Towards Argumentation as Distributed Constraint Satisfaction

Hyuckchul Jung **Milind Tambe**
Dept. of Computer Science/Information Sciences Institute
University of Southern California
Henry Salvatori Computer Center
Los Angeles, CA 90089-0781, USA
{jungh,tambe}@usc.edu

Abstract

Conflict resolution is a critical problem in distributed and collaborative multi-agent systems. Negotiation via argumentation (NVA), where agents provide explicit arguments (justifications) for their proposals to resolve conflicts, is an effective approach to resolve conflicts. Indeed, we are applying argumentation in some real-world multi-agent applications. However, a key problem in such applications is that a well-understood computational model of argumentation is currently missing, making it difficult to investigate convergence and scalability of argumentation techniques, and to understand and characterize different collaborative NVA strategies in a principled manner. To alleviate these difficulties, we present distributed constraint satisfaction problem (DCSP) as a computational model for NVA. We model argumentation as constraint propagation in DCSP. This model enables us to study convergence properties of argumentation, and formulate and experimentally compare two sets of 16 different NVA strategies (over 30 strategies in all) with different levels of agent cooperativeness towards others. One surprising result from our experiments is that maximizing cooperativeness is not necessarily the best strategy even in a completely cooperative environment. In addition to their usefulness in understanding computational properties of argumentation, these results could also provide new heuristics for speeding up DCSPs. [1]

Introduction

Distributed, collaborative agents are promising to play an important role in large-scale multi-agent applications including virtual environments for training, distributed disaster rescue applications, agent assisted human organizations, and distributed spacecrafts (Barrett 1999; Rickel & Johnson 1997; Scerri, Pynadath, & Tambe 2001; Tambe 1997). While such applications require agents to be collaborative, conflicts among these agents are inevitable, often due to their limited (shared) resources(Tessier, Chaudron, & Muller 2000). Furthermore, given that in distributed, dynamic and complex environments, no single agent can have a complete and accurate picture of the entire domain, it is often impractical, if not impossible, to avoid such conflicts via centralized planning.

Negotiation via argumentation (NVA) offers a promising approach to collaborative conflict resolution(Kraus, Sycara, & Evenchik 1998; Parsons & Jennings 1996). In this approach, while agents negotiate as usual by sending each other proposals and counter-proposals, these proposals are accompanied by supporting arguments (explicit justifications). We have successfully applied NVA to conflict resolution among agents in real world applications. While these implemented argumentation systems have performed well in small-size applications, no systematic investigation on large-scale argumentation systems has been done. Thus, in scaling up the systems, several major questions regarding the computational performance of argumentation remain open. One key open question is understanding if (and when) argumentation actually speeds up conflict resolution convergence. The presence of explicit justifications in argumentation could fail to improve convergence and may degrade performance due to processing overheads. Another key open question is formulating different collaborative NVA strategies and understanding their impact on convergence.

Answering the above questions requires that we define an abstract, well-understood *computational model of argumentation*, suitable for large-scale experimental investigations. Certainly, answering such questions by building adhoc, complex agent argumentation systems is costly, labor intensive, and problematic in identifying critical factors of their success or failure. Another alternative is to exploit existing formalizations of argumentation in logic, such as modal logic(Kraus, Sycara, & Evenchik 1998) and dialectical logic(Sawamura, Umeda, & Meyer 2000). However, these formalizations focus on modeling agents' complex mental states, and are sometimes suggested as tools for design specification, making them unsuitable as efficient computational models for large-scale experimental investigation.

To alleviate the above difficulties, this paper proposes distributed constraint satisfaction problem (DCSP)(Armstrong & Durfee 1997; Yokoo & Hirayama 1998) as a novel computational model of NVA. Argumentation is modeled in DCSP as communications of agents' local constraints and their propagation by other agents. We focus specifically on one of the best published DCSP algorithms, that of Yokoo (Yokoo & Hirayama 1998), and model argumentation as an extension to this algorithm by communicating local constraints. Argumentation essentially enables this DCSP algorithm to interleave constraint propagation in its normal execution. Next, using this extended DCSP as our computational model, we formulate different NVA strategies, varying the level of cooperativeness towards others. While cooperativeness to-

[1]This paper significantly extends our earlier conference paper(Jung, Tambe, & Kulkarni 2001).

wards others would appear to be fundamentally important in a cooperative environment, the DCSP model enables a formalization of this notion. We specifically formulate different NVA strategies as varying the value ordering heuristics(Frost & Dechter 1995). The basic idea is to make some variable values more or less preferable than the others, so as to model the varying of cooperativeness towards others.

Existing DCSP algorithms and incorporation of argumentation into DCSP enables us to investigate the impact of argumentation and different NVA strategies in the large-scale, which provides the following results. First, argumentation can significantly improve agents' conflict resolution convergence, and the overhead of argumentation is in general outweighed by its benefits. Here, the benefits of argumentation vary non-monotonically with the proportion of agents that offer tightly constraining arguments for their proposals. Second, with respect to NVA strategies, given a highly collaborative environment, the expectation was that more cooperativeness will lead to improved performance. However, a surprising result we obtain is that a maximally cooperative strategy is not the most dominant strategy. While some improvements in cooperativeness significantly improve performance, further improvements do not help and may end up degrading performance. These results are measured in negotiation cycles required to converge to a solution and thus these are not just artifacts of constraint processing overheads.

NVA and its Application Domains

This section introduces key concepts of NVA and two application domains which require a significant scale-up in NVA, thus concretely motivating the need for computational models of argumentation, such as the one we introduce later.

Negotiation is a process by which a group of agents come to a mutually acceptable agreement on some issue by communicating their proposals for the issue(Jennings *et al.* 2001). In argumentation, agents send their proposals/counter-proposals with explicit justifications (*arguments*). Following (Jennings *et al.* 2001), *negotiation objects* refer to issues over which negotiation takes place. For a negotiation object, an agent selects values based on its private information and preferences that may not be known to other agents. For instance, in a distributed spacecraft domain(Barrett 1999), a negotiation object may refer to the direction to point a telescope. Agents can then send proposals for particular values of negotiation objects to others (e.g., a specific 3D orientation for the telescope), by communicating the local information as arguments for their proposal. NVA is an iterative process in which (i) agents generate proposals for their negotiation objects and supporting arguments; (ii) agents' proposals and arguments are evaluated by the other agents; and (iii) after evaluating received proposals/arguments, agents may accept/reject it and, in the case of rejection, send a counter-proposal.

Earlier we have developed a domain independent NVA system, called CONSA (COllaborative Negotiation System based on Argumentation)(Tambe & Jung 1999). CONSA is illustrative of complex, practical systems developed for NVA, that have been successfully applied in small-scale situations. In CONSA, agents involved in a conflict start a negotiation process that consists of three stages: opening, argumentation, and termination. In the opening stage, agents agree to jointly resolve the current conflict, establishing a joint commitment for conflict resolution. In argumentation stage, they repeat cycles of communicating proposals and arguments, terminating argumentation when the conflict is resolved or found unachievable to resolve. CONSA's representation of arguments is based on Toulmin's argumentation pattern(Toulmin 1958). A proposal is supported by agents' beliefs (including private beliefs) and inference rules. Each fact can be supported by others as well, which builds a tree structure arguments. Upon receiving such a proposal, other agents may accept or reject it; if they reject, then the argumentation cycle continues. CONSA has been successfully applied to one of our application domains described below.

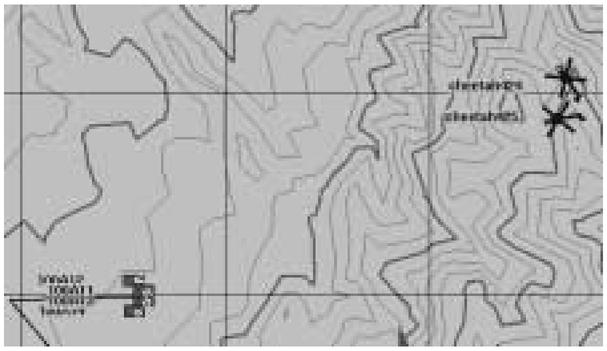


Figure 1: Helicopter combat simulation: an overhead plan-view display from the ModSAF simulator, with contour lines illustrating the terrain of the virtual environment.

Motivating Examples

The first application domain that motivates this research is the helicopter combat simulation domain(Tambe 1997). Different conflict situations arise in a team of simulated pilot agents. One example, henceforth called the *firing positions case*, involves allocating *firing positions* for a team of pilots. Individual pilots in a helicopter team attack the enemy from firing positions. Each firing position must enable a pilot to shoot at enemy vehicles while protecting the pilot from enemy fire. In addition, a pilot's firing position is constrained by the firing positions of the others: two firing positions are in conflict if they are within one kilometer of each other. Therefore, each agent has to negotiate its position with others to avoid conflict and provide safety.

Figure 1 is a snapshot of the firing position conflict where such negotiation is necessary. To attack nearby enemy units(e.g., 100A12, 100A11, etc.), two pilot agents, *cheetah424* and *cheetah425*, must occupy good firing positions. Unfortunately, they are too close to each other (only 100 meters) and need to negotiate to resolve their firing position conflict. Furthermore, while *cheetah425* is oriented in such a direction so as to see the enemy units, *cheetah424* is unable to see them since it is covering a different portion of the battlefield; in fact, *cheetah424* sees different enemy units. Thus, these agents must negotiate for appropriate firing positions.

CONSA has been successfully applied in the helicopter combat simulation domain to address conflicts such as these (and others). Agents using CONSA may negotiate as follows. First *cheetah424* computes new values for the firing positions of both agents. Often several such firing position values are possible. Here, *cheetah424* selects one value, offering equi-distant movements; and then communicates its proposal to *cheetah425*, suggesting $\{$(*cheetah424 move 450 m left, cheetah425 move 450 m right*)$\}$, where the appended justification includes $\{$(*enemy E1 position, current separation 100 m,...*)$\}$. When *cheetah425* receives and evaluates the proposal, it realizes that it cannot move 450 meters right because of other enemy units E2. Instead, it counter-proposes $\{$(*cheetah424 move 600 m left, cheetah425 move 300 m right*)$\}$, with the justification being that $\{$(*enemy E1 position, enemy E2 position,...*)$\}$. However, *cheetah424*'s 600 meter move to the left may cause a conflict with a third agent *cheetah426*, requiring further negotiation.

The second domain motivating this research is that of distributed sensor networks. This domain consists of multiple stationary sensors, each controlled by an independent agent, and targets moving through their sensing range. Each sensor is equipped with a Doppler radar with three sectors. An agent may activate at most one sector of a sensor at a given time or switch the sensor off. While all of the sensor agents must act as a team to cooperatively track the targets, there are some key difficulties in such tracking. First, in order for a target to be tracked accurately, at least three agents must concurrently turn on overlapping sectors to triangulate the target's position. Second, to minimize power consumption, sensors need to be periodically turned off. Third, sensor readings may be noisy and false detections may occur. Finally, the situation is dynamic as targets move through the sensing range.

To address this problem, agents may negotiate via argumentation to enable them to coordinate their individual sector choice. For example, if an agent A detects an object in its sector 1, it may negotiate via argumentation with neighboring agents, B and C say, so that they activate their respective sectors that overlap with A's sector 1. In particular, it may be the case that B is low in power or C is busy with another target. Thus, if agent A is able to provide an argument ("target detected in sector 1") to B and C, it may induce them to switch sectors. Alternatively, B may counter-propose that it cannot turn on its sector, with an argument "low in power".

Argumentation open questions

The above applications illustrate the importance of investigating the scale up properties of argumentation. For both domains, argumentation appears useful for a small number of agents. However, in cases involving hundreds of distributed sensors in a grid or hundreds of pilot agents in formation, argumentation may not provide significant enough benefits to outweigh its overheads (of processing arguments). Thus, to justify the use of argumentation, we need to investigate if (and when) argumentation will truly speed up conflict resolution convergence with scale up.

A second very important open question is to investigate different collaborative NVA strategies and their impact on convergence. Being cooperative is clearly important in both domains, e.g., if pilot agents refuse to move, the problem may in some cases be unsolvable. In some other cases, the pilot agents' maximal cooperativeness towards others, by offering to move the maximal distance they are allowed, would appear to be very helpful. However, as we scale up the number of agents, it is unclear if cooperativeness in general, or maximal cooperativeness in particular, will necessarily lead to improved performance. Unfortunately, answering these questions by building ad-hoc implementations is difficult — the process would be impractical and labor intensive, and the factors that led to success or failure of argumentation may remain unclear. For instance, applying CONSA to actual combat simulations involving 100s of pilot agents would appear extremely difficult. Therefore, to address the above questions, we need a computational model of argumentation that is suitable for large-scale experimental investigations.

NVA as DCSP

To provide an abstract and well-understood computational model for NVA, we propose a novel computational model, that of Distributed Constraint Satisfaction Problem (DCSP)(Armstrong & Durfee 1997; Yokoo & Hirayama 1998). DCSP allows us to easily model conflicts via constraints. As a well-investigated problem, it provides efficient algorithms to build on. *Most importantly, it also allows us to very efficiently model the use of argumentation in negotiation.*

A Constraint Satisfaction Problem (CSP) is commonly defined by a set of n variables, $X = \{x_1, ..., x_n\}$, each element associated with value domains $D_1, ..., D_n$ respectively, and a set of k constraints, $\Gamma = \{C_1, ..., C_k\}$. A solution in CSP is the value assignment for the variables which satisfies all the constraints in Γ. A distributed CSP is a CSP in which variables and constraints are distributed among multiple agents(Yokoo & Hirayama 1998). We consider DCSPs with multiple variables per agent(Yokoo & Hirayama 1998). Formally, there is a set of m agents, $Ag = \{A_1, ..., A_m\}$. Each variable (x_i) belongs to an agent A_j. There are two types of constraints based on whether variables in the constraint belong to a single agent or not:

- For a constraint $C_r \in \Gamma$, if all the variables in C_r belong to a single agent $A_j \in Ag$, it is called a *local constraint*.
- For a constraint $C_r \in \Gamma$, if variables in C_r belong to different agents in Ag, it is called an *external constraint*.

Solving a DCSP requires that agents not only satisfy their local constraints, but also communicate with other agents to satisfy external constraints. Note that DCSP is not concerned with speeding up a centralized CSP via parallelization(Yokoo & Hirayama 1998); rather, it assumes that the problem is originally distributed among the agents. This assumption suits us well, since our negotiation problem is indeed a distributed one.

Given this DCSP framework, we map argumentation onto DCSP as follows. First, we divide an agent's set of variables into two subsets. In particular, an agent's negotiation objects are modeled as externally constrained variables, henceforth referred to as *negotiation variables*. There are external constraints among negotiation variables of different agents, modeling the existing conflict. Local information (that is, at least initially, only known locally by an agent) is mapped into locally constrained variables, and referred to as *local variables*. An agent's local variables locally constrain its negotiation variables: there exist local constraints between them.

Negotiation via Argumentation	DCSP	E.g., Firing position case
Negotiation objects	Negotiation variables	Pilot agents' positions
Local information	Local variables	Enemy positions, etc.
Conflicts among agents	External constraints	Safe distance condition
Argument/justification	Internal constraints	Restriction on pilot agent's position from enemies
Argumentation	Communicating constraints and Constraint propagation	Communicate justification as restriction on own firing position; and its causing restriction on other pilots
Negotiation strategy	Value ordering heuristic	Which proposal to send first

Table 1: Mapping NVA onto DCSP.

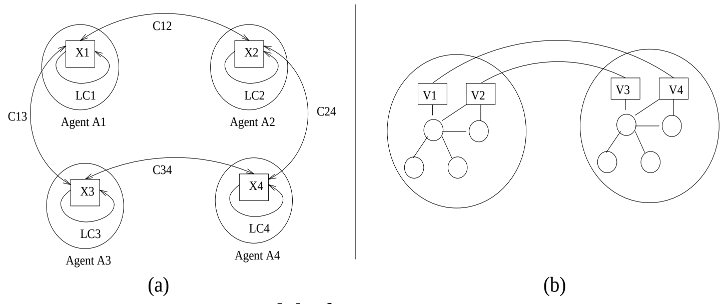


Figure 2: Model of agents in argumentation.

Table 1 provides the complete mapping. Column 1 lists a concept from Negotiation via Argumentation. Column 2 shows its mapping terms of DCSP. We have outlined the mapping for the first four rows above. The mappings of argumentation and negotiation strategies in the lower two rows will be explained later. Column 3 in Table 1 presents examples corresponding to the mapped concept from the firing position case. We will now discuss this case in more detail:

- **Example**: Each helicopter pilot's firing position is its single negotiation variable known to other agents. External constraints exist among the negotiation variables of neighboring agents: the firing positions must be at least 1000 meters apart. Enemy positions that are locally observed by individual agents form local variables. Since enemy positions (local variables) constrain firing positions (negotiation variables), there exist local constraints between them.

In our initial experimental investigations for DCSP, we found it sufficient to model each agent as having only one negotiation variable. However, there is no limitation on the number of external constraints on this negotiation variable. Thus, as illustrated in Figure 2.a, each agent (denoted by a big circle) has only one negotiation variable X_i, but there can be any number of external constraints C_{ij}, C_{ik}, etc between negotiation variables X_i and X_j, X_i and X_k, etc. Here, for expository purpose, all the local variables and local constraints that constrain the single negotiation variable are represented as a single node constraint (LC_i) on the negotiation variable. Note that we can easily extend this mapping to problems involving multiple negotiation and local variables per agent, as shown in Figure 2.b; in particular, DCSP algorithms such as Asynchronous Weak Commitment search or AWC(Yokoo & Hirayama 1998) can already deal with multiple variables per agent. Here, the squares labeled v1, v2, v3, and v4 are negotiation variables, and the small circles and the links between them are local variables and constraints. This extension will be considered in our future work.

Now, a major aspect of our mapping is to formalize argumentation in the context of DCSP. The key here is that argumentation can be mapped on as communicating constraints followed by *constraint propagation*, which is a process of reducing domains by filtering the values that cannot participate in a solution. That is, in DCSP algorithms such as AWC (Yokoo & Hirayama 1998), agents communicates the values assigned to their externally constrained negotiation variables. However, with argumentation, agents also communicate their arguments in the form of local constraints (e.g., the LC_i in Figure 2.a) under which they made the selections of values for their negotiation variables. These local constraints are propagated by the agents receiving the argument.

Concretely, argumentation in DCSP works as follows. Here we assume agents A_i and A_j are connected by the external constraint C_{ij} where X_i has domain D_i and X_j has domain D_j. Suppose an agent A_j selects a value v_j for its negotiation variable X_j. It will then send its selection v_j and its local constraint LC_j to its neighboring agent A_i whose negotiation variable is X_i. After receiving the local constraint LC_j from A_j, A_i will propagate the received constraint LC_j, reducing the domain D_i of its negotiation variable X_i. Here, this constraint propagation can be considered as addition of a new local constraint to agent A_i by which A_i's incompatible values with A_j are eliminated. To elaborate on this, we define function Φ and Ψ of A_i as follows.

- **Definition 1**: For an agent A_i and its neighboring agent A_j who share an external constraint C_{ij}, assume A_i receives A_j's local information (LC_j and D_j).
  - Domain inference function $\Phi(D_j, LCj)$ returns a restricted domain of D_j, D'_j, inferred by applying LC_j to D_j. (D'_j does not contain any value which violates LC_j.)
  - Constraint propagation function $\Psi(C_{ij}, D_j, LC_j)$ returns a new local constraint LC' for X_i such that LC' eliminates any value v from D_i if, given the external constraint C_{ij}, v for X_i has no compatible value for X_j in $\Phi(D_j, LC_j)$.

Thus, with constraint propagation, agent A_i's current local constraint LC_i is changed to $LC_i \wedge \Psi(C_{ij}, D_j, LC_j)$: A_i eliminates its own values that are not compatible with LC_i as well as those not compatible with the values in A_j's currently restricted domain. (This modified local constraint LC_i is communicated to A_i's neighbors later.) While this constraint propagation amounts only to arc consistency (Kumar 1992), it is not run by itself to solve the DCSP, rather it is interleaved with value selection. For instance, during each cycle of message communication in AWC algorithm, we first propagate communicated constraints and then select values for variables. Thus, we do not increase the number of communications, an important issue for DCSP.

NVA Strategies

A major benefit of the mapping NVA to DCSP is that different NVA strategies can now be formalized using DCSP. A negotiation strategy refers to the decision function used by an agent to make a proposal or counter-proposal. A good negotiation strategy attempts to select proposals that will speed up conflict-resolution. In the mapping to DCSP, a negotiation strategy in NVA is modeled as a value ordering heuristic used to choose a value of an assignment for a negotiation variable. A value ordering heuristic ranks the value of variables(Frost & Dechter 1995). To speed up conflict resolution, values which are more likely to lead to a solution should be tried first. Even a slight difference of the order in which values are considered can have significant impact on the time until a solution is found. In the DCSP mapping of argumentation, different value ordering heuristics can be considered, each leading to different negotiation strategies.

Formalization of Cooperative NVA Strategy

In AWC(Yokoo & Hirayama 1998), the min-conflict heuristic is used for value ordering: given a variable that is in conflict, min-conflict heuristic assigns it a value that minimizes the number of conflicts with the values of the other variables(Minton *et al.* 1990). This min-conflict heuristic is used as a baseline negotiation strategy and we refer it as S$_{basic}$. Here, S$_{basic}$ strategy selects values only based on the values of the other agents' negotiation variables (without exploiting argumentation in generating a more cooperative response to others). Argumentation enables agents to consider the constraints that the neighboring agents have on their domains, which are communicated as arguments. Considering neighboring agents' local constraints enables an agent to generate a more cooperative response, i.e., select a value which gives more choices to neighbors, and thus, potentially lead to faster negotiation convergence. To elaborate on this point, we first define our notion of cooperativeness. For this definition, let A_i be an agent with a negotiation variable X_i, domain D_i, and a set of neighboring agents N_i.

- **Definition 2**: For a value $v \in D_i$ and a set of agents $N_i^{sub} \subseteq N_i$, *flexibility function* is defined as $f_{co}^{\psi}(v, N_i^{sub})$ = $\psi(c(v, A_j))$ where (i) $A_j \in N_i^{sub}$; (ii) $c(v, A_j)$ is the number of values of X_j that are consistent with v; and (iii) ψ, referred to as the flexibility base, can be sum, min, max, $product$ or $weighted\ sum$;

As an example of the flexibility function $f_{co}^{\psi}(v, N_i^{sub})$, suppose the flexibility base ψ is chosen to be sum. Suppose we are given two neighboring agents A_1 and A_2, where a value v leaves 70 consistent values to A_1 and 40 to A_2 while another value v' leaves 50 consistent values for A_1 and 49 to A_2. Now, assuming that values are ranked based on flexibility, an agent which ranks values using the flexibility base sum will prefer v to v': $f_{co}^{sum}(v, \{A_1, A_2\})$ = 110 and $f_{co}^{sum}(v', \{A_1, A_2\})$ = 99. If ψ is set to min however, an agent will rank v' higher than v: $f_{co}^{min}(v, \{A_1, A_2\})$ = 40 and $f_{co}^{min}(v', \{A_1, A_2\})$ = 49.

We now provide two additional important definitions:

- **Definition 3**: For a value v of X_i, *cooperativeness* of v is defined as $f_{co}^{\psi}(v, N_i)$. That is, the *cooperativeness* of v measures how much flexibility is given to all of A_i's neighbors by v.
- **Definition 4**: A *maximally cooperative* value of X_i is defined as v_{max} such that, for any other value $v_{other} \in D_i$, $f_{co}^{\psi}(v_{max}, N_i) \geq f_{co}^{\psi}(v_{other}, N_i)$.

The concept of cooperativeness goes beyond merely satisfying constraints of neighboring agents and enables even faster convergence. That is, an agent A_i can provide a more cooperative response to a neighbor agent A_j, by selecting a value for its negotiation variable that not only satisfies the constraint with A_j, but maximizes flexibility (choice of values) for A_j. If A_i selects v_{max}, giving A_j more choice, then A_j can more easily select a value that satisfies A_j's local constraints and other external constraints with its neighboring agents such as A_k. This is partly the rationale for the pilots in the firing positions case (described in "Motivating Examples" section) to offer the maximum flexibility to each other. Having lower possibility of constraint violation, this cooperative response can lead to faster convergence.

S$_{basic}$ tries to minimize the number of constraint violations without taking neighboring agents' own restrictions into account for value ordering. An agent A_i's selected value v with S$_{basic}$ is thus not guaranteed to be maximally cooperative, i.e., $f_{co}^{\psi}(v, N_i) \leq f_{co}^{\psi}(v_{max}, N_i)$. Hence, S$_{basic}$ is not the most cooperative strategy to neighboring agents. Here, other cooperative strategies can be introduced and formalized in terms of value ordering, i.e., an agent A_i can rank each value (v) in its domain D_i based on the cooperativeness $f_{co}^{\psi}(v, N_i)$ of that value. Next, we define different cooperative strategies which rely on the basic AWC framework.

Cooperative NVA strategies in AWC framework

Since AWC, the state of the art DCSP algorithm is central to the NVA strategies we discuss below, it is important to first discuss AWC in some detail. In AWC, the S$_{basic}$ strategy is used in two cases described below. When an agent A_i selects a new value for its negotiation variable X_i, the value selection depends on whether A_i can find a consistent value v from the domain D_i of X_i. Here, v is said to be consistent if v satisfies A_i's constraints with higher priority agents [2]. If there exists a consistent value v in D_i, we refer to it as *good* case. In the good case, an agent applies S$_{basic}$ minimizing constraint violations with lower priority agents. On the contrary, if there is no such v, we refer to it as *nogood* case. In the *nogood* case, an agent A_i increases its priority as $max + 1$, where max is the highest priority of its neighboring agents and uses S_{basic} to minimize constraint violations over all neighboring agents(Yokoo & Hirayama 1998).

Different negotiation strategies can be described in terms of the *good* and *nogood* cases because different value ordering methods can be applied in these two cases. We will now more formally describe these negotiation strategies. Let

[2] A non-negative integer is assigned to each variable as a priority. Agent and variable in our description are used interchangeably because each agent has only one variable in the current mapping.

N_i^{high} (N_i^{low}) be the subset of N_i such that, for every $A_j \in N_i^{high}$ (N_i^{low}), the priority of A_j's negotiation variable X_j is higher (lower) than the priority of A_i's negotiation variable X_i. In the *good* case, an agent A_i computes a set (V_i) of consistent values for X_i from its domain D_i. For any flexibility base ψ, in selecting a value from V_i, four different negotiation strategies can be considered in the *good* case as follows.

- **S**$_{high}$: Each agent A_i selects a value v from V_i which maximizes $f_{co}^{\psi}(v, N_i^{high})$ i.e., A_i attempts to give maximum flexibility towards its higher priority neighbors wrt ψ.
- **S**$_{low}$: Each agent A_i selects a value v from V_i which maximizes $f_{co}^{\psi}(v, N_i^{low})$, i.e., A_i attempts to give maximum flexibility towards its lower priority neighbors wrt ψ.
- **S**$_{all}$: Each agent A_i selects a value v from V_i which maximizes $f_{co}^{\psi}(v, N_i)$, i.e. max flexibility to all neighbors.
- **S**$_{basic}$: Each agent A_i selects a value from V_i based on min-conflict heuristic as described above.

We now define cooperativeness relation among these strategies based on the cooperativeness of values they select.

- **Definition 5**: For two different strategies S_α and S_β defined on a flexibility base ψ, S_α is *more cooperative* than S_β iff (i) for all A_i, X_i, and v_α, $v_\beta \in D_i$ such that v_α and v_β are selected by S_α and S_β respectively, $f_{co}^{\psi}(v_\alpha, N_i) \geq f_{co}^{\psi}(v_\beta, N_i)$ and (ii) for some A_i, when $f_{co}^{\psi}(v_\alpha, N_i) \neq f_{co}^{\psi}(v_\beta, N_i)$, $f_{co}^{\psi}(v_\alpha, N_i) > f_{co}^{\psi}(v_\beta, N_i)$.
- **Theorem 1**: For any given flexibility base ψ, the strategy S_{all} is *maximally cooperative* strategy in the *good* case, i.e., for any other strategy S_{other} on the same flexibility base ψ, S_{all} is more cooperative than S_{other}.
  Proof: By contradiction. Assume that S_{other} is more cooperative given the flexibility base ψ. For A_i, v_{all} is selected by S_{all} and v_{other} by S_{other} such that if $f_{co}^{\psi}(v_{all}, N_i) \neq f_{co}^{\psi}(v_{other}, N_i)$, then $f_{co}^{\psi}(v_{all}, N_i) < f_{co}^{\psi}(v_{other}, N_i)$. However, by the definition of S_{all}, v_{other} would be selected by S_{all} instead of v_{all}. A contradiction.

By theorem 1, S$_{all}$ is more cooperative than the other strategies S$_{high}$, S$_{low}$, S$_{basic}$ for the *good* case. Both S$_{high}$ and S$_{low}$ have trade-offs. For instance, S$_{high}$ may leave very little or no choice to an agent's neighbors in N_i^{low}, making it impossible for them to select any value for their negotiation variables. S$_{low}$ has a converse effect. S$_{basic}$ also has trade-offs because it does not consider neighbors' flexibility.

The above four different strategies can be also considered in the *nogood* case. The computation in the *nogood* case is identical to the *good* case except that the V_i is the set of all values in D_i, since there are no consistent values. S$_{all}$ is also the most cooperative strategy in the *nogood* case. Note that, in this *nogood* case, X$_i$'s priority is increased as usual as described above, and N_i^{high} and N_i^{low} are based on the variable's priority prior to this increase.

Based on the ideas introduced above, we can combine different negotiation strategy combinations for the *good* and *nogood* cases. Thus, for each flexibility base ψ, there are 16 possible strategy combinations from the four negotiation strategies(S$_{high}$, S$_{low}$, S$_{all}$, and S$_{basic}$ described above) for

```
Procedure check_agent_view    // for a strategy S_α-S_β
1. Propagate constraints from neighbor agents ($LC_i$ may be
   changed by constraint propagation);
2. Check constraints violation for local constraint ($LC_i$) and external
   constraints ($C_{ij}$) with higher priority neighbors;
   • If there exists any violation,
   (a) Find a value set $D_{co} \subseteq D_i$ whose values are consistent;
   (b) If $D_{co} \neq \emptyset$    // good case
       – new_cooperative_value($\alpha$, $D_{co}$);
   (c) Else    // nogood case (no consistent value in $D_i$)
       – Record and communicate nogood;
       – $X_i$'s priority = max of neighbors' priorities + 1;
       – new_cooperative_value($\beta$, $D_i$)
   (d) If there exists a change for $X_i$, communicate it to neighbor
       agents;

Procedure new_cooperative_value (Input: strategy $\sigma$, domain $\Delta \subseteq$
$D_i$; Output: $X_i$'s new value $v_{new}$)
• If $\sigma \equiv$ basic,
  1. select $v_{new} \in \Delta$ where $v_{new}$ minimizes the number of con-
     straint violation with lower priority agents;
• Else ($\sigma \in$ {high, low, all})
  1. $\Omega = N_i^{\sigma}$ (Here, $N_i^{all} \equiv N_i$);
  2. For each value $v \in \Delta$, $v$'s flexibility = $f_{co}^{\psi}(v, \Omega)$;
  3. Find $v \in \Delta$ which has max flexibility;
     – Apply min-conflict heuristic to break ties;
  4. Set the selected $v$ to $v_{new}$;
```

Figure 3: Cooperative NVA strategy in AWC framework

the *good* case and *nogood* cases. Since, henceforth, we will only consider strategy combinations, we will refer to them as strategies for short. Note that all the strategies are enhanced with argumentation (constraint propagation): *indeed, except for S_{basic}, these strategies cannot be applied without argumentation*. Here, two exemplar strategies are listed:

- **S**$_{basic}$-**S**$_{basic}$: This is the original AWC. Min-conflict heuristic is used for the *good* and *nogood* case.
- **S**$_{low}$-**S**$_{high}$: For the *good* case, an agent is maximally cooperative towards its lower priority neighbor agents by using S$_{low}$ (the selected value doesn't violate the constraints with higher neighbors). On the contrary, for the *nogood* situations, an agent attempts to be maximally cooperative towards its higher priority neighbors by using S$_{high}$.

In the next section, we systematically experiment with all the 16 strategies for two flexibility bases (ψ = sum or min). Sum was selected since defining S$_{all}$ (or S$_{high}$ or S$_{low}$) on this flexibility base implies selecting a value v that maximizes the sum of number of consistent values for neighbors N_i (or N_i^{high} or N_i^{low}). However, in some circumstances, the distribution of the numbers of consistent values over N_i may be highly non-uniform. Thus, sum as the flexibility base may offer high flexibility to some neighbors, but very low flexibility to others. Selecting ψ to be min is intended to alleviate this potential problem. That is, S$_{all}$ (or S$_{high}$ or S$_{low}$) defined using min will choose a value that maximizes the minimum number of consistent values over neighbors N$_i$.

Figure 3 provides an algorithm for an agents using a co-

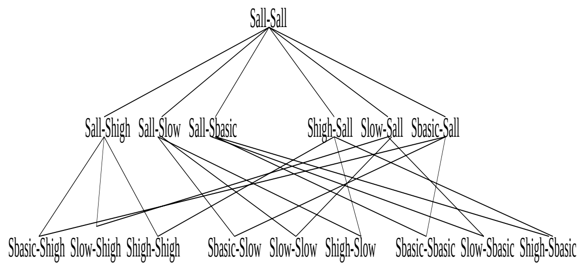


Figure 4: Cooperativeness relationship

operative negotiation strategy for value selection. While the basic algorithmic framework is from AWC, we have made three additions based on our NVA mapping: (i) incorporation of NVA strategies S$_\alpha$ and S$_\beta$; (ii) constraint communication; and (iii) constraint propagation. Check_agent_view is a procedure of AWC in which an agent checks the consistency of its value assignment with other agents' values (agent_view) and, if inconsistent, selects a new value. Cooperative negotiation strategies amount to value ordering heuristics in the procedure. For Figure 3, let's assume that A_i selects a NVA strategy S$_\alpha$-S$_\beta$ such that α, $\beta \in$ {high, low, all, basic}.

Among the cooperative strategies described above, S$_{all}$-S$_{all}$ is the most cooperative strategy because it is maximally cooperative to neighboring agents in both *good* and *nogood* cases. Figure 4 shows a partial order over the cooperativeness of 16 different strategies. A higher strategy is more cooperative than a lower one. In general, the strategies at the same level are not comparable to each other such as S$_{low}$-S$_{high}$ and S$_{high}$-S$_{low}$. However, strategies such as S$_{basic}$-S$_{basic}$ were not originally defined with the notion of cooperativeness as defined in this section; and could thus be considered less cooperative than a strategy such as S$_{low}$-S$_{high}$ that attempts to be explicitly cooperative to neighboring agents.

Experimental Evaluation

DCSP experiments in this work were motivated by the firing position case and the distributed sensor domain. In the experiments, each agent modeled a pilot whose firing position was modeled as the agent's negotiation variable. The domain of the variable was a set of firing positions restricted by local constraints such as enemy positions. There exist external constraints among negotiation variables, modeling real-world constraint that, if two pilots were neighbors, then their positions (values) must at least be some fixed distance from each other. Based on these mappings, a DCSP was constructed, and the goal of argumentation was to find values for agents' negotiation variables that satisfied all of their local and external constraints. Motivated by our two domains, two types of DCSP configurations were considered in the experiments: a chain and a grid. In the chain configuration, each agent had two neighboring agents, to its right and left (except for end points). In a grid configuration, the negotiation variables formed a grid in which a variable was constrained by its four neighbors except the ones on the grid boundary.

Our experiments followed the method used in (Yokoo & Hirayama 1998) and same criteria were used for evaluation. In particular, evaluations were performed by measuring *cycles* and *constraint checks*. *Cycles* is the number of negotiation *cycles* consumed until a solution is found, and *constraint checks* (to measure the total computation cost) is the sum of the maximal numbers of constraint checks performed by agents at each of the negotiation cycle. Experiments were performed for two sets of the 16 negotiation strategies described in "NVA Strategies" section with two different flexibility bases (ψ = sum or min). The number of agents was 512 and the domain size of each negotiation variable is one dozen for the chain and 36 for the grid. The experimental results reported below were from 500 test runs and all the problem instances were solvable with multiple solutions.

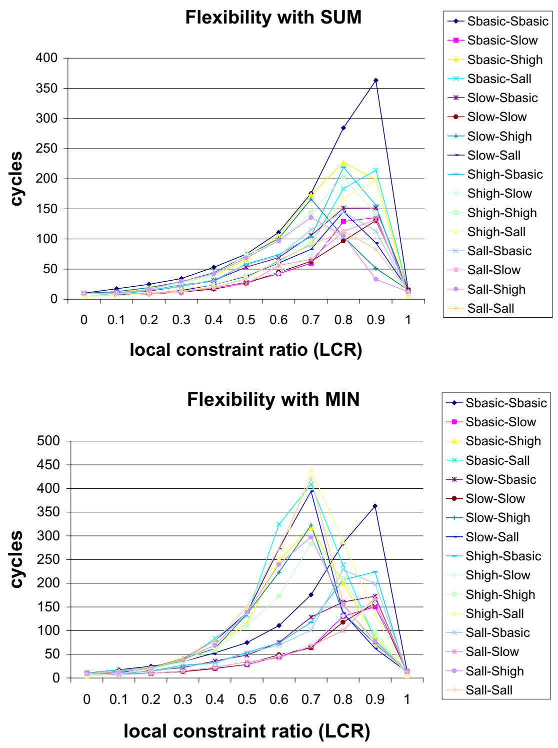
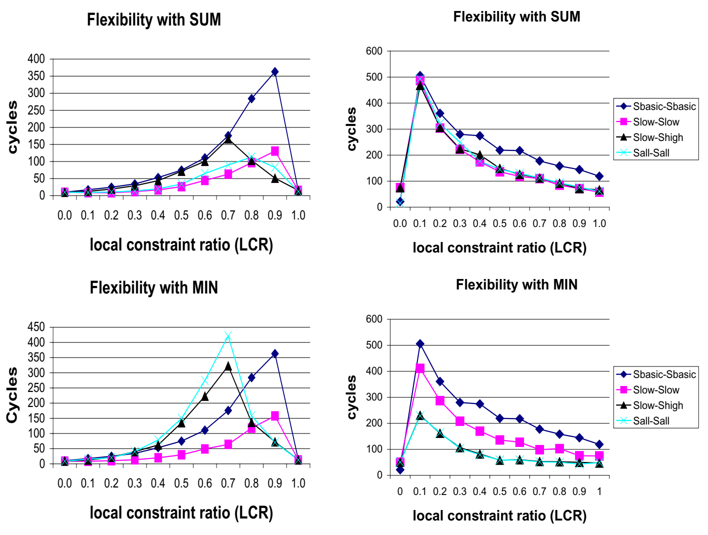


Figure 5: Comparing NVA strategies: cycles

Performance of negotiation strategies

16 NVA strategies with with two flexibility bases (ψ = sum or min) were compared on both the chain and the grid configurations. Figure 5 shows the number of *cycles* to reach a solution for all the 16 strategies with the two flexibility bases in the grid case. The horizontal axis plots the ratio of the number of locally constrained agents to the total number of agents. Each locally constrained agent has a local constraint (described in "NVA as DCSP" section) which restricts available values for its negotiation variable into a randomly selected contiguous region. Thus, for example, local constraint ratio 0.1 means that 10 percent of the agents have local constraints. Local constraint ratio will henceforth be abbreviated as LCR. The vertical axis plots the number of *cycles*. The results for all the strategies on both configurations (chain and grid) showed that S$_{low}$-S$_{low}$ or S$_{low}$-S$_{high}$ was the best, and the results also showed that those strategies with S$_{high}$ or S$_{basic}$ for the *good* case performed worse than the others.

Given 16 strategies with two flexibility bases, it is difficult to understand different patterns in Figure 5. With the same



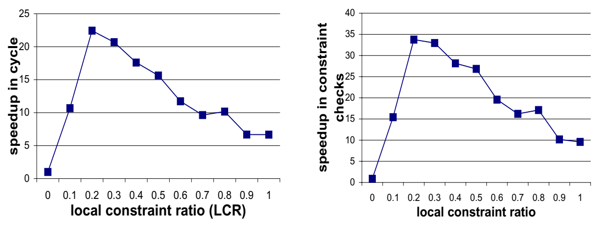
(a) Grid (b) Chain

Figure 6: Comparing NVA strategies: negotiation cycles

manner in (Jung, Tambe, & Kulkarni 2001), we will henceforth present the results from four specific strategies such as S_{basic}-S_{basic}, S_{low}-S_{low}, S_{low}-S_{high}, and S_{all}-S_{all}. Using these four strategies does not change the conclusions from our work, rather it is done solely for expository purpose.

Figure 6 shows the number of *cycles* to reach a solution for the selected strategies on both configurations described above, and both sum and min flexibility bases. These graphs show the following results: (i) cooperative strategies can improve performance, and (ii) even in the most cooperative environment, maximal cooperativeness towards neighboring agents is not the best strategy. In particular, if we focus on S_{all}-S_{all} which is the maximally cooperative strategy, it is clearly seen that S_{all}-S_{all} performs better than S_{basic}-S_{basic} in the grid and the chain (except for the LCR of 0.4 $\sim$ 0.7 in the grid with the flexibility base of min). However, S_{all}-S_{all} performs worse than other less cooperative strategies. More specifically, S_{low}-S_{low}, one of the lower level cooperative strategies from Figure 4, shows the best performance in the chain with $\psi = sum$ (except for the LCR of 0.0) and the grid (except for the LCR of 0.9 with both sum and min as a flexibility base). S_{all}-S_{all} does better in exceptional cases but, in such cases, other less cooperative strategies (e.g., S_{low}-S_{high}) are able to outperform S_{all}-S_{all}. While S_{low}-S_{low} or S_{low}-S_{high} performs better than S_{basic}-S_{basic} in terms of number of *cycles*, their computational overhead for value selection may be high. Thus, they may end up with significant slowdown in *constraint checks*. However, the graph of *constraint checks* shows the same trend with the graph of number of *cycles*, eliminating such a possibility.

The results above are surprising because, in cooperative environments, we expected that the most cooperative strategy S_{all}-S_{all} would perform the best. However, much less cooperative strategies , S_{low}-S_{low} or S_{low}-S_{high}, show the best performance. So, we conclude that a certain level of cooperativeness is useful, but even in fully cooperative settings, maximal cooperativeness is not necessarily the best



(a) negotiation cycle (b) constraint checks

Figure 7: Performance improvement from Argumentation.

NVA strategy. A related key point to note is that choosing the right negotiation strategy has significant impact on convergence. Certainly, choosing S_{basic}-S_{basic} may lead to significantly slower convergence rate while appropriately choosing S_{low}-S_{low} or S_{low}-S_{high} can lead to significant improvements in convergence. Different flexibility bases also have an impact on performance. For instance, choosing min as a flexibility base (ψ) instead of sum may degrade performance for some cooperative strategies in the grid case.

Benefits of Argumentation

One critical question to be answered was how much total amount of conflict resolution effort was saved by incorporating argumentation in negotiation, and whether the overhead of argumentation could be justified. To answer this question, two different versions of S_{basic}-S_{basic} were compared. The first version was the S_{basic}-S_{basic} described in "NVA Strategies" section. The second version was same with S_{basic}-S_{basic} except that it didn't use any argumentation (constraint propagation). That is, agents did not receive arguments from neighbors, and thus did not propagate constrain Let this second strategy be S_{basic}-S_{basic}(*noarg*). Figure 7 shows how many fold of speedup is achieved with argumentation (S_{basic}-S_{basic}) compared with non-argumentation (S_{basic}-S_{basic}(*noarg*)). The experiments were done for the chain configuration with 16 agents[3]. Argumentation helped S_{basic}-S_{basic} to reduce the total negotiation effort as measured by *cycles* (Figure 7.a) and *constraint checks* (Figure 7.b). One interesting point is that the benefit of argumentation varied non-monotonically as the LCR of the agents changed. The benefit of argumentation was the lowest when there were too few or too many locally constrained agents (e.g., at 0.0, 0.1, and 1.0). As the proportion of locally constrained agents (LCR) increased, the performance with argumentation did not monotonically improved. Thus, we can begin to offer guidance on when to actually apply argumentation for maximum benefits. Next, we provide mathematical analysis for the benefits of argumentation and the non-monotonic improvements from argumentation in the chain.

Analysis for the Benefits of Argumentation

Assuming that the time until a solution is found is proportional to the size of search space, the analysis of reduced search space from argumentation can provide the answer to the question how much effort can be saved by argumentation. For this analysis, we assume:

[3]The results for no-argumentation with larger number of agents are not shown because experiments were too slow taking an hour for an individual run with 512 agents.

1. N: the number of agents;
2. K: the number of values per domain of agent
3. α: fraction of agents that have local constraints
4. β: reduction factor in domain due to a local constraint (i.e., domain becomes K/β due to a local constraint)

Here, $\alpha * N$ agents have local constraints and their domains are reduced to K/β. Other $(N - \alpha * N)$ agents maintain their domain to be K. Thus, we have the following original search space (total number of combination of values):

$$\underbrace{K/\beta * K/\beta * ... * K/\beta}_{\alpha * N\ times} * \underbrace{K * K * ... * K}_{(N-\alpha*N)\ times} = K^N/\beta^{\alpha*N}$$

. Without argumentation, the search space for agents will remain to be the same as the original search space since agents maintain their domains without reduction during search. However, with argumentation, when there exist locally constrained agents ($\alpha \neq 0$), the domains of neighboring agents which initially have no local constraints are also reduced to K/β in size. Within K/β, we have a certain fraction (1/C) that is consistent. Thus, we have the following search space with argumentation:

$$\underbrace{(K/\beta) * (K/\beta) * (K/\beta) * ... * (K/\beta)}_{N\ times}$$

Note that, at α = 0, the search space with argumentation will be the same as the original search space at α = 0 since there are no local constraints to cause domain reduction. Therefore, the "search space without argumentation" (denoted by SP^{noarg}) and the "search space with argumentation" (denoted by SP^{arg}) will be as follows:

$$\left(\begin{array}{l} SP^{noarg} = K^N/\beta^{\alpha*N} \\ SP^{arg} = \begin{cases} K^N & \text{if } \alpha = 0 \\ K^N/\beta^N & \text{otherwise} \end{cases} \end{array} \right.$$

Now, the ratio of SP^{arg} to SP^{noarg} indicates how much effort is saved by argumentation. In the chain case, with argumentation, agents reduce search space by a fraction of SP^{arg}/SP^{noarg} as follows:

$$SP^{arg}/SP^{noarg} = \begin{cases} 1 & \text{if } \alpha = 0 \\ 1/\beta^{(1-\alpha)*N} & \text{otherwise} \end{cases}$$

This ratio also provides an answer to the question why the benefits of argumentation varied non-monotonically as the LCR of the agents changed. The ratio, SP^{arg}/SP^{noarg}, would appear to be high when α = 1, and reduce as α goes lower. (As the ratio goes lower (α goes lower), the benefit from argumentation increases.) Thus, we can explain why Figure 7 (where LCR is α in our equation) shows greater speedup in lower LCR's (with large reduction of search space), and little speedup when LCR = 1.0 or 0.0.

Related Work

While this paper builds on several previous efforts in argumentation(Kraus, Sycara, & Evenchik 1998; Parsons & Jennings 1996) and distributed constraint satisfaction(Yokoo & Hirayama 1998), it is a unique effort in synthesizing these two areas. Argumentation has been rigorously investigated using different logics including specially designed logics of argumentation(Kraus, Sycara, & Evenchik 1998; Sawamura, Umeda, & Meyer 2000). Some of these efforts focus on formal modeling of agents' detailed mental states, or specific techniques for resolving conflicts in argumentation (e.g., defining defeat in argumentation). Unfortunately, such formalization appears too detailed and computationally complex to be suitable for empirical investigation of the effects of argumentation on large-scale conflict resolution convergence. Furthermore, these efforts have not focused on formalizing different collaborative NVA strategies or empirically investigating the impact of such strategies. Indeed, we are unaware of experimental investigations in large-scale convergence using such logical frameworks of argumentation. In contrast, we have built on constraint propagation in DCSP in modeling argumentation. We have also investigated different collaborative NVA strategies using value ordering in this framework. Thus, by avoiding detailed modeling of individual agents' mental states, and by building on highly efficient DCSP algorithms, we enable systematic experimental investigation of the computational properties of argumentation systems in the large-scale.

Computational complexity of argumentation/negotiation has been studied in logics(Dimopoulos, Nebel, & Toni 1999; Wooldridge & Parsons 2000). However, these studies focused on the worst case complexity analysis in computing arguments or determining if agents can reach an agreement, and did not investigate the formalization of NVA strategies and their impact on convergence. While some computational models for negotiation strategies have been offered, e.g., (Matos, Sierra, & Jennings 1998), these efforts focus on non-collaborative strategies, do not focus on either investigating argumentation or scale-up.

Our work has built on the foundations of the existing DCSP work(Armstrong & Durfee 1997; Yokoo & Hirayama 1998). Our ability to experimentally investigate argumentation and NVA strategies is a testimony to the effectiveness of using DCSP as a computational model. We have modeled argumentation as constraint propagation(Kumar 1992), and negotiation strategies as value ordering heuristics. Our NVA strategies which interleave value selection with argumentation can be seen as the integration of distributed local search and constraint propagation. In DCSP, distributed versions of centralized constraint propagation techniques have been studied(Hamadi 1999). However, they are applied as a preprocessor rather than being interleaved with search. While our incorporation of argumentation into DCSP came about in service of studying NVA, they appear to be useful as possible enhancements to DCSP algorithms.

If we view cooperative NVA strategies from purely centralized perspective, they amount to one-step look-ahead value ordering(Frost & Dechter 1995). In centralized CSP, (full) look-ahead value ordering ranks values by some heuristic functions which determine the impact of each value on the domains of all future (uninstantiated lower order) variables. However, since there is no global variable ordering in DCSP, look ahead value ordering technique can not be directly applied to DCSP. With our mapping of argumentation onto DCSP as constraint propagation, agents can do value ordering based on the flexibility given to neighbors by its values. With constraint propagation, agents also take non-neighboring agents into account to some extent since their constraints are propagated into neighbors' local constraints.

While resource allocation is itself a broad area of research, our use of argumentation for resource conflict resolution and the use of enhanced DCSP for modeling such conflict resolution sets our work apart. For instance, (Liu & Sycara 1996) extends dispatch scheduling to improve resource allocation; and (Chia, Neiman, & Lesser 1998) on distributed airport-ground scheduling service. While these systems do not use argumentation, hopefully our research will begin to shed light on the utility of argumentation in these domains.

Finally, while this paper builds on the previous conference paper (Jung, Tambe, & Kulkarni 2001), this paper extends the flexibility function with flexibility bases and provide experimental results for this extension. The analysis of the performance improvement from argumentation is added as well.

Conclusion and Future Work

Argumentation is an important conflict-resolution technique in multi-agent research. However, much of the existing work has focused on smaller-scale systems, and major questions regarding the computational performance of large-scale collaborative argumentation and different NVA strategies remain unaddressed. Yet it is difficult to answer these questions with ad-hoc implemented argumentation systems or complex and detailed logical frameworks.

To address these issues, we provided a novel computational model of NVA as DCSP. Since, this model exploits existing efficient DCSP techniques, and efficiently incorporate argumentation as constraint propagation, it appears better suited for conducting large-scale experiments to investigate computational performance of argumentation. The key contributions of this paper are: (1) modeling of argumentation in terms of constraint communication and constraint propagation in DCSP; (2) formalizing and investigating different cooperative NVA strategies; (3) conducting large-scale experiments that quantitatively measure the performance of argumentation and different NVA strategies. These experiments illustrate that argumentation can lead to significant speedup in convergence of conflict resolution. Our experiments with NVA strategies illustrate that choosing the right strategy can lead to very significant improvement in rate of convergence. The experiments also reveal a surprising result: even in a fully cooperative setting, the most cooperative argumentation strategy is not the best in terms of convergence in negotiation. These results can help guide the development of real-world multi-agents systems. Finally, key ideas from argumentation, such as cooperative response, could feed back into improvements in existing DCSP algorithms.

Among topics for future work, one important topic is to extend the current NVA mapping onto DCSP in several ways. First, we wish to focus on multiple negotiation variables. Since existing body of AWC can handle multi-variables per agent, this extension may be straightforward. Second, constraint propagation as investigated in this paper does not model all forms of argumentation. For instance, some types of arguments may lead to shift in the NVA strategies, or may add new values to variables' domains. An additional topic is to mathematically understand the performance difference between strategies. Backing up our experimental results with mathematical analysis may help us understand different NVA strategies in more general settings.

References

Armstrong, A., and Durfee, E. 1997. Dynamic prioritization of complex agents in distributed constraint satisfaction problems. In *Proceedings of IJCAI*.

Barrett, A. 1999. Autonomy architectures for a constellation of spacecraft. In *Proceedings of Int'l Symposium on AI Robotics and Automation in Space*.

Chia, M.; Neiman, D.; and Lesser, V. 1998. Poaching and distraction in asynchronous agent activities. In *Proceedings of ICMAS*.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 1999. Preferred arguments are harder to compute than stable extensions. In *Proceedings of IJCAI*.

Frost, D., and Dechter, R. 1995. Look-ahead value ordering for constraint satisfaction problems. In *Proc. of IJCAI*.

Hamadi, Y. 1999. Optimal distributed arc-consistency. In *Proc. of Int'l Conf. on Principles and Practice of Constraint Programming*.

Jennings, N. R.; Faratin, P.; Lomuscio, A. R.; Parsons, S.; Sierra, C.; and Woodlidge, M. 2001. Automated negotiation: Prospects, methods and challenges. *J. of Group Decision and Negotiation* 10(2).

Jung, H.; Tambe, M.; and Kulkarni, S. 2001. Argumentation as distributed constraint satisfaction: Applications and results. In *Proc. of Int'l Conf. on Autonomous Agents*.

Kraus, S.; Sycara, K.; and Evenchik, A. 1998. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence* 104.

Kumar, V. 1992. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine* 12(1).

Liu, J., and Sycara, K. 1996. Multiagent coordination in tightly coupled task scheduling. In *Proc. of ICMAS*.

Matos, N.; Sierra, C.; and Jennings, N. 1998. Determining successful negotiation strategies: An evolutionary approach. In *Proceedings of ICMAS*.

Minton, S.; Johnston, M. D.; Philips, A.; and Laird, P. 1990. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proc. of AAAI*.

Parsons, S., and Jennings, N. R. 1996. Negotiation through argumentation — a preliminary report. In *Proc. of ICMAS*.

Rickel, J., and Johnson, W. L. 1997. Pedagogical agents for immersive training environments. In *Proc. of Int'l Conf. on Autonomous Agents*.

Sawamura, H.; Umeda, Y.; and Meyer, R. 2000. Computational dialectics for argument-based agent systems. In *Proc. of ICMAS*.

Scerri, P.; Pynadath, D.; and Tambe, M. 2001. Adjustable autonomy in real-world multi-agent environments. In *Proc. of Int'l Conf. on Autonomous Agents*.

Tambe, M., and Jung, H. 1999. The benefits of arguing in a team. *AI Magazine* 20(4).

Tambe, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7.

Tessier, C.; Chaudron, L.; and Muller, H. J., eds. 2000. *Conflicting Agents*. Kluwer Academic Publishers.

Toulmin, S. 1958. *The uses of argument*. London: Cambridge Univ Press.

Wooldridge, M., and Parsons, S. 2000. Languages for negotiation. In *Proc. of ECAI*.

Yokoo, M., and Hirayama, K. 1998. Distributed constraint satisfaction algorithm for complex local problems. In *Proc. of ICMAS*.