

Constraint Satisfaction Techniques for Negotiating Agents

Monique Calisti and Boi Faltings

Whitestein Technologies AG
Gotthardstrasse 50, 8002 Zurich, Switzerland
★ Laboratoire d'Intelligence Artificielle
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
mca@whitestein.com, boi.faltings@epfl.ch

ABSTRACT

This paper discusses the deployment of constraint satisfaction methods as a way of enforcing the coordination of distinct and self-interested agents in open and dynamic environments. We report back on our experience in deploying such approach for developing an agent based solution for resource allocation in multi-provider communication networks. This makes it possible to draw some conclusions on the main benefits and challenges for an effective integration of constraint satisfaction techniques in multi-agent systems.

1. INTRODUCTION

Nowadays, coordination between distributed and eventually self-interested software entities populating electronic systems represents a crucial aspect for many applications requiring flexible interactions of distinct and possibly heterogeneous components. This is particularly critical in agent-based systems where the conceptual approach behind any kind of solution design and development strongly relies upon the notion of *interaction* (or social behaviour) of autonomous processes that dynamically coordinate their actions by communicating with each other. In this paper, we argue that constraint satisfaction techniques, including both the problem modelling approach and specific solving techniques (i.e., algorithms), have the potential to be deployed in multi-agent systems at two main distinct levels:

- At the intra-agent architecture level: the Constraint Satisfaction Problem (CSP) formalism can become the support of the reasoning mechanisms that agents rely upon for decision making and future behaviour, actions and strategy choice.
- At the inter-agent coordination level: sophisticated interactions can be modelled as distributed CSPs in which autonomous agents have to make specific choices about which possible message/action they will undertake within an ongoing conversation.

Furthermore, many existing problems that require finding a consistent combination of distinct agent actions in various domains can be formalised in a natural way as a distributed

CSP. Some significant examples include resource allocation (e.g., in communication networks [31], [30]), scheduling (e.g., job scheduling [26]), truth maintenance [11], interpretation [20], etc.

This paper has the twofold objective of:

- Describing the key features of a constraint satisfaction based approach that can be used in a multi-agent system (MAS) to design and thereby implement software agent decision making and dynamic interactions.
- Discussing the practical experience of deploying such a distributed CSP based approach when developing an agent-based solution for the service provisioning in multi-provider communication networks (namely, the *Multi-provider Service Setup* (MuSS) problem).

In the next section, we shortly review background concepts about the coordination of self-interested entities in multi-agent systems. Section 3 describes the main concepts behind a constraint satisfaction based approach for modelling the decision making process of individual agents and for enforcing coordinated interactions. This approach has been directly deployed for solving the MuSS problem, which is formally defined and formulated in Section 4. The *Network Provider Interworking* (NPI) system is the proposed solution to this problem (see Section 5). In this open and distributed framework, autonomous agents acting on behalf of different network operators coordinate their actions by means of distributed constraint satisfaction techniques integrated with economic mechanisms for automated negotiations (see Section 6). The experience acquired in developing the NPI system leads to discuss its main features and draw some connections with other relevant frameworks (Section 7), before finally concluding the paper.

2. BACKGROUND

Until quite recently, the CSP and the MAS communities have been traditionally working with little direct coordination. However, this has been changing for a number of reasons:

- *A distributed CSP based approach can provide an effective infrastructure for agents coordination.* Many problems that require finding a consistent combination of agent actions can be naturally formalised as distributed CSPs. In this way, constraint satisfaction techniques can be deployed to provide methods for achieving coherence and consistency among distinct agents [29], [10].
- *A larger set of efficient CSP solving techniques (distributed and non distributed) is currently available.* The existence of a number of effective and generic algorithms that have the potential to be adapted and/or re-used for solving many hard specific problems facilitates the adoption of this technology [32].
- *Various successful stories are increasing the interest toward and the confidence in the CSP (distributed and non distributed) paradigm.* The fact that several works have successfully adopted a CSP oriented approach (see Section 7.1) to define solutions for a wide range of distributed real world problems is likely to build confidence in this approach and allow more wide-spread experimentation.

This paper does not aim to provide a complete survey of frameworks that have been integrating the distributed CSP technology within multi-agent systems. Instead, the aim is to discuss the aspects and the requirements, which characterise and balance the traditionally distinct CSP and MAS viewpoints by proposing a unified approach that has been deployed for implementing the NPI system.

2.1. THE AGENT-ORIENTED APPROACH

Today, agent technology represents one of the most dynamic fields in which various techniques are used to build distributed systems with intelligent local components designed to both cooperate and coordinate their activities. Agents receive inputs through sensors and they act on the environment through effectors. Moreover, they exhibit autonomous, reactive, opportunistic and goal-directed behaviour (i.e., the action to perform has to be consistent with the agent's goals). This approach has been adopted for dealing with many tasks in different domains, such as resource allocation, network management, e-commerce, health care, etc.¹, for its intrinsic capability of representing the decentralised nature of many problems, the existence of multiple control and logic components with distinct roles, the multiple capabilities/roles of distributed and eventually self-interested entities. However, for an effective use of this technology and for exploiting typical agents' characteristics such as social ability, responsiveness and pro-activity, a key aspect is the capability of supporting *coordinated* interactions.

¹See [12] for a good review of various multi-agent based application areas.

2.2. NEGOTIATION FOR COORDINATION IN MULTI-AGENT SYSTEMS

Research in multi-agent systems is primarily concerned with the *coordination* of autonomous (possibly heterogeneous) and self-motivated computational agents. The main assumption is that there is no global control, no globally consistent knowledge and no global goals. Therefore, self-interested agents choose the best strategy for themselves, which cannot be explicitly imposed from outside. In the NPI context, agent coordination is achieved by combining distributed constraint satisfaction techniques (such as arc consistency, see Section 5) with automated negotiations.

Negotiation can be considered as a process by which a joint decision is made by two or more parties which first verbalise contradictory demands and then move toward agreement by a process of concession making or searching for new alternatives [22]. Typically, each party starts a negotiation process by offering the most preferred solution from an individual perspective. If the offer is not accepted by other parties, then a counter-offer can be made in order to converge towards an agreement. During this process, the set of available options for each agent decreases until an agreement is or is not reached. When defining an *automated* negotiation system it is fundamental to choose the possible negotiation protocol(s) to design appropriate negotiation strategies. A negotiation protocol specifies the 'rules of encounter' between negotiation participants, which means the possible roles that participants can cover, what deals can be valid and the permissible sequences of actions, i.e., which messages are allowed and in which order. A negotiation strategy establishes the way an agent behaves in an interaction, i.e., what specific deals will be proposed during the negotiation. Given a pre-fixed protocol, there may exist several compatible strategies, each of which produces a very different outcome.

On one hand, negotiation can be considered as a technique for solving distributed CSP [18]. On the other hand, as discussed in the next section, negotiating agents can make use of CSP methods in order to: (1) decide about their strategic choices (i.e., intra-agent decision making process), (2) model the ongoing interactions in which they are involved in without specifying the selected interaction protocol a priori, and (3) enforce coordinated interactions without the need of revealing strategic information (by using, for instance, consistency methods).

3. A CSP BASED APPROACH FOR NEGOTIATING AGENTS

Constraint satisfaction is a powerful AI paradigm which proposes techniques to find assignments for problem variables subject to constraints on which only certain combinations of values are acceptable. A distributed CSP is a CSP in which the variables and constraints are distributed among distinct autonomous agents like in multi-agent systems. Here, each agent has one or multiple variables and tries to determine its/their possible value/s. At the intra-

agent architecture level, therefore, the CSP formalism can become the support of the reasoning mechanisms that agents rely upon for decision making and future behaviour, actions and strategy choice. In general, there exist intra- and inter-agent constraints (i.e., constraints between variables controlled by distinct agents) and the value assignment must satisfy all these constraints. In order to verify constraints involving external agents, some form of communication needs to be supported. The most known and deployed communication model, which the distributed CSP paradigm relies upon, assumes that agents communicate by sending messages, the delay in delivering messages is finite, though random, and messages are received in the order in which they are sent [29]. This corresponds to a multi-agent system where the various interacting entities adopt a common communication stack (e.g., a common agent communication language (ACL), a shared content language (CL) and ontologies).

3.1. A CSP-BASED DECISION MAKING MODEL

Every agent involved in a negotiation process represents a distinct decision maker that can receive and/or make offers. This can be expressed as a choice problem in the following way:

- *Variables* are choices to be made (such as which resources should be used in the network for the allocation of a service demand, which price should be asked, etc.). The set of variables is the set of choices which need to be made in order to converge to an agreement.
- *Domains* are the available sets of options for each choice. For instance, the selected network resources can only be the ones guaranteeing enough capacity to satisfy the required Quality of Service (QoS).
- *Constraints* represent the existing relationships between choices which express valid or invalid combinations (the specific selected resources are likely to be dependent on the QoS that has to be provided). The set of constraints therefore restricts the set of all possible combinations of choices made to a smaller set of desirable assignments which meet the requirements of a solution to the choice problem.

Whenever an agent has to make an offer, it assigns a specific value for every variable that the offer consists of (if there are more than one), by taking into account the existing constraints on that variable/s. The set of all possible combinations of assignments of domain values to the variables defines the negotiation space. The way constraints are evaluated and the specific choice is made represent the agent's negotiation strategy. Similarly, whenever an agent receives an offer its decision making process can be modelled as a CSP. The receiver agent can decide whether the offer is acceptable or not by mapping the offer characteristics into a set of variables that have to satisfy specific requirements (constraints). For instance, one variable could correspond to the price of the required service, and another variable could correspond

to the offered QoS, etc. Only if all the constraints existing on the variables corresponding to the received offer are satisfied, the agent will consider the offer acceptable.

3.2. A DISTRIBUTED CSP-BASED MODEL FOR AGENTS INTERACTIONS

In order to achieve their individual objectives, dynamically coordinate their actions and converge to an agreement, software agents need to dynamically interact with each other. Usually, ongoing agent conversations need to fall into typical patterns. This means that certain message sequences are expected, and, at any point in the conversation, other messages are expected to follow. These typical patterns of message exchange are called *interaction protocols*. In general, these protocols are assumed to be known by all the interacting entities (i.e., transmitter and receiver/s) either *implicitly* or *explicitly*. In the first case, human designers code common procedures and conventions, interpreting them and writing appropriate systems behaviour (i.e., the semantics providing the meaning and the context for a specific interaction is assumed to be implicitly known by the agents). In the latter case, the formal semantics behind the specific interaction protocol has to be encoded into a particular formalism and the whole interaction and each message have to be evaluated as a single expression. This is very difficult to achieve in open systems, especially for the heterogeneity of world models, which are often very domain dependent.

The main idea behind the deployment of a distributed CSP-based approach for modelling agent interactions is the capability of enabling coordinated behaviour without any a priori assumption about the specific sequence of messages to be exchanged. This relies upon the following model:

- All interacting entities make use of a common communication stack including the selected ACL, the deployed CL and the referred ontologies (i.e., common world's view).
- For any given interaction, every agent i is a decision maker controlling a variable m_i that represents the next message agent i will exchange within the ongoing conversation.
- The domain D_i for every variable m_i is the set of possible ACL performatives that the agent can make use of for communicating with external entities.
- The set C_i of constraints on the possible values that m_i can take is given by the semantics rules upon which the deployed ACL is built. Depending indeed on what the object of the message is (i.e., the state of the world that the agent wants to communicate), the type of message (or ACL performative) that can be used is bound by the ACL semantics that the interacting agents refer to.

For a better understanding, let us assume two negotiating agents A and B making use of the same ACL: agent A represents an end user asking agent B for the allocation of a specific service demand d . Agent A has the option of either (1) requesting agent B to perform the action of making

an offer for d , i.e., $m_a = request$, or (2) calling for a proposal for the service demand d , i.e., $m_b = cfp$ ². Therefore, the domain for m_a is given by $D_a = \{request, cfp\}$. Let us consider that agent A sends a call for proposal, i.e., $m_a = cfp$ (*allocation (d)*). When agent B processes m_a , a decision about what action will be undertaken as a reaction to the received call for proposal needs to be made. This choice can be influenced (i.e., constrained) by a number of elements which include the agent's mental state, the characteristics of the service demand d , the current utilisation of the network resources agent B controls, cost reasons, etc. Once the decision of 'what to do' has been made, agent B can inform agent A . The set of possible values D_b that the message m_b can take is determined by considering first what the agent B decides to do (for instance, either it accepts to make a proposal or it refuses, e.g., $D_b = \{agree, inform, confirm, refuse, not-understood\}$). Then, agent B decides how this can be communicated, i.e., which specific communicative act can be used (or which value m_b can take in D_b) based upon the ACL semantics.

This kind of approach has two main potential benefits:

- Since the semantics behind the choice of specific ACL performatives is assumed to be shared by all interacting agents, it is possible for the sender X of a message m_x to evaluate what possible message m_y can be received from agent Y as answer, without having to specify a given interaction protocol. This can be done by pruning out from the finite set of existing ACL performatives, the ones satisfying the semantics requirements implied on m_y by the value selected for m_x .
- It is possible to dynamically shape the interaction with external entities (and therefore enforce coordination), by exchanging constraints on the kind of messages that can be used within an ongoing conversation.

4. THE MUSS PROBLEM

In its most simple form, the MuSS problem is an optimisation (allocation) problem with assignment constraints. These constraints determine and influence the way a fixed amount of resources is allocated to a given number of activities needed to support the service in the most effective way. In the multi-provider context, both the increasing end user QoS requirements and the distribution of resources managed by distinct entities introduce challenging issues and additional constraints that make service allocation even more complex. This problem can be naturally modelled as a distributed CSP since the variables (local network resources such as bandwidth, routers, etc.) are distributed among agents and since constraints exist among them. More precisely, end-to-end routes are decomposed into fragments (i.e., distinct variables) corresponding to independent decision makers. In distributed CSP terms, there is one variable per provider whose values are route fragments through

²These two communicative acts are assumed to be equally feasible given the adopted CL and ontologies.

that provider. Constraints between the variables ensure that route fragments connect and specific CSP methods are applied in order to rapidly access what choices can be part of a consistent end-to-end route. The space of possible solutions provides the basis for subsequent negotiations. More than one local route at a time can be negotiated since a single variable can take as value one of a large set of end-to-end routes. Decisions on the preferences of path fragments are made by finally pruning out possible values from this set. Some preliminary assumptions about the communication model which the NPI paradigm relies upon are given in the following.

Assumption 1. There is at least one agent for every provider network. This agent is called *Network Provider Agent (NPA)*.

Assumption 2. Agents communicate using messages. Messages are encoded in a standard agent communication language, namely FIPA-ACL³ [28].

Assumption 3. The delay in delivering messages is finite, though random.

Assumption 4. In CSP terms, every provider owns and controls exactly one variable.

For large networks different variables could be handled by a hierarchy of coordinated agents controlling distinct parts of every network. Even in that case, the NPI paradigm would still be valid. However, Assumption 4 simplifies the description of the main mechanisms presented in this paper.

4.1. THE MUSS CONSTRAINT GRAPH

A distributed CSP is usually represented as a *constraint graph*, where variables are vertices and constraints are edges between vertices. It is fundamental to underline that this constraint graph is not the physical communication network. An edge in the constraint graph is not a physical communication link, but a logical relation between agents. Since each agent owns exactly one variable, a vertex in the constraint graph also represents an agent. The following definitions hold in this context:

Definition 1. A service demand d is defined as:

$$d = (x, y, qos, dur)$$

where $x \in A$ is the source node, $y \in B$ the destination node, with A and B two distinct networks, qos the vector expressing the minimal required bandwidth β^r and the required end-to-end delay e^r , and finally dur the duration of the service.

Definition 2. Considering several interconnected networks, an *abstract path* is an ordered list of distinct network provider domains between the source and the destination network provider domains. The provider controlling the source network I is called the *initiator* provider.

In Figure 1, the possible abstract paths between networks A and B are: $A-B-C$ and $A-D-C$.

Definition 3. Given a group of interconnected providers, a specific incoming demand d and a selected abstract path

³FIPA is a non-profit standardisation group that promotes interoperability of emerging agent-based applications, services and equipments (<http://www.fipa.org>).

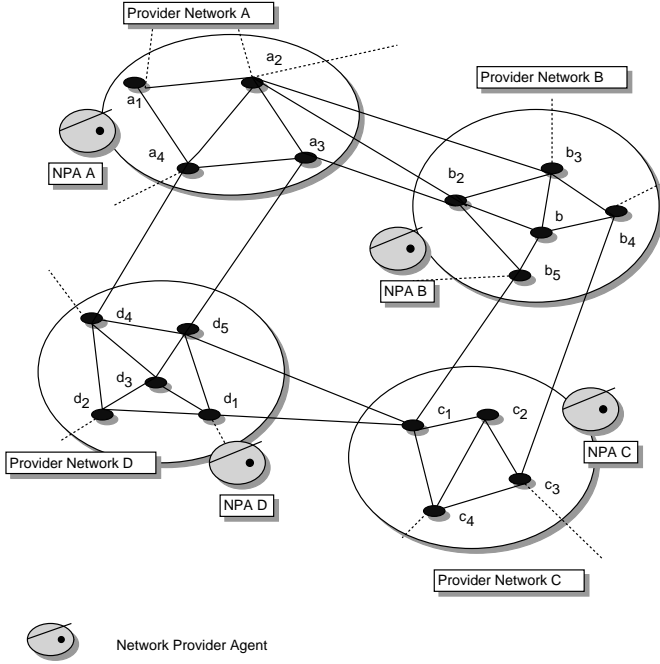


Figure 1: In this example, four providers' networks are supervised by distinct NPAs that directly control the allocation of resources (such as network nodes and links).

\mathcal{A} along which the allocation is started, the corresponding constraint graph consisting of the variables that every NPA along \mathcal{A} and including all the constraints between these variables is called the *MuSS constraint graph*.

4.2. THE VARIABLES AND THEIR DOMAINS

In the NPI framework, the *variable* every provider agent handles is a “local path” (actually it expresses the two endpoints of the intra-domain route) specified as a couple $v = (i, o)$, where i indicates a generic network entry point (or input node) and o represents a generic network exit point (or output node). The set of *values* for each “local path” (i.e., domain D) are all the possible combinations of input-output nodes, which represent the possible local routes to allocate to the demand. It is important to underline the difference between *path* and *route*. A path indicates the generic connection between two specific nodes (source and destination), while the route is a specific connection consisting of a sequence of links interconnecting the source to the destination node. Note that only *simple paths*, i.e., loop-free, are considered. In Figure 1, the path between a_1 and a_3 can correspond to any of the following routes: $r_1 = a_1 - a_2 - a_3$, $r_2 = a_1 - a_4 - a_3$, $r_3 = a_1 - a_2 - a_4 - a_3$, $r_4 = a_1 - a_4 - a_2 - a_3$. The agent NPA A receives a service demand d for establishing a video-conference between a_1 and b_4 . Assume that the selected abstract path is $A - B$. NPA A considers the two following sets:

$$I_A = \{\text{Set of possible input nodes in A for allocating } d\}$$

$$O_A = \{\text{Set of possible output nodes in A for allocating } d\}$$

Given the demand between a_1 and b_4 , $I_A = \{a_1\}$ and $O_A = \{a_2, a_3\}$, since the selected abstract path is $A - B$. The variable for agent A is the couple $p_A = (i, o)$, $i \in I_A$ and $o \in$

O_A . The *domain* for v_A is given by the set of all the possible routes connecting i to o : $D_A = \{(a_1, a_2), (a_1, a_3)\}$. Agent B owns the variable $v_B = (i, o)$, with $i \in I_B = \{b_2, b_3\}$ and $o \in O_B = \{b_4\}$. Considering only the points which are directly connected with the predecessor or with the successor along the selected abstract path, this allows the reduction of the complexity of the solving algorithm. The fewer points that are included in I_A and O_A , the fewer possible combinations there exist for allocating a given demand (search space reduction). The domain for agent B is: $D_B = \{(b_2, b_4), (b_3, b_4)\}$. Every path in the variable domain can correspond to one or several routes. For instance, given the path $p_B = (b_2, b_4)$ the set of possible internal routes is $\{(b_2, b_3, b_4), (b_2, b_3, b_1, b_4), (b_2, b_5, b_1, b_4), (b_2, b_5, b_1, b_3, b_4), (b_2, b_1, b_4), (b_2, b_1, b_3, b_4)\}$ ⁴.

4.3. THE MUSS CONSTRAINTS

In the MuSS context, two main categories of constraints are considered. *Service constraints* correspond to QoS and connectivity requirements as expressed by a specific service demand. *Network constraints* are imposed by network resource availability, i.e., they express the capacity constraints, and providers control and/or management policies [21]. Policies are expressed by inference rules either static or dynamic that every provider applies. Some of these rules depend on which network technology is deployed, others on how available network resources are managed while others on how profit, prices and costs are computed and optimised. Service constraints and network constraints are often inter-dependent. In particular, there are network constraints that can be expressed as a function of service requirements (i.e., the price of a service can be function of the required QoS).

Considering services that span distinct provider networks, QoS and connectivity service requirements first need to be locally translated into constraints between boundary network resources that neighbour providers can use for the inter-domain routing. These *boundary constraints* actually represent the only information that NPAs need to reveal to each other for achieving a globally consistent solution, i.e., an end-to-end route satisfying both inter-domain and intra-domain constraints. Intra-domain topologies, resource availability, and internal policies do not necessarily need to be shared for achieving a globally consistent solution. These *private constraints* are individually managed and deployed in order to verify and select which internal resources can be used for the service demand allocation. All boundary constraints are binary, i.e., they involve two variables and they refer to the interconnectivity of boundary nodes of adjacent provider networks. Private constraints are unary and must ensure that at least one local path verifies the QoS requirements and the provider interests. More formally, in order to define the boundary constraints it is necessary to check which are the possible combinations of input/output points from/to each other neighbour network:

$$C(A, B) = \{(o_A, i_B) \mid o_A \in O_A, i_B \in I_B, o_A \rightsquigarrow i_B\}$$

⁴Remember that only routes without loops, i.e., *simple*, are considered.

where $o_A \rightsquigarrow i_B$ means that the node o_A is directly connected to the node i_B . Considering Figure 1, the set $C(A, B)$ of possible combinations of output points of network A and input points of network B is: $C(A, B) = \{(a_2, b_2), (a_2, b_3), (a_3, b_2)\}$.

4.4. THE DISTRIBUTED CSP FORMULATION

To summarise, in distributed CSP terms, the MuSS problem can be expressed in the following way:

- The variables are intra-domain paths along the abstract path \mathcal{A} selected for allocating a given demand d .
- The domain D_i of each variable v_i is the set of all intra-domain paths that can be allocated for the demand d inside every network i along the abstract path \mathcal{A} .
- There is a constraint on each link forming the intra-domain path (unary intra-domain constraint) ensuring that there is enough available bandwidth on the link.
- There are constraints between two consecutive intra-domain paths along the abstract path \mathcal{A} (binary inter-domain constraints) ensuring that consecutive paths interconnect to each other.

A solution is then a set of intra-domain routes (one for each domain along the abstract path \mathcal{A}), respecting intra- and inter-domain constraints. Given a specific service demand, a route is said to be *feasible* when demand QoS requirements and inter-domain connectivity constraints are satisfied. Therefore, solving the MuSS problem means finding the set of feasible end-to-end routes and to select a specific one according to a coordinated interaction of the different providers involved in the allocation.

Given a provider set consisting of distinct interconnected networks, a multi-provider demand d , which has to be allocated over a sequence of interconnected local paths represented by the set of variables $\{v_1, v_2, \dots, v_j\}$ with $\{D_1, D_2, \dots, D_j\}$ the respective domains containing all the possible paths satisfying both the intra- and the inter-domain constraints:

- *Find on-demand the set of feasible routes formed by local interconnected connections so that the intra- and inter-domain constraints on the required network resources are satisfied for each link forming the connection.*
- *Select one global end-to-end route in coordination with all the other providers involved in the service demand allocation.*

5. THE NPI SOLUTION

Distinct NPAs need to coordinate themselves in order to find the space of possible end-to-end routes for a given traffic demand, thus creating a concise representation of all possible routes. This space provides the basis for subsequent negotiations about what specific route to select and at which price. In order to converge to global consistent end-to-end

routes by revealing only boundary constraints we define the *distributed arc consistency* (DAC) mechanism [4]. The main steps of the DAC process can be summarised as follows:

- **Step 1: Inter-domain source routing.** Computation and selection of a specific abstract path \mathcal{A} along which the allocation is started. *Source* routing means that the forwarding path for all data traffic generated by an incoming demand is computed at the source network, i.e., by the *initiator* NPA. If no feasible abstract paths exist, the end user is requested to relax the service requirements otherwise the service demand is rejected.
- **Step 2: Peer providers coordination.** Contacting all network providers along the selected path \mathcal{A} . The *initiator* requires peer NPAs to participate to the allocation of a multi-provider service demand. If one or more provider/s along \mathcal{A} refuse to collaborate, the *initiator* can decide to investigate an alternative abstract path when possible.
- **Step 3: Intra-domain resource check.** Local resource availability check, also called *node consistency phase*. This step is performed for determining the set of possible intra-domain routes given the current network state. If one NPA along \mathcal{A} does not have enough available resources, an alternative abstract path can be explored.
- **Step 4: Arc consistency filtering phase.** Making the set of intra-domain possible routes consistent with boundary constraints. If one NPA along \mathcal{A} has an empty feasible set, a failure message is sent to each agent involved in the allocation. The *initiator* can decide to investigate an alternative abstract path when possible.
- **Step 5: Negotiation between peer providers,** in order to converge to a specific consistent end-to-end route. Among all possible consistent solutions in its feasible set, every self-interested NPA proposes a specific local route without having to reveal internal preferences, strategies or utility functions to any other entity.

In the following, the focus is on the last three phases (for more details about the complete process we refer to [3]).

5.1. THE NODE CONSISTENCY PHASE

In order to verify if enough local network resources are available to allocate an incoming demand with specific QoS requirements, every provider evaluates its current network state. In CSP terms, this means that every NPA j tests its node consistency. This is done by verifying if the domain D_j collecting the possible values that its local variable v_j (or local path) can take is non empty and contains only values (intra-domain paths) satisfying unary (or intra-domain) constraints. Unary constraints include both service and network constraints (e.g., QoS requirements, routing policies).

In the NPI context, NPAs make use of specific resource abstraction techniques to quickly estimate the current network state. The used abstraction technique is a clustering

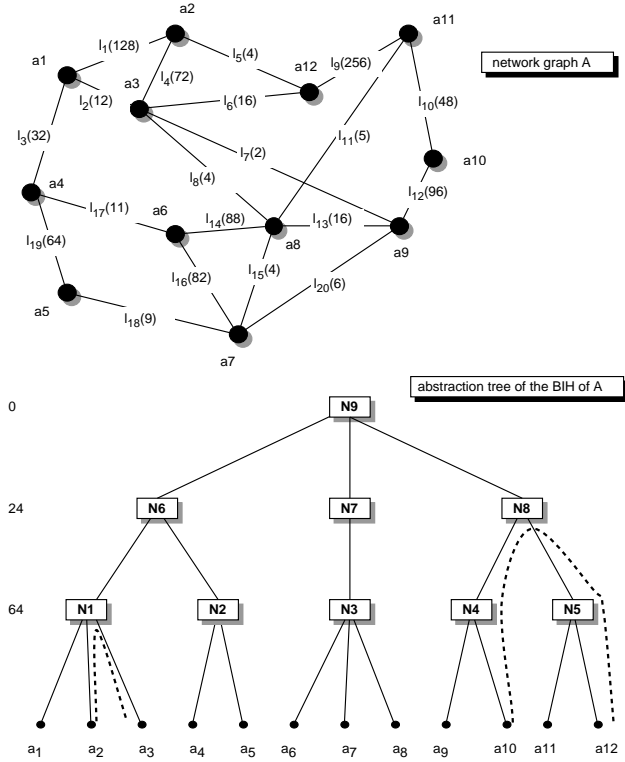


Figure 2: For the network graph A , the abstraction tree of its BIH decomposition allows the visualisation of the existing BIs for different bandwidth requirement levels. In this structure, it is immediate to verify for a given bandwidth requirement whether a route between two specific points exists or not. Whenever two nodes are in the same BI for a given bandwidth level β of the BIH, it means that there is a route interconnecting the two nodes with at least β available resources.

scheme based on Blocking Islands (BIs) [7]. The BI approach enables the representation of bandwidth availability at different levels of abstraction in a very compact way. BIs highlight the *existence* and *location* of routes. Frei proves that there is at least one route satisfying the bandwidth requirement of an unallocated demand $d_u = d(x, y, \beta_u)$ if and only if its endpoints x and y are in the same β_u -BI (see Section 5.1). Moreover, all the links that could form part of such a route lie inside this blocking island. This makes it possible to assess the existence of routes between endpoints with a given amount of bandwidth, without having to explicitly search for such a route. The node consistency step is performed by considering the *Blocking Island Hierarchy* (BIH) decomposition of every network. The BIH structure is a way of representing network resource availability based on different possible bandwidth levels. Whenever two nodes are in the same BI for a given bandwidth level β of the BIH, it means that there is a route interconnecting the two nodes with at least β available resources (route existence property of BIs, Proposition 3.4 in [7]). Consider the example depicted in Figure 2. For the network graph A , the corresponding BIH built for the bandwidth levels $\{64, 24\}$ (note that the links are omitted for clarity in the picture) consists of nine BIs. Given this structure, it is straightforward to determine that given the bandwidth requirement $\beta = 64$ there

is, for instance, a route between nodes a_2 and a_3 since they both belong to the same BI, namely N1. Analogously, it is immediate to detect that there is no route between a_{10} and a_{12} with enough available resources to satisfy the bandwidth requirement $\beta = 64$ since they belong to two distinct BIs, namely N4 and N5. The lowest bandwidth level for which both a_{10} and a_{12} belong to the same BI (i.e., N8) is $\beta = 24$.

In terms of node consistency, given a service demand d requiring β^r available bandwidth, every NPA j controlling variable v_j builds the variable domain D_j by considering all the possible couples of network nodes that belong to the same β -BI with $\beta \geq \beta^r$. Considering the example depicted in Figure 2, and the variable v_A between a_1 and a_5 with $\beta^r = 20$, the domain D_A for v_A is:

$$D_A = \{(a_1, a_2), (a_1, a_3), (a_1, a_4), (a_1, a_5)\}$$

This is indeed the set of all possible couples of nodes belonging to the same 24-BI, namely N6, with input node a_1 and output node a_5 .

5.2. THE ARC CONSISTENCY PHASE

This phase of the DAC mechanism allows the elimination of all possible intra-domain possible routes that are not consistent with constraints existing between distinct networks (i.e., inter-agent constraints). The main idea is that NPAs exchange with peer neighbour providers the minimal amount of information necessary to prune out of the feasible sets the inconsistent choices, or local routes, without revealing strategic and internal data. As anticipated in Section 4.3, the exchanged information constrains the possible boundary network resources that can be used to allocate the incoming service demand. This makes it possible to rapidly access what choices can be part of a consistent end-to-end route. Then, the space of possible solutions provides the basis for subsequent negotiations.

More formally, a given CSP is *arc consistent* if for any two distinct variables v_i and v_j forming the arc $a(v_i, v_j)$, for every value $x \in D_i$ there exists at least one value $y \in D_j$ such that all binary constraints between v_i and v_j are satisfied. It is important to underline that the concept of arc consistency is directional: if an arc $a(v_i, v_j)$ is consistent then it does not mean that the arc $a(v_j, v_i)$ is also consistent. In the NPI system, the *full arc consistency* (or bidirectional arc consistency) is performed: for every couple of variables v_i and v_j from D_i and D_j all values for which either $a(v_i, v_j)$ or $a(v_j, v_i)$ are not consistent are pruned out. This means that if the arc consistency is successful all variable domains are non-empty sets of intra-domain routes satisfying all constraints between neighbour networks. If after the arc consistency propagation, at least one variable domain D_i is empty, then the distributed CSP does not have any solution. If the variable domain size is one for all the variables, then the distributed CSP has exactly one solution, which is obtained by assigning to each variable the only possible value in its domain. Further search is needed to determine a specific solution, whenever at least one variable domain D_i contains more than one value and all the other domains D_j with $j \neq i$ are non empty. In the NPI context, the ‘search’ performed

after the arc consistency phase corresponds to NPA-to-NPA negotiations (see Section 6).

The major benefit of deploying arc consistency techniques between distinct NPAs is that they ensure the completeness of the NPI solving approach (see Section 5.3), since they detect whether a solution exists or not (if a solution exists they can guarantee to find it). However, the main drawback is that they introduce an additional level of complexity, especially when it comes to the detection of a state (i.e., global state) in which a stable property of the distributed NPI system holds [5]. The system is represented by the group of NPAs along a given abstract path, and the property defining the global state that an NPA has to detect is the full arc consistency. In our framework, the detection of the global arc consistent state is part of the procedure used for propagating boundary constraints (see Chapter 4 of [3]). Therefore, the computational complexity of the distributed arc consistency phase can be determined as it follows.

Performing the arc consistency for a binary CSP is in general bound by $O(cd^3)$ [19], where c is the number of constraints, and d the upper bound on the number of values in the domain of a variable. In the NPI context, let \mathcal{A} be the abstract path chosen for the current demand. The number of binary constraints is $|\mathcal{A}| - 1$, that is the number of variables (or involved domains/agents) along the abstract path minus 1. In order to compute the domain size $|D_A|$ of a variable v_A , all the border nodes of the network A must be considered. Let n_A be that number. There are two exclusive cases (remember that source and destination end-points are assumed not to be in the same domain – otherwise no inter-domain routing is required):

1. The network domain contains the source node or the destination node, but not both: $|D_A| \leq n_A$.
2. The network domain is a transit domain, i.e., it does not contain either the source or the destination node, but is part of the abstract path \mathcal{A} : $|D_A| \leq \frac{n_A(n_A-1)}{2}$.

Let n be the maximal amount of boundary nodes in the network of a provider along the abstract path \mathcal{A} , i.e., $n = \max_{v_i \in \mathcal{A}} n_i$. Therefore, the complexity of the arc consistency step is bound by:

$$O(|\mathcal{A}| n^6).$$

5.3. EFFICIENCY OF CONSISTENCY TECHNIQUES

The use of node and arc consistency guarantee finding one solution eventually when solutions exist, and when there exists no solution, to find it out and terminate. This characteristic is a direct consequence of the simple constraint graph that the distributed CSP representation of the MuSS problem captures, once an abstract path has been selected. This can be formally proved as follows.

The MuSS constraint graph is an *ordered* constraint graph where the vertices (i.e., NPAs or variables) are ordered linearly from the source to the destination network. This corresponds to the source-routing assumption and means that the order of variable instantiations goes logically from the

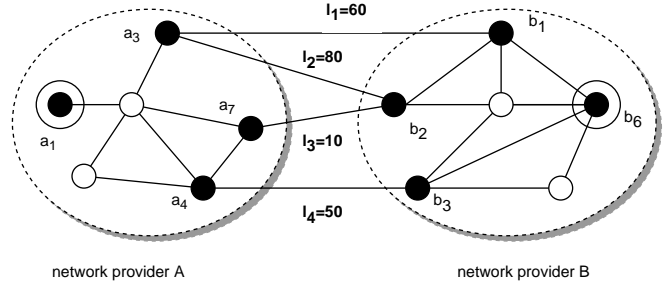


Figure 3: In this example, two distinct provider networks are involved in the service demand allocation. All network resources, both intra- and inter-domain, are displayed.

source to the destination. For every vertex in the constraint graph it is possible to define its *width* by considering the number of constraint arcs that lead from the vertex to the previous vertices (in the linear order). For a given CSP the maximum of the width of any of its vertices is the width of its constraint graph [17]. In the NPI context, it is thus possible to prove that:

Proposition 2. *The width of the MuSS constraint graph is 1.*

PROOF: This is obvious by construction of the MuSS constraint graph since once an abstract path has been selected, the constraint graph is a simple chain. Therefore, the maximum number of constraints arcs per vertex is 1. \square

In addition to the notions of ‘order’ and ‘width’, for every constraint graph it is possible to define different degrees of consistency. A given constraint graph is *K-consistent* if for any K-1 consistent variables, for any additional Kth variable there exists a value that satisfies all the constraints among these K variables. A constraint graph is *strongly K-consistent* if it is *J-consistent* for all $J \leq K$. The following theorem proved by Freuder in [8] is used to demonstrate that the DAC algorithm does not require any search to discover all potential solutions to the MuSS problem.

Theorem 1. [8] *If a constraint graph is strongly K-consistent, and K strictly greater than the width of the constraint graph, then there exists a search order that is backtrack free.*

Therefore:

Proposition 3. *If the MuSS constraint graph is node- and arc-consistent, then there exists a search order that is backtrack free.*

PROOF: The DAC algorithm uses node and arc consistency to make the MuSS constraint graph strongly 2-consistent. Therefore, by simply applying the Theorem 1 to the strongly 2-consistent MuSS constraint graph, which has width equal to 1 (see Proposition 2), it is possible to conclude that there exists a search order that is backtrack free. \square

5.4. VISUALISING THE CONSISTENCY STEPS

The distributed CSP formalism facilitates the visualisation of the links in the scenario that can or cannot support the required amount of bandwidth β^r . After the arc consistency propagation phase, the only links considered for the

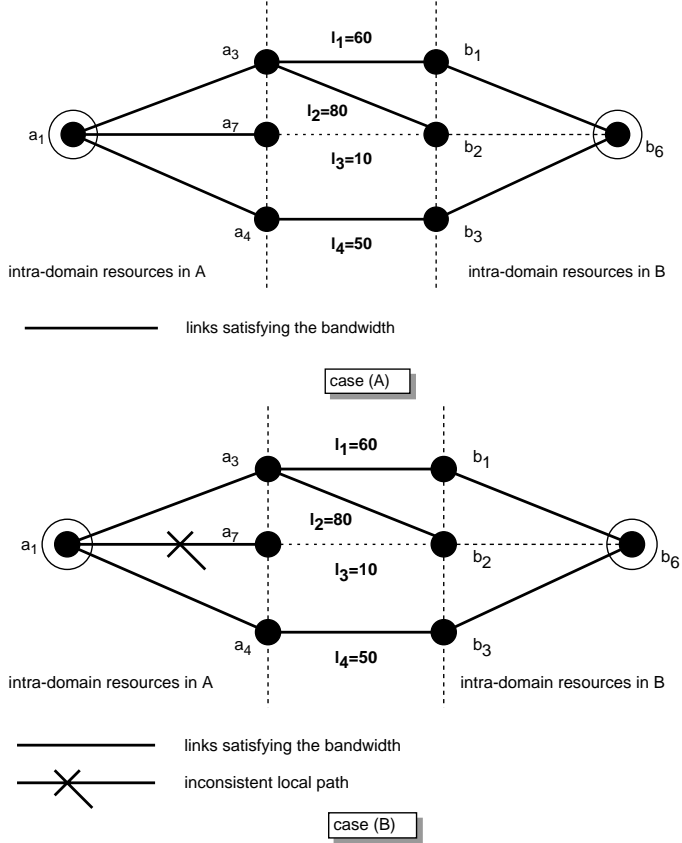


Figure 4: Visualisation of the DAC process: Case (A) shows the links satisfying the bandwidth requirements: this is the situation after the node consistency phase has been performed. Case (B) illustrates the situation after the arc-consistency phase has been performed. The value (a_1, a_7) for variable v_a has been discarded from the variable domain D_A since it is not consistent with any value in the variable domain D_B .

demand allocation are the ones which can satisfy both the bandwidth requirements and the inter-domain constraints. Consider the example depicted in Figure 3 and assume that, for instance, network provider A receives a request to allocate a service demand d between the nodes a_1 and b_6 with $\beta^r = 15$. In distributed CSP terms, NPA A controls the variable v_a that corresponds to a local path of network A . The possible values for v_a are collected into the variable domain $D_A = \{(a_1, a_3), (a_1, a_7), (a_1, a_4)\}$. Analogously, NPA B controls a variable v_b which indicates an intra-domain path of network B . The variable domain for v_b is $D_B = \{(b_1, b_6), (b_2, b_6), (b_3, b_6)\}$. In Figure 3 case (A), all the links, both intra- and inter-domain, that satisfy the bandwidth requirement $\beta^r = 15$ are highlighted. This is the configuration after every NPA has performed the node consistency step of the DAC algorithm. During the node consistency phase, NPA B prunes out the local path (b_2, b_6) from the variable domain D_B , since there is no local path connecting the network nodes b_2 and b_6 with enough available bandwidth. This can be verified by checking whether the two network nodes belong to the same β -BI, with $\beta \geq \beta^r$, or not. Figure 3 case (B) depicts the situation after the arc consistency propagation has been performed. At this stage, the variable domains are reduced to the feasible sets of v_a and v_b : $D_A = \{(a_1, a_3), (a_1, a_4)\}$ and $D_B = \{(b_1, b_6), (b_3, b_6)\}$. The value (a_1, a_7) has been pruned out from D_A since it is not consistent with any value in D_B . Remark that the provider B does not reveal the possible values of its variable v_b , but only the boundary constraint which corresponds to not using l_3 as inter-domain link. In this example, there are two possible end-to-end solutions: $(a_1, a_3) - l_1 - (b_1, b_6)$, and $(a_1, a_4) - l_4 - (b_3, b_6)$. The peer provider agents can now start to negotiate in order to select which solution to adopt.

6. NEGOTIATING NPI AGENTS

The negotiation phase represents a multi-criteria optimisation process where every NPA aims to find a solution maximising its profits while respecting the existing constraints on the resources. If the arc consistency interactions terminate successfully, i.e., all NPAs have a non empty set of local routes consistent with inter-domain constraints, the *initiator* starts bi-lateral negotiations with every NPA along \mathcal{A} for converging to a specific end-to-end route. Every contacted NPA elaborates an offer consisting of a specific local route with certain QoS characteristics at a fixed price. The decision making model which every agent relies upon to evaluate offers during the negotiation process with external entities is designed after the constraint satisfaction formalism by following the approach presented in Section 3.1. Every received offer (or proposal) for the allocation of a specific demand, is mapped into a variable containing three components: the offered price, the offered bandwidth and the offered end-to-end delay. Each of these components is constrained by a set of unary constraints. The way these constraints are evaluated and combined represents the specific agent's strategy. Since the state of the network and the

mental state of an agent can change over time, the specific level of acceptance for a given offer to be accepted should dynamically reflect those modifications. In our framework, this simply means that the set of constraints considered by an agent during the negotiation phase can dynamically vary. Moreover, the adoption of a CSP based decision making approach facilitates the combination of several strategic selection criteria. Every agent can combine different kinds of parameters, such as *time* and *resource* dependent criteria, by simply considering different types of constraints on the variables it handles. As anticipated in Section 3.2, the same mentalistic approach can be used by every NPA for engaging in meaningful conversations with other agents without having to specify a priori the exact sequence of messages that are expected to be exchanged.

6.1. A BOUNDED RATIONAL APPROACH

Interactions of self-interested entities have been extensively studied in microeconomics, especially by the game theory community [16], and in the DAI field [23]. Most of these approaches, also called the *game theoretic models*, assume perfect rationality of ‘players’, which means that agents are considered capable of flawless deduction, optimal reasoning about future circumstances and recursive modelling of other agents beliefs. In addition, in many cases, perfect computational rationality is a fundamental condition. This means that no computation is required to find mutually acceptable solutions within a feasible range of outcomes. More recently, increasing attention has been given to models where a cost is associated with both computation and decision making. In these approaches, agents are considered as rational bounded entities, which means that computational and cognitive limitations are taken into account. Among various existing bounded rational approaches for automated negotiation a further distinction can be made:

- *Heuristic Models*: a cost is associated with both computation and decision making. In particular, every agent decision making process is modelled heuristically during the course of negotiation: the chosen protocol does not prescribe an optimal course of action and the aim is to produce good solutions rather than optimal ones [1], [15].
- *Normative Models*: again a cost is associated with both computation and decision making, but here the aim is to establish the best strategy that an agent can use within a given interaction protocol. In these approaches, negotiation mechanisms can be proved to have a certain number of properties such as stability, Pareto efficiency, etc. For an overview, Rubinstein’s book [24] on modelling bounded rationality is suggested.

The negotiation framework discussed in this paper is a heuristic model that falls into a particular category of bounded rationality where boundaries, limitations and constraints are intrinsic in the CSP-based decision making model that the developed agents rely upon. In addition,

in the NPI paradigm agents have limited computational resources (i.e., a cost is associated to both computation and communication efforts) and finite time to converge to an agreement (i.e., fixed negotiation timeouts).

7. DISCUSSION

The CSP based approach discussed in this paper has been adopted for solving a complex and distributed problem such as the MuSS process. The NPI agent decision making process is modelled as constraint-based reasoning and the coordination of self-interested agents is enforced by constraint propagation methods (i.e., consistency) that determine the complete range of options (possible deals) considered by each agent during the negotiation process. By assuming a cooperative framework, how good a specific solution is for every agent is quantitatively represented by the amount of money that the agent is willing to pay for this solution. An extensive experimental analysis of the NPI solution (see Chapters 7 and 8 of [3]) validates the defined paradigm and demonstrates its feasibility in realistic network scenarios. However, for more realistic scenarios we are considering the introduction of malicious agents and therefore the need of providing truth incentive mechanisms. Therefore, for the final negotiation step of the DAC mechanism, two main alternatives are currently being explored.

- The *COalition Based* (COB) approach relies upon the dynamic coordination of providers forming on-demand coalitions and agreements supervised by a central controller. Within each coalition, this supervisor is responsible for collecting information from different providers and formulating a unique offer to the end user. In this case, incentive compatibility is achieved by integrating the Clarke tax mechanism [6]. The main idea is that each agent pays a tax corresponding to the portion of its bid (i.e., intra-domain route declared profit) that makes a difference to the final service allocation outcome (i.e., to the global coalition revenue).
- The *Vickrey Auction Based* (VAB) approach introduces a trusted auctioneer entity running second-price sealed-bid auctions in which all provider agents submit bids for the selection of a specific solution. The highest bidder wins, but at the price of the second highest bid. In private-value Vickrey auctions, the dominant strategy for all participating NPAs is to bid truthfully.

7.1. RELATED WORK

Several works have deployed CSP oriented formalism and techniques in order to enforce coordination in multi-agent systems and efficiently model and solve constraint-directed negotiation approaches. One of the earliest approaches is the *constraint-directed negotiation* paradigm [25]. Here, negotiation for resolving conflicts, i.e., coordination, in resource allocation is modelled as constraint relaxation and constraints are used for evaluation of existing alternatives as well as for creating new ones. Three main constraint-directed negotiation algorithms are proposed and experi-

mentally validated. A centralised mediator clusters several announcements and bids from multiple agents into atomic contracts (i.e., bids are grouped into *cascades*). Experimental results are discussed for the real world problem of workstation requirements within an engineering organisation: the resources are workstations that each group within the organisation uses. When projects change the groups requirements vary and therefore the initial allocation of resources have to be adapted. This kind of coordination is not always possible in scenarios where the centralisation of information may not be feasible. A fundamental limitation of the constraint-directed negotiation approach is that only constraints that are qualitative in nature are considered. In the NPI framework, utility functions have to be able to model both qualitative and quantitative constraints. Furthermore, specific search operators used for converging a global solution that accommodates all negotiation participants, such as the *relaxation* and the *reconfiguration* operators, are directly related to the specific domain they apply to and no formal and general reusable models are given. However, the approach presented by Sathi and Fox in [25] considers several aspects that are relevant and common to this work: the CSP-based decision making model, the direct mapping between agents preferences and constraints, and the use of negotiation timeouts.

Another interesting approach for coordination in multi-agent systems has been proposed by Liu and Sycara in [13]. The main idea in their framework is to partition the problem constraints into constraint types. Responsibility for enforcing constraints of a particular type is given to *specialist* agents that coordinate to iteratively change the instantiation of variables under their control according to their specialised perspective. Agents converge to the final solution through incremental local revisions of an initial, possible inconsistent, instantiation of all variables. This approach is valid in the domain of job shop scheduling.

A centralised approach integrating constraint techniques within a multi-agent system for developing a decision support system in production flow control (PFC) has been presented in [2]. The relevant part of this work is in the use of constraints for expressing relations between different entities and restrictions over resources and structures. The idea of using constraints propagation in order to coordinate several interacting parts is also close to the NPI approach. However, in our case there is no central constraint-solver as in the PFC system.

In [9], Freuder and Wallace model ‘content-focused matchmaking’ as a constraint satisfaction problem in which negotiation techniques are used. The constraint solver interacts with a human providing partial solutions (suggestions), based on partial knowledge, and gaining further information about the problem from the humans evaluation of the suggestions. Several strategies that might be useful in this context are proposed and evaluated experimentally. We believe that intelligent and automated matchmaking could be easily integrated within the NPI approach from the end user perspective, for instance, whenever a combination of Telecom services is needed.

Integrated and automated management of supply chains by means of negotiating agents deploying the CSP paradigm have recently been proposed by Sun et al. in [27]. Buyers and sellers make and/or evaluate bids by considering possible terms and conditions as CSP variables and their domains. An acceptable offer is a set of variable assignments that satisfy all constraints. Although, the decision making process of agents in this work has been modelled in a very similar way to the paradigm proposed in [27] there is a fundamental difference. Constraint satisfaction techniques are used in the supply chain framework to model a single agent decision making process: there is no propagation of constraints and no dynamic coordination between distributed entities. In the NPI context, constraint satisfaction techniques are used at two different levels: at the intra-agent level for modelling the decision making process and at the inter-agent level for enforcing coordination by means of specific consistency techniques.

Finally, a very recent framework proposes an experimental system of e-Negotiation Agents (eNAs) [14] where the negotiation problem is modelled as a constraint satisfaction problem and the negotiation process as constraint-based reasoning. Agents use the branch and bound search with the support of constraint propagation to find instantiations that satisfy the constraints of the party. However, it is not clear from the accessible results what is the amount of private information that has to be disclosed in order to achieve consistent solutions. Therefore, it is not evident to understand if self-interests prevail against cooperativeness.

8. CONCLUSION

The dynamic coordination of self-interested entities in complex and changeable environments (such as today’s multi-provider communications networks) is a very difficult problem that is receiving increasing attention. This paper argues that the deployment of constraint satisfaction techniques have major potential advantages:

- The CSP formalism can become the support of the reasoning mechanisms that agents rely upon for decision making and future behaviour, actions and strategy choice.
- Sophisticated interactions can be modelled as distributed CSPs in which autonomous agents make choices about which possible messages they will undertake within an ongoing conversation, without having to pre-select an interaction protocol.
- Several CSP techniques can be used for enforcing coordination of distinct and self-interested entities without necessarily having the need of revealing strategic and confidential information.

We finally believe that since several existing problems, such as distributed planning graphs, scheduling and resource allocation, can be represented in an abstract way as distributed routing problems, the techniques presented in this paper have the potential to be re-used in other frameworks.

REFERENCES

- [1] Mihai Barbuceanu and Wai-Kau Lo. A multi-attribute utility theoretic negotiation architecture for electronic commerce. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 239–248, Barcelona, Catalonia, Spain, 2000. ACM Press.
- [2] H. Baumgertel, S. Bussmann, and M. Klosterberg. Combining multi-agent systems and constraint techniques in production logistics. pages 361 – 367., 1996.
- [3] M. Calisti. *Coordination and Negotiation for Solving Multi-Provider Service Provisioning*. PhD thesis, Artificial Intelligence Laboratory - Swiss Federal Institute of technology of Lausanne, 2001.
- [4] M. Calisti, C. Frei, and B. Faltings. A distributed approach for QoS-based multi-domain routing. *AiDIN'99, AAAI-Workshop on Artificial Intelligence for Distributed Information Networking*, 1999.
- [5] K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *TOCS*, 3(1):63–75, February 1985.
- [6] E. H. Clarke. Multipart pricing of public goods. In *Public Choice*, pages 17–33, 11 1971.
- [7] C. Frei. *Abstraction Techniques for Resource Allocation in Communication Networks*. PhD thesis, Artificial Intelligence Laboratory - Swiss Federal Institute of technology of Lausanne, 2000.
- [8] Eugene C. Freuder. Direct independence of variables in constraint satisfaction problems. Technical Report 84-15, University of New Hampshire, Department of Computer Science, March 1984.
- [9] Eugene C. Freuder and Richard J. Wallace. Suggestion strategies for constraint-based matchmaker agents. In *Principles and Practice of Constraint Programming*, pages 192–204, 1998.
- [10] Y. Hamadi, C. Bessière, and J. Quinqueton. Backtracking in distributed constraint networks. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 219–223.
- [11] Michael N. Huhns and David M. Bridgeland. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1437–1445, - 1991.
- [12] N. R. Jennings. Building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.
- [13] L. JyiShane and K. Sycara. Emergent constraint satisfaction through multi-agent coordinated interaction. 1993.
- [14] R. Kowalczyk and V. Bui. On Constraint-Based Reasoning in e-Negotiation Agents. In F. Dignum and U. Cortes, editors, *Agent-Mediated Electronic Commerce III: Current Issues in Agent Based Electronic Commerce Systems*, page 31 ff. Springer Verlag, 2001.
- [15] Sarit Kraus, Jonathan Wilkenfeld, and Gilad Zlotkin. Multiagent negotiation under time constraints. *Artificial Intelligence*, 75(2):297–345, 1995.
- [16] David M. Kreps. *A Course in Microeconomic Theory*. Princeton University Press, Princeton, NJ, USA, 1990.
- [17] Vipin Kumar. Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine*, 13(1):32–44, 1992.
- [18] Susan E. Lander and Victor R. Lesser. Understanding the role of negotiation in distributed search among heterogeneous agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Chambery, France, / 1993.
- [19] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.
- [20] C. Mason and R. Johnson. Datms: A framework for distributed assumption based reasoning. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence*, pages 293–318. Morgan Kaufmann, 1989.
- [21] O. Prnjaj. Integrity methodology for interoperability environments. *IEEE Communications Magazine*, 37(5):126–132, 1999.
- [22] D. Pruitt. *Negotiation Behavior*. Academic Press, New York, 1981.
- [23] Rosenschein, J. S. and Zlotkin, G. *Rules of Encounter*. MIT Press: Cambridge, MA., 1994.
- [24] A. Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998., 1998.
- [25] Arvind Sathi and Mark S. Fox. Constraint-directed negotiation of resource reallocations. In Michael N. Huhns and Les Gasser, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Pitman, 1989.
- [26] Gadi Solotorevsky and Ehud Gudes. Algorithms for solving distributed constraint satisfaction problems (DCSPs). In B. Drabble, editor, *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 191–198. AAAI Press, 1996.
- [27] R. Sun, B-T. Chu, R. Wilhelm, and J. Yao. A csp-based model for integrated supply chains. In *Working Notes of the Workshop: Artificial Intelligence for Electronic Commerce*, Orlando, Florida, July 1999.
- [28] XC00061. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*, 2000.
- [29] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. Distributed Constraint Satisfaction for Formalising Distributed Problem Solving. *Proceedings 12th IEEE International Conference on Distributed Computing Systems.*, pages 614–621, 1992.
- [30] M. Yokoo and K. Hirayama. Frequency assignment for cellular mobile systems using constraint satisfaction techniques. *Proceedings of the IEEE Annual Vehicular Technology Conference*, 2000.
- [31] Makoto Yokoo. Asynchronous weak-commitment search for solving large-scale distributed constraint satisfaction problems. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, page 467, San Francisco, CA, 1995. MIT Press. (poster).
- [32] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.