Approach to solving DCSP using BDI

JOSÉ MIGUEL RUBIO\textsuperscript{1}, BRODERICK CRAWFORD\textsuperscript{2}, JIMENA AHUMADA\textsuperscript{3}

Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso
Avenida Brasil 2241, Valparaíso
CHILE
\textsuperscript{1} jose.rubio.l@ucv.cl
\textsuperscript{2} broderick.crawford@ucv.cl
\textsuperscript{3} jahumada@gmail.com

Abstract: - BDI is an agent who works based on three basic attitudes: beliefs, desires and intentions. Is presented in this paper the development of a multiagent system based on BDI agents or intentional agents to solve problems of satisfaction of typical restrictions, using the algorithm ”Asynchronous Backtracking. The problem is distributed on several agents working together to find a solution. Is shown part of the system design, through development methodology oriented to AAII agents, and the application of the above by implementing two such problems: N-Queens and Sudoku, the results obtained from experiments and conclusions based on this.

Key-Words: - BDI agents, Distributed Constraint Satisfaction Problems, Multiagent Systems, Constraint Satisfaction Problems, N-Queens, Sudoku.

1 Introduction

A Constraint Satisfaction Problem (CSP) is which is able to resolve the constraints between a set of variables. A myriad of problems that can be modeled in this way, and its resolution has led to many studies which have succeeded in doing, mostly centrally, resulting in great efforts when the problem is too big and probably in high time of response. However, the development of multiagent systems has generated interest for its ability to face problems in a distributed way, and since the first studies were published in this field [8] have been developed some algorithms of resolution today.

Another thing, there is a particular type of agent, referred as BDI agent, whose initials are the acronym of Beliefs, Desires and Intentions. This type of agent works based on these three states of mind, through environment observation and generating plans.

The idea of this work is the integration of all these concepts, since there are few studies on this [1]. Using the tools proposed by the methodology of development-oriented members of the Institute of Artificial Intelligence Australian AAII [4] has been designed and implemented a system capable of solving CSPs through a technique distributed, using the mental attitudes involved in intentional agents known as BDI.

Below are the formal definitions of the three concepts recently mentioned.

1.1 CSP definition

Literature referring to CSP has formalized it in a very similar form in the majority of publications of that kind of problems, in this case we will resort to the definition published in [7], that says that “a CSP P is a triple $P=\{X, D, C\}$ Where:

- $X$ is an n-tuple of variables $X=\{x_1, x_2, ..., x_n\}$,
- $D$ is a corresponding n-tuple of domains $D=\{D_1, D_2, ..., D_n\}$ such as $x_i \in D_i$,
- $C$ is a t-tuple of constraints $C=\{C_1, C_2, ..., C_t\}$.

A constraint $C_j$ is a pair $\langle r_{j}, s_{j} \rangle$ where $r_{j,s_{j}}$ is a relation on the variables in $S_{j}=\text{scope}(C_j)$. In other words, $r_{j}$ is a subset of the Cartesian product of the domains of the variables in $S_{j}$.

As examples of representations of classical CSP can mention the problem of the n-queens and graph coloring.

1.2 DCSP definition

The formalization of a distributed CSP is the same one for a CSP, the difference is in the form in which the problem is solved. A DCSP distributes to the variables and restrictions in several agents who negotiate and collaborate to each other to solve the problem altogether.

Formally, according to [4], there exist m agents 1, 2, ..., m. Each variable belongs to one agent i (this relation is presented as $\text{belongs}(x_i, i)$). Constraints
are also distributed among agents. The fact that an agent/knows a constraint predicate \( p_k \) is represented as \( \text{known} (p_k, I) \).

The same classic problems identified as CSP can be modeled as a DCSP.

1.3 BDI Definition
The BDI architecture (Beliefs, Desires and Intentions) is based on the cognitive model proposed by Bratman (see [5] [6]). It describes to the agents in terms of three particular mental states:

- **Beliefs**: it is the information that the agent has about its surroundings.
- **Desires**: they are motivations of the agent, things or events that the agent wants to obtain. A subgroup of related desires to each other conforms an objective that the agent persecutes. Also denominated “goal”.
- **Intentions**: Intentions: they are the objectives chosen by the agent. Which it is committed to obtain.

Typically, an interpreter [6] inside of BDI agent will carry out some functions:

- **To generate options**: it reads the queue of events and returns a list of options.
- **To deliberate**: it selects an action subgroup to be adopted.
- **To update intentions**: it adds the subgroup obtained to the structure of intentions.
- **To execute**: if the intention exists to realize an atomic strike at this time, the agent executes it.
- **To obtain new external events**: any external event that has taken place during the cycle of interpreter is added to the queue of events. The internal events are added in the same sequence as these happen.
- **To eliminate successes**: it modifies the structures of intentions and desires, leaving successful desires and the satisfied intentions.
- **To eliminate impossible**: it modifies the structures of intentions and desires, leaving impossible desires and the unrealizable intentions.

2 Architecture and Methodology
In order to form an idea of the problem to model, we will take as example a CSP distributed in four agents. In the same way that in [4], we will suppose each agent has exactly one variable, all constraints are binary and each agent knows all constraint predicates relevant to its variable. Figure 1 illustrates the previous idea, normally a DCSP will be modeled as a non directed graph where each node represents an agent who handles a unique variable within a domain, and each edge represents the restrictions between each pair of variables.

![Fig. 1: DSCP Example](image)

In the present work, the architecture of DCSP agent will be the used one by PRS system created by Georgeff and Lansky [19], which is informed in Figure 2, where an agent receives stimuli from the outside through sensors and on the basis of their sets of beliefs, objectives, plans and intentions, conducts actions by means of effectors.

Using AAII [20] methodology for the development of agents oriented systems, it is observed that exists a variable number of instances of the agent who will depend in particular on the DCSP (in the case of Figure 1 we have four instances of the agent).

![Fig. 2: DCSP agent Structure](image)

In the development of complex systems becomes necessary to adopt a methodology to obtain a clear picture of the problem to solve. It was selected for the AAII methodology (created by the Australian Institute of Artificial Intelligence [4]), which specifically allows the creation of BDI agents. This methodology includes two views of the system, each of which are different models:
• Externally includes a model of agents and a model of interaction between agents.

• Internal point of view includes a beliefs model, a targets model and a model of plans.

3 System Modeling

3.1 The algorithm: "Asynchronous Backtracking"

One of the classic algorithms for solving DCSP is Asynchronous Backtracking, which is proposed and described in more detail in [8] and is used to model and implement this work. In this algorithm, as an initial step, each agent varies concurrently and sends its value to the agents to which it is connected by outgoing links, they will "evaluate" of the value is being sent.

Each agent has a priority defined by the alphabetical order or sequence of the identifiers of the agents and a set of values of the agents are connected via inbound links. Knowing that these values will receive messages as they represent the view of the agent, the agent will try to find a value that satisfies these restrictions based on their vision. If the agent is not able to find a value, then "revert back" by sending a message "nogood" to the lower neighbor, priority that connects to it via an inbound link to that change its value.

A message "nogood" has attached information this is the subset of the vision of the agent that triggered the "nogood". Thus, upon receiving a message "nogood", the receiver changes its value only if the "nogood" is compatible with their vision and their allocation, while this view adds to a list of "bad values" so this combination of values does not happen again.

If a message "nogood" received an agent which is not contained in his vision, then a new link is established becoming the evaluator of the found agent in this information.

3.2 External point of view

From the external point of view, the system is decomposed into agents that must be modeled, as well as their instances and the interactions between them. Thus the DCSP model general agent shall be as shown in Figure 3, the circle at the origin of the dotted arrow indicates that may be multiple instances of the agent.

The model of interaction between players comes with three different messages that can be exchanged between them:

- Inform the value that the agent now has to its neighbors, either when initializing or when it changes.
- If you can not find a value, the agent may reject the value received from a higher priority agent (nogood), enclosing the vision that created this situation.
- If when you receive a "nogood", the agent received the vision in the value of an agent who is not among its list of neighbors with current restrictions, the agent calls out for a link between the agent and found itself.

The first and second mentioned interactions are modeled in Figure 4

3.3 Internal Point of view

To make the model of beliefs, some parameters that would be useful to the agent are: (i) the present value of its variable, (ii) the priority that the agent compared to its neighbors (iii) the domain of assigned variable, (iv) the list of neighbors agents (v) the restrictions of the neighbors a priority greater than or equal to itself and (vi) the view that the agent of the current situation in which the resolution trouble is.

In [1] also are identified three beliefs that the agent may form at any moment: (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.

The main objective of each agent is to get a value that is within its domain, and that is not part of a listed vision as a "bad values", it means setting got by the agents of which is evaluator, could lead to a possible solution. That is why one of the objectives that are included in this model is to achieve a correct statement of vision: Stado_Vision_ok (i)!

To achieve its targets, the agent must do plans that are modeled for each of them. So when you want to get a correct view of the agent, or of the neighbors who have sent messages of "nogood", the agent will activate plans to seek a vision that isn’t listed under the "bad values" and that is within its domain.

Another plan to implement is to establish a link with those agents that have been received within visions of "nogood" and are not part of its list of neighbors with restrictions.

4 Applications and results

4.1 CSP testing problems: N-Queens and Sudoku

When including One of the most studied problems in the field of CSP’s is the N-Queens, is for this reason that the implementation of the proposed design has been selected as a test case.

The problem of the n-queens is a board “n x n” where n queens must be located in positions such that these are not attacking each other, considering that they move as they would on a chessboard.

The model we’ll use for this problem is standard. N variables are considered (one queen per row), and therefore, n agents, where the domain of each is: column that is positioned in the queen and the restrictions are:

\[ x_i \neq x_j, \forall i, j \in \{1, ..., n\} \land i \neq j \]

\[ |x_i - x_j| \neq |i - j|, \forall i, j \in \{1, ..., n\} \land i \neq j \]

In Figure 6 is a representation of the problem with n = 4, distributed in four agents, where the graph (a) shows the agents and the values of its variables and the diagram (b) represents the board.

![Fig. 6: Example 4-Queens.](image)

The problem of Sudoku is a game that consists of a square N x N cells, where N are square numbers n, it means, n2 = N, divided into six smaller boxes of n x n. The idea of the game is to complete the square with numbers ranging from 1 to N, but neither repeats in rows and columns of the main table, or inside boxes n x n.
Then the restrictions are that each value of the board must be different to those of the same row, the same column, and the same submatrix. This is shown in Figure 7 for an example of a Sudoku order $N = 4$, where each $x_{ij}$ represents a cell of the board, and the outgoing arrows of the $x_{22}$ agent representing the messages that arrive and depart from and to the same.

![Fig. 7: Example of Sudoku.](image)

### 4.2 Implementation and Testing

To implement BDI agents, was used the JADEX [5] platform creating as many agents as $N$ does (number of queens). Tests have been carried out and comparisons with other techniques, mainly in terms of time to measure the performance of the system implemented.

Table 1 provides an overview of the first tests conducted on the problem, revealing the differences in time to delay the search; between intentional use of the system that implements the distributed algorithm Asynchronous Backtracking (ABT), and centralized backtracking algorithm with a timeout of 5 minutes (300,000 ms.). One can see that the benefits of using a distributed system are visible only when the number of queens are larger than 11.

A second test was performed for the problem of Sudoku, specifically to order $= 4$, then to order $= 9$ and older, the time it takes the computer search is too high. To this problem requires 16 agents ($N \times N$), and the results are shown in Table 2.

<table>
<thead>
<tr>
<th>Order (N)</th>
<th>ABT Intentional Time (ms.)</th>
<th>Centralized Backtracking Time (ms.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>637</td>
<td>82</td>
</tr>
</tbody>
</table>

Table 2: Results intentional ABT v/s centralized backtracking Sudoku.

### 5 Conclusions and Future Work

A multiagent system has been generated, based on BDI agent that is capable of solving a CSP problem as the N-Queens is, obtaining the best results when comparing the number of queens $N$ increases above 11. Not justify the effort and the number of messages generated by the agents to solve when this problem is very small, but it is a good indication to assume that the system behaves satisfactorily in bigger problems. However, as the number of agents increases, the number of messages increases significantly, and the algorithm is not efficient enough because if those agents with lower priority are those with problems, should be tested before a large number of combinations to realize this. There are other distributed algorithms, such as the "Asynchronous Weak-Commitment" also proposed in [8] or some variation to the algorithm "Asynchronous backtracking" that could improve the system performance.

It is necessary to mentioned that the model is not directly applicable to all CSP, because there are many restrictions that have n-ary, for which it is necessary covert them into binaries, which increases the number of agents and therefore it is expected that this increase the complexity of the problem does not lead to results as satisfactory as it is for binary problems.

As future works is the implementation and experimentation with other CSPs, especially those with the condition that they are not binary, and comparing their results with other techniques either centralized or distributed.

### 6 References


