Decomposing constraint systems: Equivalences and computational properties

Wiebe van der Hoek
Dept. of Computer Science
University of Liverpool,
United Kingdom
wiebe@csc.liv.ac.uk

Cees Witteveen
Dept. of Software Technology
Delft University of Technology,
The Netherlands
C.Witteveen@tudelft.nl

Michael Wooldridge
Dept. of Computer Science
University of Liverpool,
United Kingdom
mjw@liverpool.ac.uk

ABSTRACT

Distributed systems can often be modeled as a collection of distributed (system) variables whose values are constrained by a set of constraints. In distributed multi-agent systems, the set of variables occurring at a site (subsystem) is usually viewed as controllable by a local agent. This agent assigns values to the variables, and the aim is to provide distributed methods enabling a set of agents to come up with a global assignment (solution) that satisfies all the constraints. Alternatively, the system might be understood as a distributed database. Here, the focus is on ensuring consistency of the global system if local constraints (the distributed parts of the database) change. In this setting, the aim is to determine whether the existence of a global solution can be guaranteed. In other settings (e.g., P2P systems, sensor networks), the values of the variables might be completely out of control of the individual systems, and the constraints only characterize globally normal states or behavior of the system. In order to detect anomalies, one specifies distributed methods that can efficiently indicate violations of such constraints. The aim of this paper is to show that the following three main problems identified in these research areas are in fact identical: (*i*) the problem of ensuring that independent agents come up with a global solution; (*ii*) the problem of ensuring that global consistency is maintained if local constraint stores change; and (*iii*) the problem of ensuring that global violations can be detected by local nodes. This claim is made precise by developing a decomposition framework for distributed constraint systems and then extracting *preservation properties* that must satisfied in order to solve the above mentioned problems. Although satisfying the preservation properties seems to require different decomposition modes, our results demonstrate that in fact these decomposition properties are equivalent, thereby showing that the three main problems identified above are identical. We then show that the complexity of finding such decompositions is polynomially related to finding solutions for the original constraint system, which explains the popularity of decomposition applied to tractable constraint systems. Finally, we address the problem of finding optimal decompositions and show that even for tractable constraint systems, this problem is hard.

Cite as: Decomposing constraint systems, Wiebe van der Hoek, Cees Witteveen and Michael Wooldridge, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 149-156.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.4 [**Knowledge representation formalisms and methods**]

General Terms

Theory

Keywords

distributed constraint systems, decomposition, complexity

1. INTRODUCTION

Distributed systems can frequently be modelled as a collection of distributed (system) variables X, whose values are constrained by a set of constraints. Usually, one distinguishes a set of sites, each of which contains a disjoint subset of variables $X_i \subseteq X$. Each site is responsible for those constraints relating to the variables that occur in its variable set X_i (its set of *local constraints*). Sites must interact with respect to the set of *global constraints*[1], which relates variables of different components X_i. The way such a model of a distributed system is used largely depends on the application domain. We can identify at least three different application domains: multi-agent systems, distributed databases, and P2P systems.

In multi-agent systems, one typically assumes that there is a set of agents, with each agent A_i controlling the variables in X_i. It is the common task of all the agents to assign suitable values to their variables such that all constraints are satisfied. Since each agent tries to assign values to the variables independently from the others, only the constraints whose variables occur in the agent's control set are guaranteed to be satisfied. The main research problem is then to provide (distributive) methods enabling the agents to come up with a global assignment (solution) satisfying all constraints – both local and global.

In other domains, such as databases, such a distributed constraint system is conceived as a model of a distributed database. Here, the focus is not on finding solutions for a fixed set of constraints, but to ensure consistency of the

[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.

global system in the event that local constraints (the distributed components of the database) change. Such a notion of consistency is expressed by global integrity constraints that need to be respected whatever changes take place at local sites. Hence, here the issue is not giving values to variables such that all constraints are satisfied, but to ensure that there exists at least one possibility for such a globally satisfying assignment.

While in the above mentioned areas the focus is on finding a solution or guaranteeing the existence of at least one solution, in areas such as peer-to-peer (P2P) systems and sensor networks, modeling by distributed constraint systems is often focused on the detection of constraint *violations*. Here, the set of constraints is used to characterize acceptable states or behavior of the global system, and violations of such constraints indicate potentially problematic anomalies (e.g., a DDoS attack). In order to detect such anomalies, one specifies a set of distributed methods that aim to establish violations of such constraints as efficiently as possible.

Although these problems arise in different areas they all have a common aspect. From an abstract point of view, one might consider the set of agents (sites, local databases) with their constraints as a *decomposition* of the original constraint system induced by a partitioning of the variables. In all three areas mentioned above, the problem is how we might ensure this decomposition to have some *preservation properties* with respect to the underlying global system.

From a multi-agent systems point of view, in distributed constraint systems one often considers the local agent as autonomous. Here, 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 multi-agent systems 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 [10] decomposition[2] has been applied to ensure that independently chosen schedules for subnetworks of a Simple Temporal Network (STN) can always be merged to a joint schedule of the total network. In [11] 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.

In the database community, one wants to ensure that whenever each local database is consistent, the consistency of the global database is implied, whatever changes occur locally. The method applied here is to provide *localized versions* of global integrity constraints that ensure that, whatever local information satisfying these constraints is added to the (distributed) database, the global consistency of the total database will be preserved [2, 9, 4]. Hence the database community is interested in *consistency preserving* decompositions[3].

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

[3]Quite closely related to the database community, in the sensor network community, one distinguishes the *localization* problem, where a distributed constraint is reformulated into local constraints for mobile entities and is adjusted dynamically [12, 15]. The satisfaction of the distributed constraint is guaranteed whenever all the local constraints are satisfied.

The P2P community aims at the efficient detection of constraint *violations*. Here, normal operations are specified by a global constraint C_S. For reasons of efficiency, one prefers not to monitor all sites to establish violations of these constraints. Therefore, localizations of such constraints are provided to each node, such that it has its own violation detection mechanism [1]. A global violation detection mechanism is triggered only if some local node detects a violation, thus saving communication between the nodes. So, in the P2P community, one is interested in *safety preserving* decompositions of integrity constraints, ensuring that whenever all local states indicate safeness (no violation detection occurs) of their local states, the global state is safe (that is, the global integrity constraints are all respected), too.

In summary, it seems that we can study all three problems in a common decomposition framework, where the only difference between these problems is in their preservation properties. In fact, as, we pointed out, there has been extensive research in these three areas, focusing on either the solution preserving, the consistency preserving, or the safety preserving aspect of decompositions in distributed constraint systems. However, to the best of our knowledge, there have been no attempts to establish their equivalence. In short, the aim of the present paper is to address this issue and to investigate some computational aspects of such decompositions.

We begin in Section 2 by presenting a formal framework for investigating these properties, and present the technical preliminaries used in the remainder of the paper. In Section 3, we investigate the relationships between the properties we identified, i.e., we consider whether the properties are independent from each other, whether they imply each other, or whether they are they completely identical. In Section 4, we address some computational aspects of these preservation properties — for example, how difficult is it to find a {solution, consistency, violation}-preserving decomposition. We will establish some tight computational connections between these problems and the general problem of finding a solution to a constraint system. Then, in Section 5, we will address the problem of *information loss* inherent in solving a decomposed problem as opposed to solving the problem at a global level. We will indicate that in general the problem of establishing the exact information loss is intractable. 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 distributed constraint systems.

2.1 (Distributed) 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 of some language. We will not require any specific language for constraints $c \in C$, but it is useful to assume some fundamental properties. Specifically, we will assume the language contains constants for elements in the domains D_i, the usual Boolean connectives $(\neg, \vee, \wedge)$, and equality. We also require that it is possible to determine whether a solution satisfies a constraint in polynomial time. Formally,

a solution σ of the system $\mathcal{S} = \langle X, D, C\rangle$ is an assignment $\sigma = \{x_i \leftarrow d_i\}_{i=1}^n$ to all variables in X such that $d_i \in D_i$, and each constraint $c \in C$ is satisfied. We sometimes write $\sigma \models C$ to mean that σ is a solution to C. Given a subset $X_i \subseteq X$, we let σ_{X_i} denote the restriction of σ to the subset X_i. Where no confusion is possible, we will use σ_i as a shorthand for σ_{X_i}. The set of all assignments σ is denoted by Σ. Likewise, the set of all assignments for a subset $X_i \subseteq X$ is denoted by Σ_{X_i} or Σ_i.

The set of solutions σ to system $\mathcal{S} = \langle X, D, C\rangle$ will be denoted by $Sol(\mathcal{S})$. $\mathcal{S}$ is called *consistent* if $Sol(\mathcal{S}) \neq \emptyset$. For every $c \in C$, let $Var(c)$ denote the set of variables mentioned in c. For a set of constraints C, we let

$$Var(C) = \bigcup_{c \in C} Var(c).$$

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'.

In this paper we consider constraint systems $\mathcal{S}$ that are *distributed* [19]; 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 all $1 \leq i < j \leq N$, $X_i \cap X_j = \emptyset$.

To indicate that a constraint system $\mathcal{S} = \langle X, D, C\rangle$ is distributed by a partitioning $\{X_i\}_{i=1}^N$ of X, we write $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ and call it a *distributed constraint system*. We are particularly interested in those distributed systems $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ 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*[4]. We say that σ is a solution of $\mathcal{S}'$ if, for each i, σ_i is a solution of $\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle$. The decomposition $\mathcal{S}'$ is said to be consistent if $\bigcup_i C_i'$ is consistent.

Whenever $\{X_i\}_{i=1}^N$ is a partitioning of X, we will write $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \ldots \sqcup \sigma_N$ to indicate an assignment σ that is composed of the disjoint assignments σ_i for X_i. Likewise, we will write $Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2) \sqcup \ldots \sqcup Sol(\mathcal{S}_N)$ to indicate the set of global assignments σ that can be constructed by simply composing all local solutions of the subsystems $\mathcal{S}_i$.

2.2 Decompositions: preservation properties

We now discuss three specifications of the relationship between a distributed constraint system $\mathcal{S}$ and a decomposition $\mathcal{S}'$, with respect to the three preservation properties we informally discussed above.

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

2.2.1 Solution preserving decompositions

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

DEFINITION 1. *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ 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. $\mathcal{S}$ if it satisfies the following property:

$$\emptyset \subseteq Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2) \sqcup \ldots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S}).$$

$\mathcal{S}'$ is said to be strictly solution preserving *if the first inclusion is strict whenever $Sol(\mathcal{S}) \neq \emptyset$.*[5]

EXAMPLE 1. *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ 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 $X = \{x_1, x_2, x_3, x_4\}$ and 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 $Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2)$ is a solution to $\mathcal{S}$, because x_1 as well as x_2 is assigned to 1 in any merge, thereby satisfying C. 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 by simply merging local solutions σ_i.

2.2.2 Consistency preserving decompositions

In distributed database applications, one typically 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 implies the satisfaction of the global constraints. To simplify the discussion, we focus 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 plus locally added constraints is consistent, whenever the added information does not cause any local inconsistency. We need the following definition.

DEFINITION 2. *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ be a distributed constraint system. An extension of $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ is a constraint system $\mathcal{S}^E = \langle\{X_i\}_{i=1}^N, D, C'\rangle$ where $C \subseteq C'$.*

[5]This last condition is needed to take care for inconsistent constraint systems.

An extension $\mathcal{S}^E$ captures the idea of a constraint system $\mathcal{S}$ to which a set constraints has been added. Suppose that we have a decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ and suppose that to each local system $\mathcal{S}_i$ a set $C_i'' \setminus C_i'$ of constraints is added such that for each $\mathcal{S}_i$ we obtain an extension $\mathcal{S}_i^E$. Then consistency preservation requires that local consistency implies global consistency. That is, if the locally added information $C_i'' \setminus C_i'$ does not render any resulting extension $\mathcal{S}_i^E$ inconsistent, the total information $(C_1'' \setminus C_1') \cup \ldots \cup (C_N'' \setminus C_N')$ added to the distributed system should not render the total system inconsistent, i.e., $\mathcal{S}^E = \langle X, D, C \cup (C_1'' \setminus C_1') \cup \ldots \cup (C_N'' \setminus C_N')\rangle$ should be consistent as well.

Definition 3 (consistency preserving extensions). *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ be a distributed constraint system. 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. $\mathcal{S}$ if the following condition holds: whenever, for all $i = 1, 2, \ldots, N$, the extensions*

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

of $\mathcal{S}_i$ are consistent, the global extension

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

is consistent as well. $\mathcal{S}'$ 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 distributed constraint system specified in Example 1 and its decomposition $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$. We show that $\mathcal{S}'$ is also a strictly consistency preserving decomposition w.r.t. $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$: 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 $\mathcal{S}_i$. Since $\mathcal{S}_1$ has a unique solution, every consistent extension $\mathcal{S}_1^E$ must have the same unique solution $\sigma_1 = \{x_1 \leftarrow 1, x_2 \leftarrow 1\}$. Whatever solution σ_2 is chosen for a consistent $\mathcal{S}_2^E$)), it is always a solution to $\mathcal{S}_2$, too. But then, using the solution preservation property, $\sigma = \sigma_1 \sqcup \sigma_2 \models C$. Moreover, $\sigma_1 \models C_1''$ and $\sigma_2 \models C_2''$, therefore*

$$\sigma \models C \cup (C_1'' \setminus C_1') \cup (C_2'' \setminus C_2').$$

Hence, $\mathcal{S}^E$ is consistent and the decomposition is strictly consistency preserving.

2.2.3 Safety preserving decompositions

In areas such as P2P systems and sensor networks, one uses global constraints on the values by variables indicating vital system properties or to characterize the normal behavior of a system. As long as these global constraints are satisfied, no active control of the system is necessary. Only if violations of these global constraints occur, actions have to be performed to restore a normal state. In order to avoid excessive communication between the sites, one prefers to detect such anomalies in a distributed way. That is, the global constraints need to be localized in such a way that each site can establish independently from the others whether or not its local set of constraints is violated. Such a detection mechanism should be *safe* in the sense that whenever there is a global violation, at least one site should have detected it. But this, by contraposition, immediately implies that safeness also can be expressed as a preservation property: whenever each local site concludes that its local set of constraints is safe, the global set of constraints should be safe, too.

Definition 4 (safety preserving decompositions). *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ be a distributed constraint system. A decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is called* safety preserving *w.r.t. $\mathcal{S}$ if the following condition holds: whenever there exists a global system state (assignment) σ such that for all $i = 1, 2, \ldots, N$, $\mathcal{S}_i$ is locally safe, i.e., $\sigma_i \models C_i'$, then the global system is safe, too, i.e., $\sigma \models C$.*

Example 3. *Take $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ of Example 1. The decomposition $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$ is obviously safety preserving w.r.t. $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$: whenever there is a state σ such that $\sigma_1 \models (x_1 \wedge x_2)$ and $\sigma_2 \models true$, we must have that $\sigma \models (x_1 \wedge x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_4)$.*

3. DECOMPOSITION PROPERTIES: RELATIONSHIPS

Given the three 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? We start with the easiest one. As the reader might have noticed, there is an obvious relationship between solution preserving decompositions and safety preserving decompositions: A decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is solution preserving exactly when it is safety preserving with respect to a given distributed system $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$:

Proposition 1. *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ be a distributed constraint system. Then $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i'\rangle\}_{i=1}^N$ is solution preserving w.r.t. $\mathcal{S}$ iff $\mathcal{S}'$ is also safety preserving w.r.t. $\mathcal{S}$.*

Proof. Notice that a decomposition $\mathcal{S}'$ that is safety preserving exactly if for all $\sigma = \sigma_1 \sqcup \ldots \sqcup \sigma_N \in \Sigma$ it holds that $\forall i = 1, \ldots, N$ $\sigma_i \in Sol(\mathcal{S}_i)$ implies $\sigma \in Sol(\mathcal{S})$.

Hence, $\mathcal{S}'$ is safety preserving iff

$$\emptyset \neq Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2) \sqcup \ldots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$$

iff $\mathcal{S}'$ is solution preserving. □

With respect to consistency preserving and solution preserving decompositions, intuitively, it should be easy to show that solution preservation subsumes consistency preservation: if it is ensured that updates to a *local* constraint store ensure locally consistent stores, there exist local solutions σ_i for every updated local store. In particular, these solutions are solutions for the initial versions of the local stores. Hence, by solution preservation, merging these solutions constitutes a solution σ for the global (initial) store. But then it is easy to show that σ satisfies all the local updates as well. Hence, the global constraint store plus the added constraints is a consistent set as well. More precisely:

Proposition 2. *Let $\mathcal{S} = \langle\{X_i\}_{i=1}^N, D, C\rangle$ 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. $\mathcal{S}$, then $\mathcal{S}'$ is also consistency preserving w.r.t. $\mathcal{S}$.*

PROOF. Assume $\mathcal{S}'$ to be solution preserving w.r.t. $\mathcal{S}$. For $i = 1, 2, \ldots, N$, consider arbitrary (consistent) extensions $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$ of the local subsystems $\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle$. For each subsystem $\mathcal{S}_i^E$, select an arbitrary assignment $\sigma_i \in Sol(\mathcal{S}_i^E)$.

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

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

Hence, by solution preservation, the assignment $\sigma = \sigma_1 \sqcup \ldots \sqcup \sigma_N$ will satisfy $\mathcal{S}$. Therefore,

$$\sigma \models C \tag{1}$$

By definition of σ_i, $\sigma_i \models C_i'' \setminus C_i'$. Moreover, every $C_i'' \setminus C_i'$ is a set of variable disjoint constraints over X_i. Hence, it follows that

$$\sigma \models (C_1'' \setminus C_1') \cup (C_2'' \setminus C_2') \cup \ldots (C_N'' \setminus C_N') \tag{2}$$

Hence, by equation (1) and (2),

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

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

Although consistency preservation might seem to be a weaker property, somewhat 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. More formally:

PROPOSITION 3. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ 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. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, then $\mathcal{S}'$ is also solution preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$.*

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. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. By assumption, for every subsystem $\mathcal{S}_i$ and every extension $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$ of $\mathcal{S}_i$, it must hold that, whenever the extended local systems $\mathcal{S}_i^E$ are consistent, then the global extended system

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

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 $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$, 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 $i = 1, 2, \ldots, N$, $\mathcal{S}_i^E$ is consistent and each σ_i is the unique solution of $\mathcal{S}_i^E$.

By consistency preservation, the extension

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

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

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

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

As an easy consequence of these propositions we have the following result:

THEOREM 1. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. Then $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$ is solution preserving w.r.t. $\mathcal{S}$ iff $\mathcal{S}'$ is safety preserving w.r.t. $\mathcal{S}$ iff $\mathcal{S}'$ is consistency preserving w.r.t. $\mathcal{S}'$.*

It is not difficult to show that these equivalences also hold for the strictly preserving versions. This immediately implies that all results that have been obtained for consistency preserving decompositions such as occur in [4, 12] can be used for solution preserving approaches to decomposition as well.

4. FINDING SOLUTION PRESERVING DECOMPOSITIONS

The equivalence between the three preservation properties of 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 equivalences, 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 [4] 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. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ 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 $\mathcal{S}$.[6]

We start by defining the notion of a safe decomposition:

DEFINITION 5 ([4]). *Given a distributed constraint system $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, 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. $\mathcal{S}$ if

$$\bigcup_{i=1}^N C_i' \models C.$$

Note that this property is also sometimes known as the *covering property* [1] and should not be confused with the *safety preservation* property we discussed in the previous section.

PROPOSITION 4. *The decomposed system*

$$\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$$

is safe w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ iff $\mathcal{S}'$ is solution preserving w.r.t. $\mathcal{S}$.

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

PROOF SKETCH. Assume that $\mathcal{S}'$ is solution preserving w.r.t. $\mathcal{S}$. Then $Sol(\mathcal{S}_1) \sqcup \ldots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$. 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 $\sigma_i \models C'_i$ 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. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. 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,

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

and $\mathcal{S}'$ is solution preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. □

Using this notion of a safe decomposition, we can show that the problem of deciding whether a decomposition $\mathcal{S}'$ is safe w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ is a coNP-complete problem:

PROPOSITION 5. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ 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. $\mathcal{S}$ 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 6. *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. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$.*

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.[7] 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 $\left|\bigcup_{i=1}^{N} C'_i\right|$ 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 known that such constraint systems are solvable in polynomial time [6]. 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. □

Using this result and Theorem 1, we now may conclude:

THEOREM 2. *Finding a strictly {consistency, solution, safety}-preserving decomposition $\mathcal{S}'$ for a distributed constraint system $\mathcal{S}$ is, neglecting polynomial-time differences, as hard as finding a solution for $\mathcal{S}$.*

It is well-known that for general constraint systems, finding a solution is NP-hard [7]. 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 [10] or linear arithmetic constraints [4].

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. For example, taking a safety preserving decomposition, one would like to minimize false alarms, i.e., one would minimize those events where a local constraint is violated, but the global integrity constraint would still be satisfied.

Hence, the information loss due to the decomposition $\mathcal{S} = \{\mathcal{S}_i = \langle X_i, D_i, C_i \rangle\}_{i=1}^N$ can be defined as
$Sol(\mathcal{S}) \setminus (Sol(\mathcal{S}_1) \sqcup \ldots \sqcup Sol(\mathcal{S}_N))$: the set of solutions of the original system that cannot be obtained by merging the local solutions using the decomposition.

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

Therefore, given a distributed constraint system $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, 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 $|Sol(\mathcal{S}) \setminus Sol(\mathcal{S}')|$ where $Sol(\mathcal{S}') = Sol(\mathcal{S}_1) \sqcup \ldots \sqcup Sol(\mathcal{S}_N)$ is the set of solutions obtainable from the decomposed system.[8]

This optimality problem can be easily shown to be intractable, even if the underlying constraint system contains two variables and one (binary) constraint[9], and finding a solution preserving decomposition is trivial:

PROPOSITION 7. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ 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 (bi-clique) 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 & 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 multi-agent systems applications, one looks for solution preserving decompositions that allow for easy composition of local solutions. In this paper, we showed formally that these preservation notions in decomposition are 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: First of all, Hunsberger [10] 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 [4] 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 this broader perspective, these two results can be seen as consequences of more general results.

With respect to decomposition in distributed scheduling problems, solution preserving decomposition methods of the type 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 [16, 5] 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.

Note that there are other views on decomposition in constraint systems, as expressed by *structural decomposition methods* [8, 17, 6, 14] and by the *distributed constraint optimization (DCOP) approach* [19, 13]. 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 vari-

[8]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 $P^{\#P}$ and hardness for this class is still open.

[9]A binary constraint is a constraint in which only two variables do occur

ables 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.

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 [18] 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.

Furthermore, we should investigate the idea of stratified decomposition in AI applications where first a solution preserving decomposition of the first layer of their constraints can be provided, the agents submit their own preferred solutions and conditional on these solutions the decomposition of the next layer is provided, etc. This would allow for some kind of synchronisations, for example, when exactly one of two variables needs to be true, but each are owned by different agents (this is a very common problem when assigning duties or tasks to agents).

Finally, there is another interesting extension of the current approach quite similar to the work of [3], 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.

7. REFERENCES

[1] S. Agrawal, S. Deb, K. V. M. Naidu, and R. Rastogi. Efficient detection of distributed constraint violations. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, April 15-20, 2007, Istanbul, Turkey*, pages 1320–1324. IEEE, 2007.

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

[3] J.C. Boerkoel and E.H. Durfee. Partitioning the multiagent simple temporal problem for concurrency and privacy. In R. I. Brafman, H. Geffner, J. Hoffmann, and H.A. Kautz, editors, *Proceedings of the 29th International Conference on Automated Planning and Scheduling, ICAPS 2010*, pages 26–33. AAAI, 2010.

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

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

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

[7] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

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

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

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

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

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

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

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

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

[16] A.W. ter Mors, C. Yadati, C. Witteveen, and Y. Zhang. 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), 2009.

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

[18] C. Yadati, C. Witteveen, Y. Zhang, M. Wu, and H. La Poutré. Autonomous scheduling with unbounded and bounded agents. In Ralph Bergmann, Gabriela Lindemann, Stefan Kirn, and Michal Pechoucek, editors, *Multiagent System Technologies. 6th German Conference, MATES 2008*, Lecture Notes In Computer Science, pages 195–206. Springer -Verlag, 2008.

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