarrow i,\ x_4 \leftarrow j\} : i, j \in \{0, 1\}\}$. Every solution in $Sol(\mathcal{S}_1) \times Sol(\mathcal{S}_2)$ is a solution to $\mathcal{S}$, because x_1 as well as x_2 is assigned to true. Hence, $\{\mathcal{S}_1, \mathcal{S}_2\}$ is strictly solution preserving.*

Note that, in general, not every solution $\sigma \in Sol(\mathcal{S})$ will be obtainable as the merge of local solutions σ_i.

Consistency preserving decompositions

In distributed database applications one distinguishes local constraints from global (integrity) constraints. Usually, in such applications, agents are free to add constraints to their set of local constraints as long as the resulting set remains consistent. The problem then is to ensure that local consistency ensures global consistency. This global consistency has to be ensured by the set of integrity constraints. In order to prevent communication overload between the distributed sites, one often tries to distribute these integrity constraints over the sites in such a way that satisfaction of all the local versions of the constraints imply the satisfaction of the global constraints. To simplify the discussion, we concentrate on the case where each site is allowed to *add* constraints to their local store. Consistency preservation then means that the total set of original constraints + locally added constraints is consistent, whenever the added information does not cause any local inconsistency. We need the following definitions.

Definition 2 *Let $\mathcal{S} = \langle X, D, C\rangle$ be a constraint system. An extension of $\mathcal{S}$ is a constraint system $E(\mathcal{S}) = \langle X, D, C'\rangle$ where $C \subseteq C'$.*

Definition 3 (consistency preserving extensions) *Let $(\mathcal{S}, \{X_i\}_{i=1}^N)$ be a distributed constraint system, where $\mathcal{S} =$*

[4]This is needed to take care for inconsistent constraint systems.

$\langle X, D, C\rangle$. *A decomposition* $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ *is called* consistency preserving *w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$ *if the following condition holds: whenever every local extension* $E(\mathcal{S}_i) = \langle X_i, D_i, C_i''\rangle$ *of* $\mathcal{S}_i$ *is consistent,* $E(\mathcal{S}) = \langle X, D, C \cup (C_1'' - C_1') \cup \ldots \cup (C_N'' - C_N')\rangle$ *is consistent.*
$\mathcal{S}'$ *is said to be strictly consistency preserving if, moreover, it holds that every* $\mathcal{S}_i$ *is consistent whenever* $\mathcal{S}$ *is consistent.*

Example 2 *Consider the constraint system specified in Example 1. It is not difficult to show that the given decomposition* $\{\mathcal{S}_1, \mathcal{S}_2\}$ *is also a strictly consistency preserving decomposition w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$*: Every constraint over* X_i *added to the local constraint systems* $\mathcal{S}_i$ *that keeps it consistent, will imply the existence of a non-empty set of solutions for* $\mathcal{S}_i$*. But then, using the solution preservation property, there exists a solution satisfying the total set of constraints, showing that the decomposition is strictly consistency preserving as well.*

3 Consistency and solution preserving decompositions

Given the two preservation properties we distinguished in decompositions of constraint systems, the first question we should answer is how they are related: are they independent, is one subsumed by the other, or are they in fact equivalent?

Intuitively, it seems not hard to conclude that consistency preservation is subsumed by solution preservation: whatever information is added to a local constraint store, if the result is consistent, a solution for the global store can be found by solution preservation. Hence, there should be a global solution, and, therefore, it is not difficult to show that the global constraint store + the added constraints is a consistent set as well. More precisely:

Proposition 1 *Let* $(\mathcal{S}, \{X_i\}_{i=1}^N)$ *be a distributed constraint system. If* $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ *is solution preserving w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$*, then* $\mathcal{S}'$ *is also consistency preserving w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$*.*

PROOF Assume $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ to be solution preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$. Then we have

$$Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S}).$$

For $i = 1, 2, \ldots, N$, consider arbitrary (consistent) extensions $E(\mathcal{S}_i) = \langle X_i, D_i, C_i''\rangle$ of the local subsystems $\mathcal{S}_i$. For each subsystem $E(\mathcal{S}_i)$, select an arbitrary assignment $\sigma_i \in Sol(E(\mathcal{S}_i))$.

Since $C_i'' \supseteq C_i'$, it follows that

$$\emptyset \neq Sol(E(\mathcal{S}_i)) \subseteq Sol(\mathcal{S}_i).$$

Hence, by solution preservation, the assignment $\sigma = \sigma_1 \sqcup \ldots \sqcup \sigma_N$ will satisfy $\mathcal{S}$. Therefore, $\sigma \models C$. By definition of σ_i and the fact that every $C_i'' - C_i'$ is a set of constraints over X_i, and the sets X_i are disjoint, it follows that

$$\sigma \models (C_1'' - C_1') \cup (C_2'' - C_2' \cup \ldots (C_N'' - C_N').$$

Hence,

$$\sigma \models C \cup (C_1'' - C_1') \cup (C_2'' - C_2' \cup \ldots (C_N'' - C_N')$$

and therefore, $\sigma \in Sol(E(\mathcal{S}))$. So, $Sol(E(\mathcal{S})) \neq \emptyset$ and, consequently, $\mathcal{S}'$ is consistency preserving with respect to $(\mathcal{S}, \{X_i\}_{i=1}^N)$. □

Perhaps surprisingly, the converse is also true: consistency preservation implies solution preservation. The intuition behind this result is that every solution to a constraint system can be *encoded* as a special update of the constraint store. The resulting constraint store will have this solution as its unique solution. By consistency preservation, the resulting global constraint store will be consistent. Hence, this decomposition will also be solution preserving, since the merge of all local solutions will be the unique solution of the resulting system.

Proposition 2 *Let* $(\mathcal{S}, \{X_i\}_{i=1}^N)$ *be a distributed constraint system. If* $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ *is consistency preserving w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$*, then* $\mathcal{S}'$ *is also solution preserving w.r.t.* $(\mathcal{S}, \{X_i\}_{i=1}^N)$*.*

PROOF Assume $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ to be consistency preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$. By assumption, for every subsystem $\mathcal{S}_i$ and every extension $E(\mathcal{S}_i) = \langle X_i, D_i, C_i''\rangle$ of $\mathcal{S}_i$, it must hold that, whenever the extended local systems $E(\mathcal{S}_i)$ are consistent, then the global extended system

$$E(\mathcal{S}) = \langle X, D, C \cup (C_1'' - C_1') \cup \ldots \cup (C_N'' - C_N'\rangle$$

is also consistent.

For each $i = 1, \ldots, N$, let σ_i be an arbitrary solution to $\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle$. Since $\{X_i\}_{i=1}^N$ is a partition, the assignment $\sigma = \sigma_1 \sqcup \ldots \sqcup \sigma_N$ is well-defined. We have to show that $\sigma \in Sol(\mathcal{S})$.
For $i = 1, \ldots, N$, consider the extensions

$$E(\mathcal{S}_i) = \langle X_i, D_i, C_i''\rangle,$$

where

$$C_i'' = C_i' \cup \{x = \sigma(x) : x \in X_i\}.$$

That is, each C_i' is extended with a set of unary constraints encoding the assignment $x \leftarrow \sigma(x)$ for every variable $x \in X_i$. Then, for every $i = 1, 2, \ldots, N$, $E(\mathcal{S}_i)$ is consistent and each σ_i is the unique solution of $E(\mathcal{S}_i)$.
By consistency preservation, the extension

$$E(\mathcal{S}) = \langle X, D, C \cup (C_1'' - C_1' \cup \ldots \cup (C_N'' - C_N')\rangle$$

is consistent, too. Hence $Sol(E(\mathcal{S})) \neq \emptyset$. Now observe that

$$C \cup (C_1'' - C_1') \cup \ldots \cup (C_N'' - C_N') = C \cup \{x = \sigma(x) : x \in X\}$$

Hence, it follows that σ is the *unique solution* of $E(\mathcal{S})$ and therefore, $\sigma \models C$. Hence $\sigma \in Sol(\mathcal{S})$ and the decomposition $\mathcal{S}'$ is also solution preserving. □

As a consequence of both propositions we have that a decomposition is consistency preserving iff it is solution preserving. It is not difficult to show that this equivalence also holds for the strictly preserving versions. This immediately implies that all results that have been obtained for consistency preserving decompositions such as occur in (Brodsky, Kerschberg, and Varas 2004; Mazumbar and Chrysantis 2004) can be used for solution preserving approaches to decomposition as well.

4 Finding solution preserving decompositions

The equivalence between solution preserving and consistency preserving decompositions does not tell us how we could obtain such decompositions. In this section, we will discuss the problem of finding suitable decompositions. Given the above proven equivalence, in this section we concentrate on the solution preservation property of decompositions.

First, we prove the equivalence between our notion of solution preserving decompositions and the notion of *safe decompositions* as introduced by (Brodsky, Kerschberg, and Varas 2004) for the purpose of consistency preserving decompositions. Then, using the definition of safe decompositions we show that deciding whether a decomposition $\mathcal{S}'$ is solution preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$ in general is a coNP-complete problem. Next, we prove that finding such a decomposition $\mathcal{S}'$ is as hard (neglecting polynomial differences) as finding a solution for the original system $\mathcal{S}$.[5]

We start with defining the notion of a safe decomposition:

Definition 4 ((Brodsky, Kerschberg, and Varas 2004)) *Given a distributed constraint system $(\mathcal{S}, \{X_i\}_{i=1}^N)$, the decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is said to be a safe decomposition w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$ if $\bigcup_{i=1}^N C_i' \models C$.*

Proposition 3 *The decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is safe w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$ iff $\mathcal{S}'$ is solution preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$.*

PROOF [Sketch] Assume that the decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is solution preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$. Then $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$. Take an arbitrary assignment σ satisfying $\bigcup_{i=1}^n C_i'$. Then σ can be written as $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \ldots \sqcup \sigma_N$, where each σ_i satisfies C_i' since $\{X_i\}_{i=1}^N$ is a partitioning. Therefore, for $i = 1, 2, \ldots, N$, $\sigma_i \in Sol(\mathcal{S}_i)$. By solution preservation we have $\sigma \in Sol(\mathcal{S})$. Therefore, $\sigma \models C$ and the decomposition is safe w.r.t. $\mathcal{S}$.
Conversely, assume the decomposition $\mathcal{S}'$ to be safe w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$. Then $\bigcup_{i=1}^n C_i' \models C$. So every assignment $\sigma : X \to D$ satisfying $\bigcup_{i=1}^n C_i'$ will also satisfy C. Each such a solution σ can be written as $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \ldots \sqcup \sigma_N$ where each $\sigma_i : X_i \to D_i$ satisfies C_i'. Hence, $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$ and $\mathcal{S}'$ is solution preserving w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$. □

Using this notion of a safe decomposition, it is rather easy to show that deciding whether a decomposition $\mathcal{S}'$ is safe w.r.t. a tuple $(\mathcal{S}, \{X_i\}_{i=1}^N)$, is a coNP-complete problem:

Proposition 4 *Let $(\mathcal{S}, \{X_i\}_{i=1}^N)$ be a distributed constraint system and $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ be a decomposition. Then the problem to decide whether $\mathcal{S}'$ is safe w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$ is coNP-complete.*

PROOF To show that the problem is in coNP, just guess an assignment satisfying $\bigwedge_{i=1}^N C_i'$, but falsifying C. This shows the complement is in NP. coNP-hardness immediately follows from a reduction from the coNP-complete LOGICAL CONSEQUENCE problem: Given two propositional formulas ϕ and ψ, does it hold that $\phi \models \psi$. To see this, given arbitrary ϕ and ψ, let X_1 be the non-empty set of propositional atoms occurring in ϕ and ψ. Let $X_2 = \{y\}$ where y does not occur in X_2. Consider the constraint system $\mathcal{S} = \langle X, D, C\rangle$ where $X = X_1 \cup X_2$, D is a set of boolean domains and $C = \{\phi, \psi \vee y, \neg y\}$. Let $\mathcal{S}_1 = \langle X_1, D_{X_1}, \{\phi\}\rangle$ and $\mathcal{S}_2 = \langle X_2, D_{X_2}, \{\neg y\}\rangle$. Then $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$ is a safe decomposition w.r.t. $(\mathcal{S}, \{X_1, X_2\})$ iff $(\phi \wedge \neg y) \models \{\phi, \psi \vee y, \neg y\}$ iff $\phi \models \{\phi, \psi\}$ iff $\phi \models \psi$. □

So, unless P=NP, it is hard to decide whether a decomposition is solution preserving. We can, however, obtain a more detailed result by relating the difficulty of finding a strictly solution preserving decomposition for a constraint system $\mathcal{S}$ belonging to a class of constraint systems to the difficulty of finding a solution to $\mathcal{S}$:

Proposition 5 *Let $\mathcal{C}$ be an arbitrary class of constraint systems allowing at least equality constraints. Then there exists a polynomial algorithm to find a solution for constraint systems $\mathcal{S}$ in $\mathcal{C}$ iff there exists a polynomial algorithm that, given a constraint system $\mathcal{S} \in \mathcal{C}$ and an arbitrary partition $\{X_i\}_{i=1}^N$ of X, finds a strictly solution preserving decomposition w.r.t. $(\mathcal{S}, \{X_i\}_{i=1}^N)$.*

PROOF Suppose that there exists a polynomial algorithm A to find a solution for constraint systems in $\mathcal{C}$. We show how to construct a polynomial algorithm for finding a decomposition for an arbitrary partition of such a constraint system. Let $\mathcal{S} \in \mathcal{C}$ be constraint system and $\{X_i\}_{i=1}^N$ an arbitrary partitioning of X. To obtain a decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ of $\mathcal{S}$, first, using A, we compute a solution σ of $\mathcal{S}$. For every X_i, let

$$C_{\sigma_i} = \{x = d \mid x \leftarrow d \in \sigma, x \in X_i\}$$

be a set of unary constraints for variables in X_i directly obtained from σ. Then the subsystems $\mathcal{S}_i = (X_i, D_i, C_i')$ are simply obtained by setting $C_i' = C_{X_i} \cup C_{\sigma_i}$. Note that each of these subsystems $\mathcal{S}_i$ has a unique solution $\sigma_i = \{x \leftarrow d \in s \mid x \in X_i\}$ and the merging of these solutions σ_i equals σ, i.e. a solution to the original system $\mathcal{S}$. Clearly, $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is a solution preserving decomposition for $\mathcal{S}$ that can be obtained in polynomial time.

Conversely, suppose we can find a strictly solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ for a constraint system $\mathcal{S} \in C$ w.r.t. any partitioning $\{X_i\}_{i=1}^N$ in polynomial time. We show how to obtain a solution σ of $\mathcal{S}$ in polynomial time.[6] Since the decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ can be obtained for *any* partitioning of X, we choose the partitioning $\{X_i\}_{i=1}^N$ where $X_i = \{x_i\}$ for $i = 1, 2, \ldots, N$. Since the decomposition can be obtained in polynomial time, it follows that $|\bigcup_{i=1,\ldots N} C_i'|$ is polynomially bounded in the size of the input $\mathcal{S}$. Hence, the resulting decomposed subsystems $\mathcal{S}_i$ each consist of a polynomially bounded set of *unary* constraints. It is well

[5] Here, we assume that the class of allowable constraints always comprises the class of unary equality constraints of the form $x = d$ where $d \in Dom(x)$.

[6] The case where $\mathcal{S}$ is inconsistent is easy and omitted, here.

known that such constraint systems are solvable in polynomial time (Cohen, Gyssens, and Jeavons 2006). Therefore, in polynomial time for each subsystem $\mathcal{S}_i$ an arbitrary value $d_i \in D_i$ for x_i can be obtained, satisfying all constraints. Let $\sigma_i = \{x_i \leftarrow d_i\}$ denote the solution obtained for $\mathcal{S}_i$. Since $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i\rangle\}_{i=1}^N$ is a solution preserving decomposition, the merging $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \ldots \sqcup \sigma_N$ must be a solution of $\mathcal{S}$ as well. Therefore, σ is a solution of $\mathcal{S}$, too. Hence, given a polynomial algorithm for achieving a solution preserving decomposition, we can construct a solution $\sigma \in Sol(\mathcal{S})$ in polynomial time. □

Given the equivalence between solution preserving and consistency preserving constraint systems, we now may conclude:

Theorem 1 *Finding a strictly consistency preserving decomposition as well as finding a strictly solution preserving decomposition for a constraint system $\mathcal{S}$ is as hard as finding a solution for $\mathcal{S}$.*

It is well-known that for general constraint systems, finding a solution is NP-hard (Dechter 2003). This theorem explains why sometimes finding decompositions for a constraint system is easy: one should restrict one's attention to tractable constraint systems as STNs (Hunsberger 2002) or linear arithmetic constraints (Brodsky, Kerschberg, and Varas 2004).

5 Optimal solution preserving decompositions

Finding an arbitrary solution preserving decomposition for a given distributed constraint system might not always be sufficient. One important property we also should pay attention to is the amount of information that is preserved in determining a (solution preserving) decomposition.

Note that every decomposition enables the merging of local solutions to obtain a global solution. Hence the total set of global solutions obtainable by using a decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i\rangle\}_{i=1}^N$ for $\mathcal{S}$ is the set $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_N)$. The information loss due to the decomposition therefore can be defined as $Sol(\mathcal{S}) \setminus (Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_N))$: the set of solutions of the original system that cannot be obtained by merging the local solutions using the decomposition.

Therefore, given a distributed constraint system $(\mathcal{S}, \{X_i\}_{i=1}^N)$, we would like to call a solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i\rangle\}_{i=1}^N$ an *optimal decomposition* if it minimizes $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ where $Sol(\mathcal{S}')$ is the set of solutions obtainable from the decomposed system.[7]

This optimality problem can be easily shown to be intractable, even if the underlying constraint system contains two variables and one (binary) constraint:

Proposition 6 *Let $(\mathcal{S}, \{X_i\}_{i=1}^N)$ be a distributed constraint system where $|X| = 2$ and C contains only one binary constraint. Then the problem to find an optimal solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i\rangle\}_{i=1}^N$ w.r.t. $\mathcal{S}, \{X_i\}_{i=1}^N)$ is an* NP*-hard problem.*

PROOF (Sketch) Consider the COMPLETE BIPARTITE SUBGRAPH problem: Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a positive integer K, does there exists a complete bipartite subgraph (biclique) of order K in G? This problem can be easily shown to be NP-complete by a reduction from the standard CLIQUE problem.

Let $G = (V_1 \cup V_2, E)$ be an instance of the NP-hard MAXIMUM COMPLETE BIPARTITE SUBGRAPH problem. We create an instance of the optimal decomposition problem as follows: Let $\mathcal{S} = (X, D, C)$ be a constraint system and let $\{X_1, X_2\}$ be a partitioning of $X = \{x_1, x_2\}$, where $X_1 = \{x_1\}$ and $X_2 = \{x_2\}$ and the domain of x_1 is $D_1 = V_1$ and the domain of x_2 is $D_2 = V_2$. C contains only one constraint R_E which consists of exactly those tuples that occur in E, that is $(v_1, v_2) \in R_E$ iff $\{v_1, v_2\} \in E$.

Finding a solution preserving optimal decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i\rangle\}_{i=1}^N$ for $(\mathcal{S}, \{X_1, X_2\})$ would imply that we have to find two subsystems $\mathcal{S}_1 = (\{x_1\}, \{V_1\}, C_1)$ and $\mathcal{S}_2 = (\{x_2\}, \{V_2\}, C_2)$, such that $C_1 \times C_2$ is a cardinality maximal subset of the tuples of R_E. Note that both C_1 and C_2 contain unary relations R_1 and R_2 respectively, where $R_1 = V'_1 \subseteq V_1$ and $R_2 = V'_2 \subseteq V_2$, respectively. Then $C_1 \times C_2$ is a cardinality maximal subset of the tuples of R_E iff $(V'_1 \cup V'_2, V'_1 \times V'_2)$ is a cardinality maximal complete bipartite subgraph of G. □

Note that this result shows that finding optimal solution preserving decompositions can be hard even in cases finding a solution preserving decomposition is easy. Hence the intimate complexity connection between finding optimal solution preserving decompositions and finding solutions for the underlying constraint system has been lost.

6 Conclusions, implications and future work

This paper considered decompositions of distributed constraint problems and studied the relationship between two well-known properties of such decompositions: solution preservation and consistency preservation. While in database applications one is interested in finding consistency preserving decompositions that allow for local updating, in artificial intelligence applications one looks for solution preserving decompositions that allow for easy composition of local solutions. In this paper, we showed that these preservation notions in decomposition are formally equivalent.

Concentrating on solution preserving decompositions, we proved that there exists an intimate connection between finding solution preserving decompositions for a given constraint system $\mathcal{S}$ and finding solutions for $\mathcal{S}$: they are computationally equally hard, neglecting polynomial differences. Finally, we discussed finding optimal decompositions and showed that this problem is NP-hard even for partitions having only two blocks. Moreover, the connections between finding optimal decompositions for a constraint system and finding solutions for it are lost.

We would like to point out the following implications:

[7] Here, we concentrate on the case that these solution sets are *finite*, e.g., by requiring the domains D_i to be finite. Note that the problem then is in $P^{\#P}$.

First of all, Hunsberger (Hunsberger 2002) showed the tractability of the decomposition method in the special case of Simple Temporal Networks (STNs); in particular he showed that there exists a polynomial algorithm for finding solution preserving decompositions. This result should not come as a surprise given the results we have shown above and the fact that finding a solution for STNs is solvable in polynomial time. Secondly, in (Brodsky, Kerschberg, and Varas 2004) it is shown that a safe decomposition can be easily found in case the constraints are linear arithmetic constraints. Again, this result is a consequence of the relationship between finding decompositions of a system $\mathcal{S}$ and finding solutions for it. Therefore, viewed in our broader perspective these two results can be seen as simple consequences of more general results.

With respect to decomposition in distributed scheduling problems, solution preserving decomposition methods like we have discussed, can be applied to enable autonomous distributed scheduling without the necessity to coordinate the integration of the solutions and to solve conflicts between the individual schedules. Our results also show that if these decompositions are strictly solution preserving, such a decomposition would also allow for adding local constraints while maintaining local consistency without endangering the feasibility of the joint schedule.

Furthermore, we should point out that the work on plan coordination by design (ter Mors et al. 2009; Buzing et al. 2006) is closely related to the current decomposition approach. This work on plan coordination allows a set of partially ordered tasks to be distributed among a set of agents in such a way that each of the agents is able to compose its own plan for the set of tasks assigned to it while guaranteeing that merging these independently constructed plans always will result in a feasible joint plan. This preservation property can be conceived as an acyclicity preservation property, since it guarantees that the joint plan always is acyclic whenever the local plans are. Instead of allowing all possible additions of constraints by the individual planning agents, the only constraints an agent is allowed to add are precedence order constraints between tasks assigned to the agent.

Concerning future work, we would like to point out that in distributed scheduling there are other important preservation properties like makespan or tardiness preservation in decompositions that can be studied. In (Yadati et al. 2008) we have made a preliminary investigation into minimal makespan preserving decompositions of scheduling problems, but a systematic investigation of the correspondence between these and other preservation properties is still lacking.

Finally, there is another interesting extension of the current approach quite similar to the work of (Boerkoel and Durfee 2010), where decomposition is restricted to local constraints and variables occurring in the global constraints might be subject to further negotiation between agents, or subject to a special decomposition approach after agents have had an opportunity to express their preferences for the values of these variables. In such a way we could make a distinction between those parts of a constraint network that can be solved by the agents independently from each other and those parts that would require some additional processing.

References

Alwan, A.; Ibrahim, H.; and Udzir, N. I. 2009. Improved integrity constraints checking in distributed databases by exploiting local checking. *Journal of Computer Science and Technology* 24(4):665–674.

Boerkoel, J., and Durfee, E. 2010. Partitioning the multiagent simple temporal problem for concurrency and privacy. *ICAPS 2010, forthcoming*.

Brodsky, A.; Kerschberg, L.; and Varas, S. 2004. Optimal constraint decomposition for distributed databases. In Maher, M. J., ed., *Advances in Computer Science - ASIAN 2004*, volume 3321 of *Lecture Notes in Computer Science*, 301–319. Springer.

Buzing, P.; ter Mors, A.; Valk, J.; and Witteveen, C. 2006. Coordinating self-interested planning agents. *Autonomous Agents and Multi-Agent Systems* 12(2):199–218.

Cohen, D. A.; Gyssens, M.; and Jeavons, P. 2006. A unifying theory of structural decompostions for the constraint satisfaction problems. In *Complexity of Constraints*. Dagstuhl Seminar Proceedings 06401.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann Publishers.

Gottlob, G.; Leone, N.; and Scarcello, F. 1999. A comparison of structural CSP decomposition methods. *Artificial Intelligence* 124:2000.

Gupta, A., and Widom, J. 1993. Local verification of global integrity constraints in distributed databases. *SIGMOD Rec.* 22(2):49–58.

Hunsberger, L. 2002. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 468–475.

Hunsberger, L. 2003. Distributing the control of a temporal network among multiple agents. *Proceedings of the second international joint conference on ...*.

Karimadini, M., and Lin, H. 2009. Synchronized Task Decomposition for Cooperative Multi-agent Systems. *ArXiv e-prints, 0911.0231K*.

Mazumbar, S., and Chrysantis, P. 2004. Localization of integrity constraints in mobile databases and specification in PRO-MOTION. *Mobile Networks and Applications* 9(5):481–490.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 161–168. New York, NY, USA: ACM.

Naanaa, W. 2009. A domain decomposition algorithm for constraint satisfaction. *J. Exp. Algorithmics* 13:1.13–1.23.

Pietrzyk, M.; Mazumdar, S.; and Cline, R. 1999. Dynamic adjustment of localized constraints. *Lecture Notes in Computer Science* 791–801.

ter Mors, A.; Yadati, C.; Witteveen, C.; and Zhang, Y. 2009. Coordination by design and the price of autonomy. *Journal of Autonomous Agents and Multi-Agent Systems* (on-line version, http://dx.doi.org/10.1007/s10458-009-9086-9).

Wah, B. W., and Chen, Y. 2006. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence* 170(3):187–231.

Yadati, C.; Witteveen, C.; Zhang, Y.; Wu, M.; and Poutr'e, H. L. 2008. Autonomous scheduling with unbounded and bounded agents. In Bergmann, R.; Lindemann, G.; Kirn, S.; and Pechoucek, M., eds., *Multiagent System Technologies. 6th German Conference, MATES 2008*, Lecture Notes In Computer Science, 195–206. Springer -Verlag.

Yokoo, M., and Hirayama, K. 1996. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the Second International Conference on Multiagent Systems*, 401–408.