Argumentation as Distributed Constraint Satisfaction: Applications and Results

Hyuckchul Jung, Milind Tambe, Shriniwas Kulkarni
University of Southern California/Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292, USA
{jungh,tambe,kulkarni}@isi.edu

ABSTRACT

Conflict resolution is a critical problem in distributed and collaborative multi-agent systems. Negotiation via argumentation (NVA), where agents provide explicit arguments or justifications for their proposals for resolving 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 investigating NVA. We model argumentation as constraint propagation in DCSP. This model enables us to study convergence properties of argumentation, and formulate and experimentally compare 16 different NVA strategies 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. The paper illustrates the usefulness of these results in applying NVA to multi-agent systems, as well as to DCSP systems in general.

1. INTRODUCTION

Distributed, collaborative agents[4, 12] are promising to play an important role in large-scale multi-agent applications including virtual environments for training, distributed robots for exploration, and distributed sensors. Such collaborative agents may enter into conflicts over their shared resources, joint plans, or task assignments, etc. requiring effective collaborative conflict resolution. Indeed, resolving such conflicts is a critical issue for collaborative agents, particularly for large-scale applications.

Negotiation via argumentation (NVA) is a promising approach to collaborative conflict resolution[6]. 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). Such argumentation appears particularly appropriate in collaborative settings, since agents need not hide information from each other. Furthermore, revealing this information is *hypothesized* to speed up the rate and likelihood of converging to a solution[5]. Indeed, we are currently applying argumentation in real-world multi-agent settings, which require scale-up to large numbers of agents.

Unfortunately, while previous implemented argumentation systems have performed well in small-size applications, no systematic investigation on large-scale argumentation systems has been done. Thus, 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, particularly in the face of scale-up. Indeed, 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. This question is particularly important in collaborative contexts, since well-formulated strategies from non-collaborative settings, such as threats, appeals to self interest or attempts to undercut one's opponent[6], are inapplicable.

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 ad-hoc, complex agent argumentation systems is costly and very labor intensive. Furthermore, such complex systems often make it difficult to identify the critical factors that contributed to their success or failure. Another alternative is to exploit existing formalizations of argumentation in logic, such as modal logic[6] and dialectical logic[11]. 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. Indeed, to the best of our knowledge, these formalizations have not been used in investigating large-scale argumentation systems.

To alleviate the above difficulties, this paper proposes distributed constraint satisfaction problem (DCSP)[1, 15] as a novel computational model of NVA. Argumentation is modeled in DCSP as follows: when an agent communicates to others its assignments of its local variables, it also includes the local constraints that led to the assignments, as a justification. These communicated local constraints are exploited in service of constraint propagation by other agents to attempt to speed up a conflict resolution process. We focus specifically on one of the best published DCSP algorithms, that of Yokoo and Hirayama [15], 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.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

Next, using this extended DCSP as our computational model, we formulate different NVA strategies, varying the level of cooperativeness towards others. While cooperativeness towards others would appear to be fundamentally important in a cooperative environment, the DCSP model enables a formalization of this notion. We specifically formulate different negotiation strategies as varying the value ordering heuristics[3]. 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.

Essentially, the existing body of work on efficient DCSP algorithms and further efficient incorporation of argumentation in DCSP enables us to investigate the impact of argumentation and different NVA strategies in the large-scale. Furthermore, we can vary different argumentation parameters and investigate their impact on convergence. In particular, we conduct detailed experiments on 16 different NVA strategies, and provide the following results. First, argumentation can indeed significantly improve agents' conflict resolution convergence, i.e., agents can more quickly resolve their conflicts and the overhead of argumentation is in general outweighed by its benefits. However, the benefits of argumentation vary non-monotonically with the proportion of agents that offer tightly constraining arguments to support their proposals — indeed, the benefits of argumentation are lowest when either too few or too many agents offer such arguments. Second, with respect to NVA strategies, given that our system operates in 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. Essentially, while some improvements in cooperativeness significantly improve performance, further improvements do not help and may end up degrading performance. This degradation is not only in terms of overheads but more fundamentally in negotiation cycles required to converge to a solution.

Additional benefits of the DCSP model of argumentation are seen in our ability to directly apply this model in real-world multi-agent settings, where it can provide sound performance guarantees (based on the guarantees of DCSP). Finally, notions of argumentation and cooperative NVA strategies may potentially advance the state of the art in DCSP research as well.

2. DOMAINS AND MOTIVATIONS

Among the domains that motivate this work, the first is a distributed sensor domain. This domain consists of multiple stationary sensors, each controlled by an independent agent, and targets moving through their sensing range (Figure 1.a and Figure 1.b illustrates the real hardware and simulator screen, respectively). 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. (This allows the target's position to be triangulated). 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 choice of sectors. 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

(a) sensor(left) and target(right) (b) simulator (top-down view)

Figure 1: A distributed sensor domain

C is busy with another target. Thus, if agent A is able to provide an argument to B and C for its request, such as "I detect a target in my sector 1", it may induce them to switch sectors. Alternatively, B may counter-propose that it cannot turn on its sector, with an argument "low on power". Here, following [5], *negotiation objects* refers to issues (e.g. choice of sectors) over which negotiation takes place. Agents propose and counter-propose values for these negotiation objects, with explicit justifications.

The second application domain is the helicopter combat simulation domain[12]. Different conflict situations arise in a team of simulated pilot agents. One example, henceforth called the *firing position example*, 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 enemies 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.

Earlier an argumentation system called CONSA (COllaborative Negotiation System based on Argumentation) was developed for the above combat simulation domain[13]. As an example of argumentation in CONSA, consider two pilot agents, A1 and A2, and two enemy positions E1 and E2, where A1 knows only about E1, while A2 knows only about E2. Suppose that A1 and A2 are only 100 meters apart, which are in conflict since they are not one kilometer apart as required. Here, agents' firing positions are the negotiation objects. A1 computes new values for the negotiation objects (positions for both agents). It then communicates its proposal to A2, suggesting {(*A1 move 450 m left, A2 move 450 m right*)}, where the appended justification includes {(*enemy E1 position, current separation 100 m,...*)}. When A2 receives and evaluates the proposal, it realizes that it cannot move 450 meters right because of E2. Instead, since the maximum A2 can move is 300 meters, it counter-proposes {(*A1 move 600 m left, A2 move 300 m right*)}, with the justification being that {(*enemy E1 position, enemy E2 position,...*)}. However, if there were a third agent A3, A1's 600 meter move to the left may cause a conflict between A1 and A3, requiring further negotiation.

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 100s of distributed sensors in a grid or 100s 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) the argumentation will truly speed up conflict resolution convergence with scale up. In addition, it is important 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. Indeed, in some cases, the pilot agents' maximal cooperativeness towards others, by offering to move the maximal distance they are allowed, would appear to be very helpful. Similarly in distributed sensors, an agent's turning its own sector on so that others may conserve power, would appear to be helpful. However, as we scale up the number of agents, it is unclear if maximal cooperativeness will necessarily lead to improved performance. Unfortunately, answering these questions by building adhoc implementations is difficult — the process would be difficult and labor intensive, and in the end, the factors that led to success or failure of argumentation may remain unclear.

Finally, it is useful to understand that, in this paper, we focus on distributed NVA to resolve conflicts as opposed to a centralized approach, where a single agent gathers all information to provide a solution. Indeed, in many applications, such a centralized approach could prove problematic. First, this approach introduces a central point of failure, so that there is no fault tolerance. Second, centralization of all information could be a significant security risk, open to actual physical or cyber-attacks, particularly in hostile adversarial environments. Third, centralization requires all agents to accept a central authority, which may not always be feasible. Finally, a centralized agent could be a significant computational and communication bottleneck. Specifically, in domains such as distributed sensors, negotiations must continually occur among all agents for readjustment of the sensors. Centralization would require all sensors to continuously communicate their local information to the centralized agent which can be a significant bottleneck given scale-up to thousands of agents. A distributed system provides fault tolerance, reduces the security risk, avoids a central authority and avoids a centralized communication/computational bottleneck.

3. ARGUMENTATION AS DCSP

To advance the current research, we need to provide an abstract and well-understood computational model for NVA. To this end, we propose a novel computational model, that of Distributed Constraint Satisfaction Problem (DCSP)[1, 15] to investigate NVA. 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.*

3.1 Description of Computational Model

A Constraint Satisfaction Problem (CSP) is commonly defined by a set of n variables x_1, ..., x_n associated with finite domains D_1, ..., D_n respectively, and a set of k constraints C_1, ..., C_k on the values of the variables. A solution is the value assignment for the variables which satisfies all the constraints. A distributed CSP is a CSP in which variables and constraints are distributed among multiple agents. We consider DCSPs with multiple variables per agent[15]. Each variable (x_i) belongs to an agent A_j. A constraint defined only on variables belonging to a single agent is called a *local constraint*. In contrast, an *external constraint* involves variables of different agents. Solving a DCSP requires that agents not only solve their local constraints, but also communicate with other agents to satisfy external constraints. DCSP is not concerned with speeding up a centralized CSP via parallelization[15]; 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, agents' 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. The remaining variables belonging to an agent are only locally constrained, and are referred to as *local variables*. These local variables model the facts that are (at least initially) only known locally by an agent, i.e., not known by other agents. An agent's local variables locally constrain its negotiation variables. In our initial experimental investigations in DCSP, we found it sufficient to model each agent as having only one negotiation variable. We represent all the local variables and local constraints as a single node constraint 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 and one local constraint LC_i. Note that the local constraint often involves very complex computation. Furthermore, there is no limitation on the number of external constraints C_{ij} for each agent.

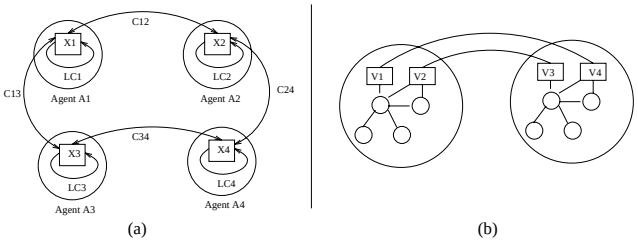


Figure 2: Model of agents in argumentation

We can now begin to model the firing position example using DCSP. Here, each helicopter pilot's firing position is its single negotiation variable which is known to other agents. Furthermore, there are external constraints among the negotiation variables of neighboring agents: the firing positions must be at least 1000 meters apart. Since enemy positions are locally observed by individual agents and they constrain the firing positions, the enemy positions form local constraints on an agent's negotiation variable. Note that we can easily extend this mapping to problems involving multiple negotiation variables per agent, as shown in Figure 2.b. 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 constraint propagation. That is, in DCSP algorithms such as Asynchronous Weak Commitment search (AWC) [15], agents communicates the values assigned to their externally constrained negotiation variables. However, with argumentation, agents also communicate their argument (justification) 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 variables. These local constraints are propagated by the agents receiving the argument. Such constraint propagation requires that when negotiation variables X_i and X_j (belonging to agents A_i and A_j respectively) share an external constraint $C_{ij}(X_i, X_j)$ then these agents know the domains of the negotiation variables. In particular, A_i is aware of X_j's domain, and A_j is aware of X_i's domain; otherwise the communication of local constraints such as LC_i may not be interpretable by others. This assumption models the situations where our pilot agents are aware of the overall region or set of positions that neighboring pilots could take, but they do not know the local constraints that restrict those positions. (Alternatively, the local constraint LC_i may explicitly outline the set of allowed values for the given negotiation variable. There are tradeoffs in the different techniques, and these may be optimized based on the domain.)

Concretely, argumentation in DCSP works as follows. Suppose an agent A_i selects a value v_i for its negotiation variable X_i. It

will then send its selection v_i and its local constraint LC_i to its neighboring agent A_j with the negotiation variable X_j. Here we assume agents A_i and A_j are connected by the external constraint $C_{ij}(X_i, X_j)$ where X_i has domain D_i and X_j has domain D_j. After receiving information from A_i, agent A_j will propagate the received constraint to reduce X_j's domain D_j. This may be accomplished as follows. A_j first infers D_i' by applying LC_i to X_i (alternatively, LC_i directly provides the values of D_i'), and A_j then uses D_i' to change its domain D_j to D_j' by applying the external constraint $C_{ij}(X_i, X_j)$. A_j modifies its local constraint LC_j to reflect the domain change (the modified LC_j is communicated to A_j's neighbors with v_j later).

While this constraint propagation amounts only to arc consistency, it is not run by itself to solve the DCSP, rather it is interleaved with value selection. For instance, during each cycle of AWC, we first propagate communicated constraints and then select values for variables. Thus, we do not increase the number of communicated messages, an important issue for DCSP. Here, one assumption is that communication cost is mainly dependent on the number of communications rather than message size. Hence, communication of local constraints is not counted as extra communication cost.

3.2 Applications of Model

The DCSP based computational model of argumentation can be applied to the firing position example and the distributed sensor network. We have already seen the mapping to the pilot agents with their firing positions as negotiation variables and enemy positions as local constraints in Section 3.1. Here, argumentation enables other pilots to quickly rule out incompatible firing positions from their domains. With respect to the distributed sensor network, we desire that whenever a target is detected by an agent, all neighboring agents turn on overlapping sectors so as to also detect this target. We model this as follows: Each sensor A_i has a negotiation variable, $sector_i$, with domain {0,1,2,don't care}. Intuitively, the choice of value of this variable corresponds to A_i's choice to make that particular sector active or to go in a "dont'care" mode. *'Overlaps'* is an external constraint over sector variables of two agents, A_i and A_j. Thus, overlaps($sector_i$, $sector_j$) is true iff $sector_i$ and $sector_j$ cover an overlapping region of space or anyone has a value "don't care". Furthermore, each agent has a local constraint, $sector_i = \mu$, on its sector value. This constraint is enforced when a target is detected in sector μ by A_i. "don't care" has other constraints that we will not go into here. This model is an initial mapping of the distributed sensor network. As we take more resources and testing environment into account (e.g., power, bandwidth, noise, etc.), the mapping can develop to include them.

Argumentation leads to constraint propagation in this domain as follows. Suppose A_i detects a target in sector 0. A_i's local constraint on its sector value ($sector_i = 0$) will be enforced preventing it from changing its sector to something else. Since $sector_i$ is a negotiation variable, A_i will send the value assigned to this variable, along with the local constraint ($sector_i = 0$) as an argument, to all its neighbors such as A_j. A_j will then process the argument (propagate A_i's local constraint on its own domain), eliminating non-overlapping sector values, e.g., "0" from its own domain. Thus, A_j may be left with {1, 2, don't care}. A_j may select value 2 for its sector, overlapping with A_i. It may then announce its choice of value and reduced domain to its neighbors.

While constraint propagation caused by argumentation reduces an agents' domain, there is still a choice of values available. Thus, a key question that remains open is how should an agent choose a value for its negotiation variable from its reduced domain? This issue leads us to consider different negotiation strategies discussed in the next section.

4. NEGOTIATION STRATEGIES

Given the mapping of argumentation to DCSP presented in the previous section, different NVA strategies can be formalized using DCSP. A negotiation strategy refers to the decision function used by an agent to make a proposal or counter-proposal. In the mapping to DCSP, a negotiation strategy is modeled as a value ordering used to choose a value of an assignment for a negotiation variable. A value ordering heuristic ranks the value of variables[3]. Different value ordering heuristics lead to different negotiation strategies.

In AWC[15], 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[10]. This min-conflict heuristic is used as a baseline negotiation strategy and we refer it as S_{basic}. Choosing S_{basic}, a state of the art strategy as baseline is critical to ensure that any improvements we suggest are real. However, the S_{basic} strategy doesn't exploit argumentation in generating a more cooperative response to other agents. Argumentation enables agents to consider the constraints that the neighboring agents have on their domains, which are communicated as arguments. Taking the domains of neighboring agents into account 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 agent N_i.

- **Definition 1**: For a value $v \in D_i$ and a set of agents $N_i^{sub} \subseteq N_i$, *flexibility function* is defined as $f_{co}(v, N_i^{sub}) = \Sigma_j c(v, A_j)$ such that $A_j \in N_i^{sub}$ and $c(v, A_j)$ is the number of values of X_j that are consistent with v. [1]
- **Definition 2**: For a value v of X_i, *cooperativeness* of v is defined as $f_{co}(v, N_i)$. That is, the *cooperativeness* of v measures how much flexibility (choice of values) is given to all of A_i's neighbors by v.
- **Definition 3**: A *maximally cooperative* value of X_i is defined as v_{max} such that, for any other value $v_{other} \in D_i$, $f_{co}(v_{max}, N_i) \geq f_{co}(v_{other}, N_i)$.

Here, 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 indeed partly the rationale for the helicopter pilots in the firing position example (described in Section 2) 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., $f_{co}(v, N_i) \leq f_{co}(v_{max}, N_i)$. Hence, S_{basic} is not the most cooperative strategy to neighboring agents. However, other cooperative strategies can be introduced and formalized in terms of value ordering; i.e.,

[1] One objection to using a sum (Σ) is that a value v with higher sum may be selected even if some constituent X_j are inconsistent — sum may be unfair to some neighboring agents. However, because of our constraint propagation, we guarantee that such inconsistency does not arise, reducing any potential unfairness.

an agent A_i can rank each value (v) in its domain D_i based on the cooperativeness $f_{co}(v, N_i)$ of that value. These strategies rely on the basic framework from AWC.

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 the AWC framework, 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 increases its priority and uses S_{basic} to minimize constraint violations over all neighboring agents[15].

Different negotiation strategies are described in terms of the *good* and *nogood* cases because different value ordering methods can be applied in these two cases. To explain the negotiation strategies, let 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. Based on cooperativeness, four different negotiation strategies can be considered in the *good* case as follows:

- $\mathbf{S}_{high}$: each agent A_i selects a value v from V_i which maximizes $f_{co}(v, N_i^{high})$ i.e., A_i attempts to give maximum flexibility towards its higher priority neighbors.
- $\mathbf{S}_{low}$: each agent A_i selects a value v from V_i which maximizes $f_{co}(v, N_i^{low})$.
- $\mathbf{S}_{all}$: each agent A_i selects a value v from V_i which maximizes $f_{co}(v, N_i)$, i.e. max flexibility to all neighbors.
- $\mathbf{S}_{basic}$: 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 4**: For two different strategies S_α and S_β, 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, $f_{co}(v_\alpha, N_i) \geq f_{co}(v_\beta, N_i)$ and (ii) for some A_i, when $f_{co}(v_\alpha, N_i) \neq f_{co}(v_\beta, N_i)$, $f_{co}(v_\alpha, N_i) > f_{co}(v_\beta, N_i)$.
- **Theorem 1**: The strategy S_{all} is *maximally cooperative* strategy in the *good* case, i.e., for any other strategy S_{other}, S_{all} is more cooperative than S_{other}.
  Proof: By contradiction. Assume that S_{other} is more cooperative. For A_i, v_{all} is selected by S_{all} and v_{other} by S_{other} such that if $f_{co}(v_{all}, N_i) \neq f_{co}(v_{other}, N_i)$, then $f_{co}(v_{all}, N_i) < f_{co}(v_{other}, N_i)$. 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 the flexibility of neighboring agents.

[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.

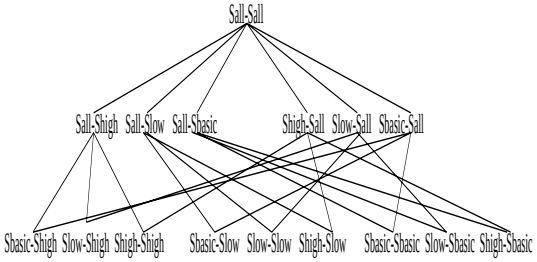


Figure 3: Cooperativeness relationship

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 isn't a set of 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: there are 16 possible strategy combinations from the four negotiation strategies(S_{high}, S_{low}, S_{all}, and S_{basic} above) for the *good* case and *nogood* cases. Since, henceforth, we will only consider strategy combinations, we will refer to them as strategies for short. In the following, some examples of NVA strategies used for experiments are described. Each strategy below is described in terms of its response in the *good* and *nogood* cases. Note that all the strategies are enhanced with argumentation (constraint propagation): indeed, except for S_{basic}, these strategies cannot be applied without argumentation. In the next section, we systematically experimented with all the 16 strategies. Here, three exemplar strategies are listed.

- $\mathbf{S}_{basic}$-$\mathbf{S}_{basic}$: This is the original AWC. Min-conflict heuristic is used for the *good* and *nogood* case.
- $\mathbf{S}_{low}$-$\mathbf{S}_{high}$: 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}.
- $\mathbf{S}_{all}$-$\mathbf{S}_{all}$: In both the *good* and the *nogood* cases, an agent uses S_{all} for the value ordering, which is to select a value that maximizes flexibility of all neighbor agents.

Figure 4 describes a cooperative negotiation strategy chosen by an agent A_i in AWC[15] framework. 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 selects a new value if it violates any constraint. Cooperative negotiation strategies amount to value ordering heuristics in the procedure. For Figure 4, let's assume that A_i selects a negotiation strategy S_α-S_β such that $\alpha, \beta \in$ {high, low, all, basic}. In the new_cooperative_value procedure (Figure 4), S_{high}, S_{low}, and S_{all} use min-conflict heuristic to break ties among the values with the same max flexibility.

Among the cooperative strategies described above, S_{all}-S_{all} is the most cooperative strategy because it is maximally cooperative to neighboring agents by S_{all} in both *good* and *nogood* cases. Figure 3 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.

Procedure **check_agent_view** // for a strategy S_α-S_β

1. Propagate constraints from neighbor agents (D_i may change);
2. Check constraints violation for local (LC_i) constraint and external constraints(C_{ij}) with higher priority neighbor agent A_j;
   If there is any violation,
   {Find a value set $D_{co} \subseteq D_i$ whose values are consistent;
   If $D_{co} \neq \emptyset$ // *good* case
   new_cooperative_value(α, D_{co})
   check_agent_view;
   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(β, D_i)
   check_agent_view;}
   Else // no violation
   {If there exists a change for X_i, communicate it to neighbor agents;}

Procedure **new_cooperative_value** (Input: strategy σ, domain $\Delta \subseteq D_i$; Output: X_i's new value v_{new})

1. If $\sigma \equiv$ basic, select $v_{new} \in \Delta$ where v_{new} minimizes the number of constraint violation (min-conflict) with lower priority agents;
2. Else ($\sigma \in$ {high, low, all})
   $\Omega = N_i^\sigma$ (Here, $N_i^{all} \equiv N_i$);
   for each value $v \in \Delta$
   v's flexibility = sum of flexibility for each $A_j \in \Omega$ given by v;
   find $v \in \Delta$ which has max flexibility and set the selected v to v_{new};

Figure 4: Cooperative negotiation strategy in AWC framework

5. EXPERIMENTAL EVALUATION

DCSP experiments in this work were motivated by the firing position example and the distributed sensor domain. In the experiments, each pilot was modeled as an agent and each agent had one negotiation variable to model the pilot's firing position. The domain of this variable was the set of positions that the agent could take. However, the domain was restricted by local node constraints such as the enemy positions for firing position example. The negotiation variable also had external constraints with the negotiation variables of its neighboring agents. This external constraint modeled the real-world constraint that, in firing position example, if two pilots were neighbors, then their positions must at least be some fixed distance from each other. If this external constraint was violated, then clearly, the current values of the agents were in conflict with each other. This mapping also works well for the distributed sensor domain, where each sensor is modeled as an agent whose negotiation variable takes on sector values, and is constrained by local constraints from target position, power usage, etc. The external constraint for the variables modeled the overlap constraint explained in Section 3.2.

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. Two different 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). Since there was a constraint between the negotiation variables of neighboring agents, the negotiation variables essentially formed a chain. 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. These configurations were motivated by our two domains.

Our experiments followed the method used in [15] 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 the 16 negotiation strategies described in Section 4. The

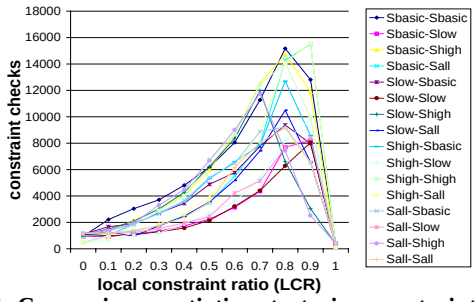


Figure 5: Comparing negotiation strategies: constraint checks

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 [3].

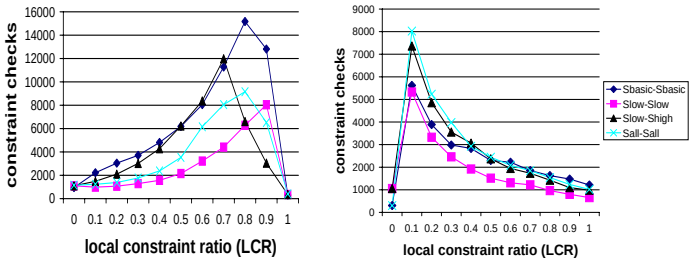
5.1 Performance of negotiation strategies

NVA strategies described in Section 4 were compared on both the chain and the grid configurations. In Figure 5, *constraint checks* in the grid is shown for all the 16 strategies. 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 Section 3) 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. Having local constraints, agents have less choice to assign a value to their negotiation variables. The vertical axis plots the number of *constraint checks*. The results for all the 16 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, it is difficult to understand different patterns in Figure 5. For expository purposes, we will henceforth present the results from four specific strategies. First, S_{all}-S_{all} is selected because it is the maximally cooperative strategy. Second, the original AWC strategy (S_{basic}-S_{basic}) is selected to compare it with other negotiation strategies. Third, S_{low}-S_{low} is selected because it showed the best performance overall in the chain and the grid. Lastly, S_{low}-S_{high} is selected because it performed better than S_{low}-S_{low} in a special case. Using these four strategies does not change the conclusions from our work, rather it is done solely for expository purpose.

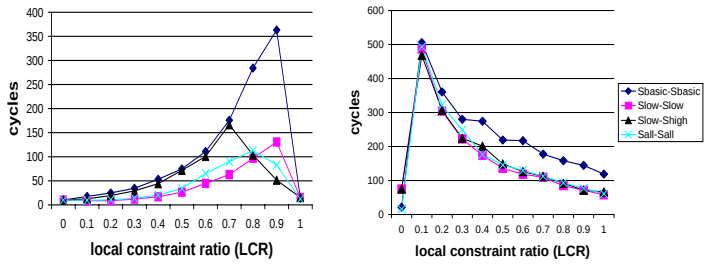
Figure 6 shows the *constraint checks* of the selected strategies on both configurations described above. These graphs show the interesting result that maximal cooperativeness towards neighboring agents is not the best strategy. Though S_{all}-S_{all} performed better than S_{basic}-S_{basic} in the grid and at higher LCR in the chain, it was worse than other less cooperative strategies. More specifically, S_{low}-S_{low}, one of the lower level cooperative strategies from Figure 3, showed the best performance in the chain and the grid except for the LCR of 0.0 and 0.9. (S_{all}-S_{all} and S_{low}-S_{high} were better than S_{low}-S_{low} at 0.0 in the chain and at 0.9 in the grid, respectively.) While S_{all}-S_{all} did not win in terms of *constraint checks*, one possible explanation is its overhead in value selection - thus, it may still win in the number of *cycles*. However, the results in Figure 7 eliminate such a possibility because *cycles* show that S_{all}-S_{all} was not the winner: even in terms of negotiation *cycles*, it was

[3]The results shown in the graphs below are for top 10% harder problems since we're more interested in difficult problem instances. At each ratio, the problems were sorted with the number of negotiation cycles with S_{basic}-S_{basic}. The average over all problems showed a similar pattern.



(a) Grid (b) Chain

Figure 6: Comparing negotiation strategies: constraint checks



(a) Grid (b) Chain

Figure 7: Comparing negotiation strategies: negotiation cycle

equivalent to or worse than other less cooperative strategies.

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 strategy S_{low}-S_{low} or S_{low}-S_{high} showed 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 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. For instance, between S_{basic}-S_{basic} and the best cooperative strategy in the grid configuration, max average difference was 4-fold in *constraint checks* and 7-fold in *cycles*. For some individual cases, there was more than 30-fold speedup in *constraint checks* and *cycles*.

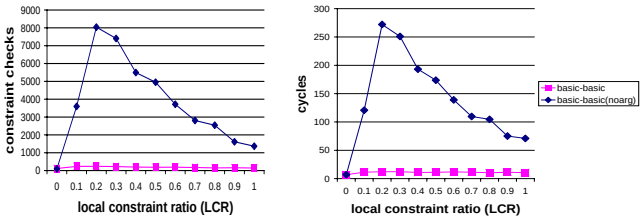
Here, to check the statistical significance of the performance difference, two-tailed t-test was done with the following two null hypotheses. For the null hypotheses, let $\mu_{\alpha-\beta}$ be an average *constraint checks* (or *cycles*) of a strategy S_α-S_β.

1. $\mu_{basic-basic} = \mu_{low-low}$
2. $\mu_{all-all} = \mu_{low-low}$

The t-test was done at each LCR in the chain and the grid: at the LCR of 0.9 in the grid, $\mu_{low-low}$ was replaced with $\mu_{low-high}$. Both null hypotheses above are rejected, i.e., the differences are significant, with p-value < 0.01 for all values of LCR (one exception is in the chain, p-value at LCR of 0.1 is less than 0.03).

5.2 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. With argumentation, agents avoid making proposals which cannot be accepted by others, which may speed up convergence. However, because of computational overhead in constraint propagation, there could be a possibility for argumentation



(a) (b)

Figure 8: Argumentation versus No-argumentation.

to be counterproductive. To answer the question, two different versions of S_{basic}-S_{basic} were compared. The first version was the S_{basic}-S_{basic} described in Section 4. The second version was same with S_{basic}-S_{basic} except that it didn't use any argumentation (constraint propagation). Let this second strategy be S_{basic}-S_{basic}(*noarg*). In S_{basic}-S_{basic}, each agent received arguments (local constraints) from neighbors and used the propagated constraints for reducing its local problem space. However, the agent had an overhead of checking extra constraints. In S_{basic}-S_{basic}(*noarg*), agents did not receive arguments from neighbors, and thus did not have to propagate constraints. Figure 8 shows the experimental results for the chain configuration with 16 agents[4]. Argumentation helped S_{basic}-S_{basic} to reduce the total negotiation effort as measured by *constraint checks* (Figure 8.a) and *cycles* (Figure 8.b).

One interesting point in Figure 8 is that the benefit of argumentation varied non-monotonically as the LCR of the agents changed. In particular, in terms of both *cycle* and *constraint checks*, the convergence speed difference between S_{basic}-S_{basic} and S_{basic}-S_{basic}(*noarg*) was the lowest when there were too few or too many locally constrained agents (e.g., at 0.0, 0.1, and 1.0). Understandably, when there were too few locally constrained agents, there were few arguments to communicate. So, argumentation was not very helpful. What was surprising was that, as the proportion of locally constrained agents 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.

5.3 Real-world Applications: Initial Results

A key benefit of our new computational model of argumentation is that it can be efficiently implemented in real-world applications. Indeed, we have successfully applied the DCSP-based model of argumentation in the distributed sensor network (using the formalization introduced in Section 3.2). Because the real hardware (Figure 1.a) was not available in our lab, this implementation has been extensively tested in a distributed sensor node simulator (Figure 1.b) that mirrors the hardware. A key evaluation criteria for this implementation is how accurately it is able to track targets, e.g., if agents do not switch on overlapping sectors at the right time, the target tracking has poor accuracy. The accuracy is measured in terms of the RMS (root mean square) error in distance between the real position of a target and the target's position as estimated by sensor agents. Although domain experts termed the RMS error of up to 3 units as acceptable, our initial results showed that in test configurations, our RMS error was less than 1 unit.

In addition to the applicability of DCSP-based model in this domain, lessons from our NVA strategy investigation are directly applicable, particularly as we scale up the number of agents. Our initial results showed that combining scale-up to 6 or 8 nodes with inappropriate NVA strategies leads the tracking errors above 3 units.

[4]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.

However, in some cases, a cooperative response enables targets to be tracked with RMS error less than 3 units. These results illustrate the utility of the DCSP model and the NVA strategy investigation.

6. RELATED WORK

While this paper builds on several previous efforts in argumentation[6] and distributed constraint satisfaction[15], it is a unique effort in synthesizing these two areas. Argumentation has been rigorously investigated using different logics including specially designed logics of argumentation[6][11]. 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.

In other related work, some computational models for negotiation strategies have been offered, e.g., [9]. However, these efforts focus on non-collaborative strategies, do not focus on investigating argumentation, and do not focus on scale-up.

Our work has built on the rich foundations of the existing DCSP work[1, 14, 15]. 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[7], and negotiation strategies as value ordering heuristics[3]. While these enhancements came about in service of studying NVA, they appear to be useful as possible enhancements to DCSP algorithms.

Finally, research on distributed resource allocation is also related. 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 instances, [8] extends dispatch scheduling to improve resource allocation; and [2] on distributed scheduling airport-ground scheduling service. While these systems do not use argumentation, hopefully our investigations will begin to shed light on the utility of argumentation in these domains.

7. CONCLUSION

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.

Instead, to address these issues, we provided a novel computational model of argumentation in terms of constraint propagation in 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 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 indeed lead to significant improvement in convergence of conflict resolution. Our experiments with NVA strategies illustrate that choosing the right strategy can lead to very significant improvement in 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.

8. ACKNOWLEDGEMENTS

This research is funded by DARPA ITO award number F30602-99-2-0507. We thank Pragnesh Modi and Wei-Min Shen for discussion related to formalization and Makoto Yokoo and Katsutoshi Hirayama for discussion related to DCSP.

9. REFERENCES

[1] A. Armstrong and E. H. Durfee, Dynamic prioritization of complex agents in distributed constraint satisfaction problems, *Proc. of IJCAI*, 1997.

[2] Mike Chia, Daniel Neiman, and Victor Lesser, Poaching and distraction in asynchronous agent activities, *Proc. of the Intl. Conf. on Multi-Agent Systems*, 1998.

[3] Daniel Frost and Rina Dechter, Look-ahead value ordering for constraint satisfaction problems, *Proc. of IJCAI*, 1995.

[4] B. Grosz, Collaborating systems, *AI magazine*, 17(2), 1996.

[5] N. R. Jennings et al., On argumentation-based negotiation, *Proc. of the Intl. Workshop on Multi-Agent Systems*, 1998.

[6] S. Kraus, K. Sycara, and A. Evenchik, A., Reaching agreements through argumentation: a logical model and implementation, *Artificial Intelligence*, 104:1-70, 1998.

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

[8] Jyi-Shane Liu and Katia P. Sycara, Multiagent coordination in tightly coupled task scheduling, *Proc. of ICMAS*, 1996.

[9] N. Matos, C. Sierra, N. R. Jennings, Determining successful negotiation strategies: an evolutionary approach, *Proc. of the Intl. Conf. on Multi-Agent Systems*, 1998.

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

[11] H. Sawamura, Y. Umeda, and R. K. Meyer, Computational dialectics for argument-based agent systems, *Proc. of the Intl. Conf. on Multi-Agent Systems*, 2000.

[12] M. Tambe, Towards flexible teamwork, *Journal of Artificial Intelligence Research (JAIR)*, 7:83-124, 1997.

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

[14] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, The distributed constraint satisfaction problem: formalization and algorithms, *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673-685, 1998.

[15] M. Yokoo and K. Hirayama, Distributed constraint satisfaction algorithm for complex local problems, *Proc. of the Intl. Conf. on Multi-Agent Systems*, 1998.