Unifying Distributed Constraint Algorithms in a BDI Negotiation Framework

Bao Chau Le Dinh and Kiam Tian Seow
School of Computer Engineering
Nanyang Technological University
Republic of Singapore
{ledi0002,asktseow}@ntu.edu.sg

ABSTRACT

This paper presents a novel, unified distributed constraint satisfaction framework based on automated negotiation. The Distributed Constraint Satisfaction Problem (DCSP) is one that entails several agents to search for an agreement, which is a consistent combination of actions that satisfies their mutual constraints in a shared environment. By anchoring the DCSP search on automated negotiation, we show that several well-known DCSP algorithms are actually mechanisms that reach agreements through a common Belief-Desire-Intention (BDI) protocol, but using different strategies. A major motivation for this BDI framework is that it not only provides a conceptually clearer understanding of existing DCSP algorithms from an agent model perspective, but also opens up the opportunities to extend and develop new strategies for DCSP. To this end, a new strategy called Unsolicited Mutual Advice (UMA) is proposed. Performance evaluation shows that the UMA strategy can outperform some existing mechanisms in terms of computational cycles.

Categories and Subject Descriptors
1.2.11 [Distributed Artificial Intelligence]: Intelligent Agents, Multiagent Systems

General Terms
Algorithms, Design, Experimentation

Keywords
Distributed Constraint Satisfaction Problem, Belief-Desire-Intention model, Agent Negotiation

1. INTRODUCTION

At the core of many emerging distributed applications is the distributed constraint satisfaction problem (DCSP) - one which involves finding a consistent combination of actions (abstracted as domain values) to satisfy the constraints among multiple agents in a shared environment. Important application examples include distributed resource allocation [1] and distributed scheduling [2]. Many important algorithms, such as distributed breakout (DBO) [3], asynchronous backtracking (ABT) [4], asynchronous partial overlay (APO) [5] and asynchronous weak-commitment (AWC) [4], have been developed to address the DCSP and provide the agent solution basis for its applications. Broadly speaking, these algorithms are based on two different approaches, either extending from classical backtracking algorithms [6] or introducing mediation among the agents.

While there has been no lack of efforts in this promising research field, especially in dealing with outstanding issues such as resource restrictions (e.g., limits on time and communication) [7] and privacy requirements [8], there is unfortunately no conceptually clear treatment to prise open the model-theoretic workings of the various agent algorithms that have been developed. As a result, for instance, a deeper intellectual understanding on why one algorithm is better than the other, beyond computational issues, is not possible.

In this paper, we present a novel, unified distributed constraint satisfaction framework based on automated negotiation [9]. Negotiation is viewed as a process of several agents searching for a solution called an agreement. The search can be realized via a negotiation mechanism (or algorithm) by which the agents follow a high level protocol prescribing the rules of interactions, using a set of strategies devised to select their own preferences at each negotiation step.

Anchoring the DCSP search on automated negotiation, we show in this paper that several well-known DCSP algorithms [3] are actually mechanisms that share the same Belief-Desire-Intention (BDI) interaction protocol to reach agreements, but use different action or value selection strategies. The proposed framework provides not only a clearer understanding of existing DCSP algorithms from a unified BDI agent perspective, but also opens up the opportunities to extend and develop new strategies for DCSP. To this end, a new strategy called Unsolicited Mutual Advice (UMA) is proposed. Our performance evaluation shows that UMA can outperform ABT and AWC in terms of the average number of computational cycles for both the sparse and critical coloring problems [6].

The rest of this paper is organized as follows. In Section 2, we provide a formal overview of DCSP. Section 3 presents a BDI negotiation model by which a DCSP agent reasons. Section 4 presents the existing algorithms ABT, AWC and DBO as different strategies formalized on a common protocol. A new strategy called Unsolicited Mutual Advice is proposed in Section 5; our empirical results and discussion attempt to highlight the merits of the new strategy over existing ones. Section 6 concludes the paper and points to some future work.

2. DCSP: PROBLEM FORMALIZATION

The DCSP [4] considers the following environment.

• There are $n$ agents with $k$ variables $x_0, x_1, \ldots, x_{k-1}$, $n \leq k$, which have values in domains $D_1, D_2, \ldots, D_k$, respectively. We define a partial function $B$ over the product-range $\{0, 1, \ldots, (n-1)\} \times \{0, 1, \ldots, (k-1)\}$ such that, that...
variable \( x_j \) belongs to agent \( i \) is denoted by \( B(i,j)! \).

- There are \( m \) constraints \( c_0, c_1, \ldots, c_{m-1} \) to be conjunctively satisfied. In a similar fashion as defined for \( B(i,j) \), we use \( E(l,j)! \), \( 0 \leq l < m, 0 \leq j < k \), to denote that \( x_j \) is relevant to the constraint \( c_l \).

The DCSP may be formally stated as follows.

**Problem Statement:** \( \forall i,j \ (0 \leq i < m) \) \( c_l \) find the assignment \( x_j = d_j \in D_j \) such that \( \forall l \ (0 \leq i < m) \) \( E(l,j)! \), \( c_l \) is satisfied.

A constraint may consist of different variables belonging to different agents. An agent cannot change or modify the assignment values of other agents’ variables. Therefore, in cooperatively searching for a DCSP solution, the agents would need to communicate with one another, and adjust and re-adjust their own variable assignments in the process.

### 2.1 DCSP Agent Model

In general, all DCSP agents must cooperatively interact, and essentially perform the assignment and reassignment of domain values to variables to resolve all constraint violations. If the agents succeed in their resolution, a solution is found.

In order to engage in cooperative behavior, a DCSP agent needs five fundamental parameters, namely, (i) a variable \([4]\), or a variable set \([10]\), (ii) domains, (iii) priority, (iv) a neighbor list and (v) a constraint list.

Each variable assumes a range of values called a domain. A domain value, which usually abstracts an action, is a possible option that an agent may take. Each agent has an assigned priority. These priority values help decide the order in which they revise or modify their variable assignments. An agent’s priority may be fixed (static) or changing (dynamic) when searching for a solution. If an agent has more than one variable, each variable can be assigned a different priority, to help determine which variable assignment the agent should modify first.

A DCSP agent which shares the same constraint with another agent is called the latter’s neighbor. Each agent needs to refer to its list of neighbors during the search process. This list may also be kept unchanged or updated accordingly in runtime. Similarly, each agent maintains a constraint list. The agent needs to ensure that there is no violation of the constraints in this list. Constraints can be added or removed from an agent’s constraint list in runtime.

As with an agent, a constraint can also be associated with a priority value. Constraints with a high priority are said to be more important than constraints with a lower priority. To distinguish it from the priority of an agent, the priority of a constraint is called its weight.

### 3. THE BDI NEGOTIATION MODEL

The BDI model originates with the work of M. Bratman [11]. According to [12, Ch.1], the BDI architecture is based on a philosophical model of human practical reasoning, and draws out the process of reasoning by which an agent decides which actions to perform at consecutive moments when pursuing certain goals.

Grounding the scope to the DCSP framework, the common goal of all agents is finding a combination of domain values to satisfy a set of predefined constraints. In automated negotiation [9], such a solution is called an agreement among the agents. Within this scope, we found that we were able to unearth the generic behavior of a DCSP agent and formulate it in a negotiation protocol, prescribed using the powerful concepts of BDI. Thus, our proposed negotiation model can be said to combine the BDI concepts with automated negotiation in a multiagent framework, allowing us to conceptually separate DCSP mechanisms into a common BDI interaction protocol and the adopted strategies.

#### 3.1 The generic protocol

Figure 1 shows the basic reasoning steps in an arbitrary round of negotiation that constitute the new protocol. The solid line indicates the common component or transition which always exists regardless of the strategy used. The dotted line indicates the component or transition which may or may not appear depending on the adopted strategy.

Two types of messages are exchanged through this protocol, namely, the info message and the negotiation message.

An info message is a message sent by another agent. The message will contain the current selected values and priorities of the variables of that sending agent. The main purpose of this message is to update the agent about the current environment. The agent also updates its constraint list using some criteria of the adopted strategy.

An info message is sent out at the end of one reasoning round (also called a negotiation cycle), and received at the beginning of next round.

A negotiation message is a message which may be sent within a round. This message is for mediation purposes. The agent may put different contents into this type of message as long as it is agreed among the group. The format of the negotiation message and when it is to be sent out are subject to the strategy. A negotiation message can be sent out at the end of one reasoning step and received at the beginning of the next step.

Mediation is a step of the protocol that depends on whether the agent’s interaction with others is synchronous or asynchronous. In synchronous mechanism, mediation is required in every negotiation round. In an asynchronous one, mediation is needed only in a negotiation round when the agent receives a negotiation message. A more in-depth view of this mediation step is provided later in this section.

The BDI protocol prescribes the skeletal structure for DCSP negotiation. We will show in Section 4 that several well-known DCSP mechanisms all inherit this generic model.

The details of the six main reasoning steps for the protocol (see Figure 1) are described as follows for a DCSP agent. For a conceptually clearer description, we assume that there is only one variable per agent.

- **Percept.** In this step, the agent receives info messages from its neighbors in the environment, and using its Percept function, returns an image \( P \). This image contains the current values assigned to the variables of all agents in their neighbor list. The image \( P \) will drive the agent’s actions in subsequent steps. The agent also updates its constraint list \( C \) using some criteria of the adopted strategy.

- **Belief.** Using the image \( P \) and constraint list \( C \), the agent will check if there is any violated constraint. If there is no violation, the agent will believe it is choosing a correct option and therefore will take no action. The agent will do nothing if it is in a local stable state - a snapshot of the variables assignments of the agent and all its neighbors by which they satisfy their shared constraints. When all agents are in their local stable states, the whole environment is said to be in a global stable state and an agree-
ment is found. In case the agent finds its value in conflict with some of its neighbours, i.e., the combination of values assigned to the variables leads to a constraint violation, the agent will first try to reassign its own variable using a specific strategy. If it finds a suitable option which meets some criteria of the adopted strategy, the agent will believe it should change to the new option. However it does not always happen that an agent can successfully find such an option. If no option can be found, the agent will believe it has no option, and therefore will request its neighbours to reconsider their variable assignments.

To summarize, there are three types of beliefs that a DCSP agent can form: (i) it can change its variable assignment to improve the current situation, (ii) it cannot change its variable assignment and some constraints violations cannot be resolved and (iii) it need not change its variable assignment as all the constraints are satisfied.

Once the beliefs are formed, the agent will determine its desires, which are the options that attempt to resolve the current constraint violations.

- **Desire.** If the agent takes Belief (i), it will generate a list of its own suitable domain values as its desire set. If the agent takes Belief (ii), it cannot ascertain its desire set, but will generate a sublist of agents from its neighbor list, whom it will ask to reconsider their variable assignments. How this sublist is created depends on the strategy devised for the agent. In this situation, the agent will use a virtual desire set that it determines based on its adopted strategy. If the agent takes Belief (iii), it will have no desire to revise its domain value, and hence no intention.

- **Intention.** The agent will select a value from its desire set as its intention. An intention is the best desired option that the agent assigns to its variable. The criteria for selecting a desire as the agent’s intention depend on the strategy used. Once the intention is formed, the agent may either proceed to the execution step, or undergo mediation. Again, the decision to do so is determined by some criteria of the adopted strategy.

- **Mediation.** This is an important function of the agent. Since, if the agent executes its intention without performing intention mediation with its neighbours, the constraint violation between the agents may not be resolved. Take for example, suppose two agents have variables, $x_1$ and $x_2$, associated with the same domain $\{1, 2\}$, and their shared constraint is $(x_1 + x_2 = 3)$. Then if both the variables are initialized with value 1, they will both concurrently switch between the values 2 and 1 in the absence of mediation between them.

There are two types of mediation: local mediation and group mediation. In the former, the agents exchange their intentions. When an agent receives another’s intention which conflicts with its own, the agent must mediate between the intentions, by either changing its own intention or informing the other agent to change its intention. In the latter, there is an agent which acts as a group mediator. This mediator will collect the intentions from the group - a union of the agent and its neighbours - and determine which intention is to be executed. The result of this mediation is passed back to the agents in the group. Following mediation, the agent may proceed to the next reasoning step to execute its intention or begin a new negotiation round.

- **Execution.** This is the last step of a negotiation round. The agent will execute by updating its variable assignment if the intention obtained at this step is its own. Following execution, the agent will inform its neighbours about its new variable assignment and updated priority. To do so, the agent will send out an info message.

### 3.2 The strategy

A strategy plays an important role in the negotiation process. Within the protocol, it will often determine the efficiency of the search process in terms of computational cycles and message communication costs.

The design space when devising a strategy is influenced by the following dimensions: (i) asynchronous or synchronous, (ii) dynamic or static priority, (iii) dynamic or static constraint weight, (iv) number of negotiation messages to be communicated, (v) the negotiation message format and (vi) the completeness property. In other words, these dimensions provide technical considerations for a strategy design.

### 4. DCSP ALGORITHMS: BDI PROTOCOL + STRATEGIES

In this section, we apply the proposed BDI negotiation model presented in Section 3 to expose the BDI protocol and the different strategies used for three well-known algorithms, ABT, AW and DBO. All these algorithms assume that there is only one variable per agent. Under our framework, we call the strategies applied the ABT, AW and DBO strategies, respectively.

To describe each strategy formally, the following mathematical notations are used:

- $n$ is the number of agents, $m$ is the number of constraints;
- $x_i$ denotes the variable held by agent $i$, $(0 \leq i < n)$;
- $D_i$ denotes the domain of variable $x_i$; $F_j$ denotes the neighbor list of agent $i$; $C_i$ denotes its constraint list;
- $p_j$ denotes the priority of agent $j$; and $P_i = \{(x_j = v_j, p_j = k) | \text{agent } j \in F_i, v_j \in D_j, \text{the current value assigned to } x_j \text{ and the priority value } k \text{ is a positive integer }\}$ is the perception of agent $i$;
- $w_j$ denotes the weight of constraint $i$, $(0 \leq i < m)$;
- $S_i(v)$ is the total weight of the violated constraints in $C_i$ when its variable has the value $v \in D_i$.

#### 4.1 Asynchronous Backtracking

Figure 2 presents the BDI negotiation model incorporating the Asynchronous Backtracking (ABT) strategy. As mentioned in Section 3, for an asynchronous mechanism that ABT is, the mediation step is needed only in a negotiation round when an agent receives a negotiation message.

For agent $i$, beginning initially with $(w_i = 1, \text{ for } 0 \leq i < m)$ and $F_i$ contains all the agents who share the constraints with agent $i$, its BDI-driven ABT strategy is described as follows.

**Step 1 - Percept:** Update $P_i$ upon receiving the info messages from the neighbors in $F_i$. Update $C_i$ to be the list of
constraints which only consists of agents in $F_i$ that have equal or higher priority than this agent.

**Step 2 - Belief:** The belief function $GB$ ($P_i(a)$) will return a value $b_i \in \{0,1,2\}$, decided as follows:

- $b_i = 0$ when agent $i$ can find an optimal option, i.e., if $(S_i(a) \neq 0$ or $v_i$ is in bad values list) and $a$ is in list of domain values called bad values list. Initially this list is empty and it will be cleared when a neighbor of higher priority changes its variable assignment.
- $b_i = 1$ when it cannot find an optimal option, i.e., if $(\forall a \in D_i)(S_i(a) = 0)$ or $a$ is in bad values list.
- $b_i = 2$ when its current variable assignment is an optimal option, i.e., if $S_i(v_i) = 0$ or $v_i$ is in bad value list.

**Step 3 - Desire:** The desire function $GD$ ($b_i$) will return a desire set denoted by $DS$, decided as follows:

- If $b_i = 0$, then $DS = \{a | (a \neq v_i), S_i(a) = 0\}$ and $a$ is not in the bad value list.
- If $b_i = 1$, then $DS = \emptyset$, the agent also finds agent $k$ which is determined by $\{k | p_i = \min(p_j)\}$ with agent $j \in F_i$ and $p_k > p_i$.
- If $b_i = 2$, then $DS = \emptyset$.

**Step 4 - Intention:** The intention function $GI$ ($DS$) will return an intention, decided as follows:

- If $DS \neq \emptyset$, then select an arbitrary value (say, $a' \in DS$) as the intention.
- If $DS = \emptyset$, then assign nil as the intention (to denote its lack thereof).

**Step 5 - Execution:**

- If agent $i$ has a domain value as its intention, the agent will update its variable assignment with this value.
- If $b_i = 1$, agent $i$ will send a negotiation message to agent $k$, then remove $k$ from $F_i$ and begin its next negotiation round. The negotiation message will contain the list of variable assignments of those agents in its neighbor list $F_i$ that have a higher priority than agent $i$ in the current image $P_i$.

**Mediation:** When agent $i$ receives a negotiation message, several sub-steps are carried out, as follows:

- If the list of agents associated with the negotiation message contains agents which are not in $F_i$, it will add these agents to $F_i$, and request these agents to add itself to their neighbor lists. The request is considered as a type of negotiation message.
- Agent $i$ will first check if the sender agent is updated with its current value $v_i$. The agent will add $v_i$ to its bad values list if it is so, or otherwise send its current value to the sender agent.

Following this step, agent $i$ proceeds to the next negotiation round.

### 4.2 Asynchronous Weak Commitment Search

Figure 3 presents the BDI negotiation model incorporating the Asynchronous Weak Commitment (AWC) strategy. The model is similar to that of incorporating the ABT strategy (see Figure 2). This is not surprising; AWC and ABT are found to be strategically similar, differing only in the details of some reasoning steps. The distinguishing point of AWC is that when the agent cannot find a suitable variable assignment, it will change its priority to the highest among its group members ($\{|i| \cup F_i\}$).

For agent $i$, beginning initially with $(v_i = 1, (0 \leq i \leq m); p_i = i, (0 \leq i < n))$ and $F_i$ contains all the agents who share the constraints with agent $i$, its BDI-driven AWC strategy is described as follows:

**Step 1 - Percept:** This step is identical to the Percept step of ABT.

**Step 2 - Belief:** The belief function $GB$ ($P_i(a)$) will return a value $b_i \in \{0,1,2\}$, decided as follows:

- $b_i = 0$ when agent $i$ can find an optimal option, i.e., if $(S_i(a) \neq 0) or (v_i$ is in bad values list) and $a$ is in list of domain values called bad values list. Initially this list is empty and it will be cleared when a neighbor of higher priority changes its variable assignment.
- $b_i = 1$ when it cannot find an optimal option, i.e., if $(\forall a \in D_i)(S_i(a) = 0)$ or $a$ is in bad values list.
- $b_i = 2$ when its current variable assignment is an optimal option, i.e., if $S_i(v_i) = 0$ or $v_i$ is in bad value list.

**Step 3 - Desire:** The desire function $GD$ ($b_i$) will return a desire set denoted by $DS$, decided as follows:

- If $b_i = 0$, then $DS = \{a | (a \neq v_i), S_i(a) = 0\}$ and $a$ is not in the bad value list.
- If $b_i = 1$, then $DS = \emptyset$, the agent also finds agent $k$ which is determined by $\{k | p_i = \min(p_j)\}$ with agent $j \in F_i$ and $p_k > p_i$.
- If $b_i = 2$, then $DS = \emptyset$.

**Step 4 - Intention:** The intention function $GI$ ($DS$) will return an intention, decided as follows:

- If $DS \neq \emptyset$, then select an arbitrary value (say, $a' \in DS$) as the intention.
- If $DS = \emptyset$, then assign nil as the intention (to denote its lack thereof).

**Step 5 - Execution:**

- If agent $i$ has a domain value as its intention, the agent will update its variable assignment with this value.
- If $b_i = 1$, agent $i$ will send a negotiation message to agent $k$, then remove $k$ from $F_i$ and begin its next negotiation round. This step is identical to the Percept step of ABT. However, for this strategy, the negotiation message will contain the list of variable assignments of those neighbors in $F_i$ who have higher priority than this agent, similar to that of incorporating the ABT strategy (see Figure 2).

**Asynchronous Weak Commitment (AWC) strategy**

- $b_i = 0$ when the agent can find an optimal option i.e., if $(S_i(v_i) \neq 0)$ or the assignment $x_i = v_i$, and the current variables assignments of the neighbors in $F_i$ who have higher priority form a nogood [4] stored in a list called nogood list and $a \in D_i, S_i(a) = 0$ (initially the list is empty).
- $b_i = 1$ when the agent cannot find any optimal option i.e., if $\forall a \in D_i, S_i(a) \neq 0$.
- $b_i = 2$ when the current assignment is an optimal option i.e., if $S_i(v_i) = 0$ and the current state is not a nogood in nogood list.

**Step 3 - Desire:** The desire function $GD$ ($b_i$) will return a desire set $DS$, decided as follows:

- If $b_i = 0$, then $DS = \{a | (a \neq v_i), (S_i(a) = 0)\}$ and the number of constraint violations with lower priority agents is minimized.
- If $b_i = 1$, then $DS = \{a | a \in D_i\}$ and the number of violations of all relevant constraints is minimized.
- If $b_i = 2$, then $DS = \emptyset$.

Following, if $b_i = 1$, agent $i$ will find a list $K_i$ of higher priority neighbors, defined by $K_i = \{k | a \in F_i\}$ and $p_k > p_i$.

**Step 4 - Intention:** This step is similar to the Intention step of ABT. However, for this strategy, the negotiation message will contain the variable assignments (of the current image $P_i$) for all the agents in $K_i$. This list of assignment is considered as a nogood. If the same negotiation message had been sent out before, agent $i$ will have nil intention. Otherwise, the agent will send the message and save the nogood in the nogood list.

**Step 5 - Execution:**

- If agent $i$ has a domain value as its intention, the agent will update its variable assignment with this value.
- If $b_i = 1$, it will send the negotiation message to its neighbors in $K_i$, and set $p_i = \max\{p_j\} + 1$, with agent $j \in F_i$.

**Mediation:** This step is identical to the Mediation step of ABT, except that agent $i$ will now add the nogood contained in the negotiation message received to its own nogood list.

### 4.3 Distributed Breakout

Figure 4 presents the BDI negotiation model incorporating the Distributed Breakout (DBO) strategy. Essentially, by this synchronous strategy, each agent will search iteratively for improvement by reducing the total weight of the violated constraints. The iteration will continue until no agent can improve further, at which time if some constraints remain violated, the weights of

![Figure 3: BDI protocol with Asynchronous Weak Commitment strategy](image-url)
Figure 4: BDI protocol with Distributed Breakout strategy

these constraints will be increased by 1 to help ‘breakout’ from a local minimum.
For agent $i$, beginning initially with $(a_l = 1, \ (0 \leq l < m))$, $p_i = i, \ (0 \leq i < n)$ and $F_i$ contains all the agents who share the constraints with agent $i$. its BDI-driven DBO strategy is described as follows.

**Step 1 - Percept:** Update $P_i$ upon receiving the info messages from the neighbors (in $F_i$). Update $C_i$ to be the list of its relevant constraints.

**Step 2 - Belief:** The belief function $GB(P_i,C_i)$ will return a value $b_i \in \{0,1,2\}$, decided as follows:
- $b_i = 0$ when agent $i$ can find an option to reduce the number of violations of the constraints in $C_i$, i.e., if $\exists a \in D_i, S_i(a) < S_i(v_i)$;
- $b_i = 1$ when it cannot find any option to improve situation, i.e., if $\forall a \in D_i, a \neq v_i, S_i(a) \geq S_i(v_i)$;
- $b_i = 2$ when its current assignment is an optimal option, i.e., if $S_i(v_i) = 0$.

**Step 3 - Desire:** The desire function $GD(P_i,C_i)$ will return a desire set $DS$, decided as follows:
- If $b_i = 0$, then $DS = \{a \mid a \neq v_i, S_i(a) < S_i(v_i) \text{ and } (S_i(v_i) - S_i(a)) \text{ is maximized}\}$. $(\max((S_i(v_i) - S_i(a)))$ will be referenced by $h_{i}^{max}$ in subsequent steps, and it defines the maximal reduction in constraint violations).
- Otherwise, $DS = \emptyset$.

**Step 4 - Intention:** The intention function $GI(DS)$ will return an intention, decided as follows:
- If $DS \neq \emptyset$, then select an arbitrary value (say, $v_i^{j}$) from $DS$ as the intention.
- If $DS = \emptyset$, then assign $nil$ as the intention.

Following, agent $i$ will send its intention to all its neighbors. In return, it will receive intentions from these agents before proceeding to Mediation step.

**Mediation:** Agent $i$ receives all the intentions from its neighbors. If it finds that the intention received from a neighbor agent $j$ is associated with $h_{i}^{max} > h_{j}^{max}$, the agent will automatically cancel its current intention.

**Step 5 - Execution:**
- If agent $i$ did not cancel its intention, it will update its variable assignment with the intended value.

Figure 5 presents the BDI negotiation model incorporating the Unsolicited Mutual Advice (UMA) strategy.

Unlike when using the strategies of the previous section, a DCSP agent using UMA will not only send out a negotiation message when concluding its Intention step, but also when concluding its Desire step. The negotiation message that it sends out to conclude the Desire step constitutes an unsolicited advice for all its neighbors. In turn, the agent will wait to receive unsolicited advice from all its neighbors, before proceeding on to determine its intention.

For agent $i$, beginning initially with $(a_l = 1, \ (0 \leq l < m))$, $p_i = i, \ (0 \leq i < n)$ and $F_i$ contains all the agents who share the constraints with agent $i$. its BDI-driven UMA strategy is described as follows.

**Step 1 - Percept:** Update $P_i$ upon receiving the info messages from the neighbors (in $F_i$). Update $C_i$ to be the list of constraints relevant to agent $i$.

**Step 2 - Belief:** The belief function $GB(P_i,C_i)$ will return a value $b_i \in \{0,1,2\}$, decided as follows:
- $b_i = 0$ when agent $i$ can find an option to reduce the number of violations of the constraints in $C_i$, i.e., if $\exists a \in D_i, S_i(a) < S_i(v_i)$ and the assignment $x_i = a$ and the current variable assignments of its neighbors do not form a local state stored in a list called bad states list (initially this list is empty).
- $b_i = 1$ when it cannot find a value $a$ such as $a \in D_i, S_i(a) < S_i(v_i)$, and the assignment $x_i = a$ and the current variable assignments of its neighbors do not form a local state stored in the bad states list.
- $b_i = 2$ when its current assignment is an optimal option, i.e., if $S_i(v_i) = 0$.

**Step 3 - Desire:** The desire function $GD(P_i,C_i)$ will return a desire set $DS$, decided as follows:
- If $b_i = 0$, then $DS = \{a \mid a \neq v_i, S_i(a) < S_i(v_i) \text{ and } (S_i(v_i) - S_i(a)) \text{ is maximized}\}$. $(\max((S_i(v_i) - S_i(a)))$ will be referenced by $h_{i}^{max}$ in subsequent steps, and it defines

5. THE UMA STRATEGY

If all intentions received and its own one are nil intention, the agent will increase the weight of each currently violated constraint by 1.
the maximal reduction in constraint violations. It is also referred to as an improvement).

- If \( b_i = 1 \), then \( DS = \{ \{ a \mid a \neq c_i, S_i(a) \text{ is minimized} \} \) and the assignment \( x_i = a \) and the current variable assignments of agent \( i \)'s neighbors do not form a state in the bad states list. In this case, DS is called a set of reluctant desires.

- If \( b_i = 2 \), then \( DS = \emptyset \).

Following, if \( b_i = 0 \), agent \( i \) will send a negotiation message containing \( h_i^{max} \) to all its neighbors. This message is called a voluntary advice. If \( b_i = 1 \), agent \( i \) will send a negotiation message called change advice to the neighbors in \( F_i \) who share the violated constraints with agent \( i \).

Agent \( i \) receives advices from all its neighbors and stores them in a list called \( A \), before proceeding to the next step.

**Step 4 - Intention:** The intention function \( GI (DS, A) \) will return an intention, decided as follows:

- If there is a voluntary advice from an agent \( j \) which is associated with \( h_j^{max} > h_i^{max} \), assign nil as the intention.
- If \( DS \neq \emptyset \), DS is a set of voluntary desires and \( h_i^{max} \) is the biggest improvement among those associated with the voluntary advices received, select an arbitrary value \( \{ a \mid v_i = a \wedge v_i \neq h_i^{max} \} \) from DS as the intention. This intention is called a voluntary intention.
- If \( DS \neq \emptyset \), DS is a set of reluctant desires and agent \( i \) receives some change advices, select an arbitrary value \( \{ a \mid v_i = a \wedge v_i \neq h_i^{max} \} \) from DS as the intention. This intention is called reluctant intention.
- If \( DS = \emptyset \), then assign nil as the intention.

Following, if the improvement \( h_i^{max} \) is the biggest improvement and equal to some improvements associated with the received voluntary advices, agent \( i \) will send its computed intention to all its neighbors. If agent \( i \) has a reluctant intention, it will also send this intention to all its neighbors. In both cases, agent \( i \) will attach the number of received change advices in the current negotiation round with its intention. In return, agent \( i \) will receive the intentions from its neighbors before proceeding to Mediation step.

**Mediation:** If agent \( i \) does not send out its intention before this step, i.e., the agent has either a nil intention or a voluntary intention with biggest improvement, it will proceed to next step. Otherwise, agent \( i \) will select the best intention among all the intentions received, including its own (if any). The criteria to select the best intention are listed, applied in descending order of importance as follows:

- A voluntary intention is preferred over a reluctant intention.
- A voluntary intention (if any) with biggest improvement is selected.
- If there is no voluntary intention, the reluctant intention with the lowest number of constraint violations is selected.
- The intention from an agent who has received a higher number of change advices in the current negotiation round is selected.
- Intention from an agent with highest priority is selected.

If the selected intention is not agent \( i \)'s intention, it will cancel its intention.

**Step 5 - Execution:** If agent \( i \) does not cancel its intention, it will update its variable assignment with the intended value.

**Termination Condition:** Since each agent does not have full information about the global state, it may not know when it has reached a solution, i.e., when all the agents are in a global stable state. Hence an observer is needed that will keep track of the negotiation messages communicated in the environment. Following a certain period of time when there is no more message communication (and this happens when all the agents have no more intention to update their variable assignments), the observer will inform the agents in the environment that a solution has been found.

### 5.1 An Example

To illustrate how UMA works, consider a 2-color graph problem [6] as shown in Figure 6. In this example, each agent has a color variable representing a node. There are 10 color variables sharing the same domain \{Black, White\}.

The following records the outcome of each step in every negotiation round executed.

**Round 1:**

**Step 1 - Percept:** Each agent obtains the current color assignments of those nodes (agents) adjacent to it, i.e., its neighbors.

**Step 2 - Belief:** Agents which have positive improvements are agent 1 (this agent believes it should change its color to White), agent 2 (this agent believes should change its color to White), agent 7 (this agent believes it should change its color to Black) and agent 10 (this agent believes it should change its value to Black). In this negotiation round, the improvements achieved by these agents are 1. Agents which do not have any improvements are agents 4, 5 and 8. Agents 3, 6 and 9 need not change as all their relevant constraints are satisfied.

**Step 3 - Desire:** Agents 1, 2, 7 and 10 have the voluntary desire (White color for agents 1, 2 and Black color for agents 7, 10). These agents will send the voluntary advices to all their neighbors. Meanwhile, agents 4, 5 and 8 have the reluctant desires (White color for agents 4 and Black color for agents 5, 8). Agent 4 will send a change advice to agent 2 as agent 2 is sharing the violated constraint with it. Similarly, agents 5 and 8 will send change advices to agents 7 and 10 respectively. Agents 3, 6 and 9 do not have any desire to update their color assignments.

**Step 4 - Intention:** Agents 2, 7 and 10 receive the change advices from agents 4, 5 and 8 respectively. They form their voluntary intentions. Agents 4, 5 and 8 receive the voluntary advices from agents 2, 7 and 10, hence they will not have any intention. Agents 3, 6 and 9 do not have any intention. Following, the intention from the agents will be sent to all their neighbors.

**Mediation:** Agent 1 finds that the intention from agent 2 is better than its intention. This is because, although both agents have voluntary intentions with improvement of 1, agent 2 has received one change advice from agent 4 while agent 1 has not received any. Hence agent 1 cancels its intention. Agent 2 will keep its intention. Agents 7 and 10 keep their intentions since none of their neighbors has an intention.

The rest of the agents do nothing in this step as they do not have any intention.

**Step 5 - Execution:** Agent 2 changes its color to White. Agents 7 and 10 change their colors to Black.

The new state after round 1 is shown in Figure 7.

**Round 2:**

**Step 1 - Percept:** The agents obtain the current color assignments of their neighbors.

**Step 2 - Belief:** Agent 3 is the only agent who has a positive improvement which is 1. It believes it should change its...
Figure 7: The graph after round 1

color to Black. Agent 2 does not have any positive improvement. The rest of the agents need not make any change as all their relevant constraints are satisfied. They will have no desire, and hence no intention.

**Step 3 - Desire:** Agent 3 desires to change its color to Black voluntarily, hence it sends out a voluntary advice to its neighbor, i.e., agent 2. Agent 2 does not have any value for its reluctant desire set as the only option. Black color, will bring agent 2 and its neighbors to the previous state which is known to be a bad state. Since agent 2 is sharing the constraint violation with agent 3, it sends a change advice to agent 3.

**Step 4 - Intention:** Agent 3 will have a voluntary intention while agent 2 will not have any intention as it receives the voluntary advice from agent 3.

**Mediation:** Agent 3 will keep its intention as its only neighbor, agent 2, does not have any intention.

**Step 5 - Execution:** Agent 3 changes its color to Black.

The new state after round 2 is shown in Figure 8.

Round 3: In this round, every agent finds that it has no desire and hence no intention to revise its variable assignment. Following, with no more negotiation message communication in the environment, the observer will inform all the agents that a solution has been found.

Figure 8: The solution obtained

5.2 Performance Evaluation

To facilitate credible comparisons with existing strategies, we measured the execution time in terms of computational cycles as defined in [4], and built a simulator that could reproduce the published results for ABT and AWC. The definition of a computational cycle is as follows.

- In one cycle, each agent receives all the incoming messages, performs local computation and sends out a reply.
- A message which is sent at time \( t \) will be received at time \( t+1 \). The network delay is neglected.
- Each agent has its own clock. The initial clock’s value is 0. Agents attach their clock value as a time-stamp in the outgoing message and use the time-stamp in the incoming message to update their own clock’s value.

Four benchmark problems [6] were considered, namely, \( n \)-queens and node coloring for sparse, dense and critical graphs. For each problem, a finite number of test cases were generated for various problem sizes \( n \). The maximum execution time was set to 10000 cycles for critical graphs and 1000 cycles for other problems. The simulator program was terminated after this period and the algorithm was considered to fail a test case if it did not find a solution by then. In such a case, the execution time for the test was counted as 1000 cycles.

5.2.1 Evaluation with \( n \)-queens problem

The \( n \)-queens problem is a traditional problem of constraint satisfaction. 10 test cases were generated for each problem size \( n \in \{10, 50 \text{ and } 100\} \).

Figure 9 shows the execution time for different problem sizes when ABT, AWC, and UMA were run.

5.2.2 Evaluation with graph coloring problem

The graph coloring problem can be characterized by three parameters: (i) the number of colors \( k \), the number of nodes/agents \( n \) and the number of links \( m \). Based on the ratio \( m/n \), the problem can be classified into three types [3]: (i) sparse (with \( m/n = 2 \)), (ii) critical (with \( m/n = 2.7 \) or 4.7) and (iii) dense (with \( m/n = (n-1)/4 \)). For this problem, we did not include ABT in our empirical results as its failure rate was found to be very high. This poor performance of ABT was expected since the graph coloring problem is more difficult than the \( n \)-queens problem, on which ABT already did not perform well (see Figure 9).

The sparse and dense (coloring) problem types are relatively easy while the critical type is difficult to solve. In the experiments, we fix \( k = 3 \). 10 test cases were created using the method described in [13] for each value of \( n \in \{60, 90, 120\} \), for each problem type.

The simulation results for each type of problem are shown in Figures 10 - 12.

Figure 10: Comparison between AWC and UMA (sparse graph coloring)

5.3 Discussion

5.3.1 Comparison with ABT and AWC
than DBO for the following reasons:

**5.3.2 Comparison with DBO**

![Figure 11: Comparison between AWC and UMA (critical graph coloring)](image1)

![Figure 12: Comparison between AWC and UMA (dense graph coloring)](image2)

Figure 10 shows that the average performance of UMA is slightly better than AWC for the sparse problem. UMA outperforms AWC in solving the critical problem as shown in Figure 11. It was observed that the latter strategy failed in some test cases. However, as seen in Figure 12, both the strategies are very efficient when solving the dense problem, with AWC showing slightly better performance.

The performance of UMA, in the worst (time complexity) case, is similar to that of all evaluated strategies. The worst case occurs when all the possible global states of the search are reached. Since only a few agents have the right to change their variable assignments in a negotiation round, the number of redundant computational cycles and info messages is reduced. As we observe from the backtracking in ABT and AWC, the difference in the ordering of incoming messages can result in a different number of computational cycles to be executed by the agents.

5.3.2 Comparison with DBO

The computational performance of UMA is arguably better than DBO for the following reasons:

- **UMA** can guarantee that there will be a variable reassignment following every negotiation round whereas DBO cannot.
- **UMA** introduces one more communication round trip (that of sending a message and awaiting a reply) than DBO, which occurs due to the need to communicate unsolicited advice. Although this increases the communication cost per negotiation round, we observed from our simulations that the overall communication cost incurred by UMA is lower due to the significantly lower number of negotiation rounds.
- Using UMA, in the worst case, an agent will only take 2 or 3 communication round trips per negotiation round, following which the agent or its neighbor will do a variable assignment update. Using DBO, this number of round trips is uncertain as each agent might have to increase the weights of the violated constraints until an agent has a positive improvement; this could result in an infinite loop [3].

6. CONCLUSION

Applying automated negotiation to DCSP, this paper has proposed a protocol that prescribes the generic reasoning of a DCSP agent in a BDI architecture. Our work shows that several well-known DCSP algorithms, namely ABT, AWC and DBO, can be described as mechanisms sharing the same proposed protocol, and only differ in the strategies employed for the reasoning steps per negotiation round as governed by the protocol. Importantly, this means that it might furnish a unified framework for DCSP that not only provides a clearer BDI agent-theoretic view of existing DCSP approaches, but also opens up the opportunities to enhance or develop new strategies. Towards the latter, we have proposed and formulated a new strategy - the UMA strategy. Empirical results and our discussion suggest that UMA is superior to ABT, AWC and DBO in some specific aspects.

It was observed from our simulations that UMA possesses the completeness property. Future work will attempt to formally establish this property, as well as formalize other existing DSCP algorithms as BDI negotiation mechanisms, including the recent endeavor that employs a group mediator [5]. The idea of DCSP agents using different strategies in the same environment will also be investigated.

7. REFERENCES


