ANNALS OF THE FACULTY OF ENGINEERING HUNEDOARA
2004, Tome II, Fascicole 2

EDUCATIONAL SOFTWARE MEANT FOR THE IMPLEMENTATION AND EVALUATION OF THE ASYNCHRONOUS TECHNIQUES OF THE ABT FAMILY, CREATED IN NETLOGO

[1] MUSCALAGIU Ionel, MUSCALAGIU Diana,
PĂNOIU Manuela, IORDAN Anca

[1] THE UNIVERSITY „POLITEHNICA" DIN TIMISOARA,
THE FACULTY OF ENGINEERING OF HUNEDOARA

ABSTRACT:

The wide spreading of computer networks and of the Internet will result in the necessity of developing distributed software, which is supposed to work under these media, and to turn into account the advantages of a distributed and concurrent medium. Writing distributed applications is a lot more difficult than classical programming as it implies handling several different concepts. The implementation of such techniques can be done in any programming language allowing a distributed programming, such as Java, by means of RMI. Nevertheless, for the study of such techniques, for the analysis of their completeness and for their evaluation, it is easier and more efficient to implement the techniques under certain distributed media, which offer various facilities, such as NetLogo.

The aim of this article is to introduce an as general as possible model of implementation and evaluation for the asynchronous techniques of the ABT family. This model can be used in creating some educational software to be used in the study of asynchronous techniques with agents.

Keywords
Artificial intelligence, distributed programming, constraints, agents.

1. INTRODUCTION

The adjustment of the software technologies to the distributed equipment represents an important challenge for the next years. The wide spreading of computer networks and of the Internet will result in the necessity of developing distributed software, which is supposed to work under these media, and to turn into account the advantages of a distributed and concurrent medium. Writing distributed applications is a lot more difficult than classical programming as it implies handling several different concepts.

The constraint programming is a model of the software technologies, used to describe and solve large classes of problems as, for instance, searching problems, combinatorial problems, planning problems, etc. A large variety of problems in the A.I field and other domains specific to computer sciences could be regarded as a special case of constraint programming. Lately, the A.I community showed a greater interest towards the distributed problems that are solvable through modeling by constraints and agents. The idea of sharing various parts of the problem between agents that act independently and that collaborate between them using messages, in the prospective of gaining the solution, proved itself useful, as it conducted to obtaining a new modeling type called Distributed Constraint Satisfaction Problem(DCSP) [3,4].

There are more complete or incomplete asynchronous searching techniques available, for the DCSP modeling, which allow the solving of a problem in this constraints network. These techniques are remarked because of the diversity of applied ideas concerning the agents' work in an asynchronous and competitive way, but they also assure the completeness or a better efficiency, too.

The implementation of such techniques can be done in any programming language allowing a distributed programming, such as Java, by means of RMI. Nevertheless, for the study of such techniques, for the analysis of their completeness and for their evaluation, it is easier and more efficient to implement the techniques under certain distributed media, which offer various facilities, such as NetLogo [5,6].

The aim of this article is to introduce an as general as possible model of implementation and evaluation for the asynchronous techniques of the ABT family. This model can be used in creating some educational software to be used in the study of asynchronous techniques with agents. For this purpose we chose the NetLogo medium [5,6], which is a programmable modeling medium that can be used for the simulation of natural or social phenomena.

We will see the way one can simulate agents, how constraints can be implemented, how various measurement units for asynchronous techniques in the ABT family can be implemented. Unfortunately, there is no distributed medium dedicated to modeling with distributed constraints, all the existent media are general ones, with more general targets. The implementation of agents and constraints implies a certain calculation effort, bigger or smaller, according to the performances of the given medium. The use of this support for educational software can ease the actual implementation of asynchronous techniques in ABT family, and why not, of the other asynchronous techniques.

2. THE IMPLEMENTATION OF APPLICATIONS WITH AGENTS IN NETLOGO.

We are going to introduce the way of implementing, using as a support Yokoo's ABT algorithm, applied to the problem of the n queens. Starting from this framework, one can implement build any educational software for other derivate techniques, used in solving certain problem.

2.1. The Implementation Of the ABT Family Algorithms In NetLogo.

We are going to introduce the way of implementing the ABT family algorithms in NetLogo, using the example of the problem with n queens. There are two aspects we will analyze. The first refers to the way in which we will program the asynchronous technique and the way of representing in NetLogo. This is going to be approached using in general objects of the ***turtle*** type. The second aspect is more closely connected to the problem to be solved and it refers to the way of interacting with the user, to the desktop. As to this aspect, NetLogo offers objects of the ***patch*** type, as well as various graphical controls. In any case, the patch type objects will allow us to simulate the surface of the application.

First, the agents will be represented by calling the breed type objects (as we know, they are turtle type variants). For our example, we will create the agents by using the ***breeds [queens])*** construction. This will allow programming each DCSP agent according to the ABT algorithm. More exactly, we will implement the three routines of the ABT algorithm, using the NetLogo-type algorithm. These routines will be applied for each asynchronous agent.

The initial NetLogo code is:

```
breeds [queens]
globals [no-more-messages tmp]
breeds [queens]
;rows, and cols go from [0..num-queens-1]
;message-queue contains the incoming messages. We take new ones out from the
head.
;col is the position of the queen on it's row.
;current-view is a list indexed by queen number [col0 col1 col2...] col = -1if
unknown.
;nogoods is a list of inconsistent positions [0 1 1 0 ... ] where 0 is good and 1 is
no-good.
;messages-recieved is the number of messages this queen has received.
queens-own [message-queue col current-view nogoods messages-received_ok
messages-received_nogood]
```

One can notice the way the agents called *queens* are built. For each agent we declare the variables and the structures of property data, which will be global variables for the agents. Note a queue-type structure existing for each agent. This queue (called message-queue) will contain all the ok and nogood type messages received by the respective agent. These queues have a very important role in detecting the algorithm end. As we know, the ABT algorithm (as most asynchronous algorithms) ends at the moment a pause appears in message sending. At the moment all the queues are empty, we can consider that the ABT algorithm is over.

As to the first aspect (the algorithm one) it is recommendable to define a procedure for initializing each agent, like in the sequence given below, used to initialize the ABT algorithm:

```
to setup-queens
;it creates agents
  create-custom-queens num-queens [
    set col 0 ;initial value is 0 for all.
    setxy (col - screen-edge-x) (screen-edge-y - who)
    …]
end
```

As to the interface aspect, we will use for the graphical representation of the queen positions a ***patch*** for each. It is recommendable to create a procedure of initializing the display surface for the agent values.
The two initializing procedures will be attached (by means of a set up procedure) to an application start button, as in the following sequence:

```
to setup
  ca
  setup-patches
  setup-queens
  ask queens [initialize]
end
```

In order to start running the algorithm we can build a go or update type button to which we attach a NetLogo procedure, representing some kind of “main program”, a command center. Within such a procedure, which is supposed to run continuously (until the queues run empty of messages), the message queue is checked for each agent (in order to detect a pause in the sending of the messages). For standardizing reasons, we defined another procedure, called ***handle-message***, in charge with handling the messages that are specific to the ABT algorithm. For another technique (which, as known, works with other types of messages) one can adapt this procedure in order to handle such messages. We introduce hereinafter the two procedures, maybe the most important ones from the standpoint of the way of implementing in NetLogo the asynchronous way of working with messages, which is characteristic for asynchronous techniques:

```
to update
  set no-more-messages true
  ask queens [
    if (not empty? message-queue)[
      set no-more-messages false]]
  if (no-more-messages) [stop]
  ask queens [handle-message]
  ask queens [
    create-temporary-plot-pen "q" + who
    plot messages-received_nogood
    plot messages-received_ok]
end
```

One can notice the call "ask queens …", which allows the asynchronous performing of the next calculations for each agent. The message handling procedure is:

```
 to handle-message
  locals [msg xj dj]
  if (empty? message-queue) [stop]
  set msg retrieve-message
  if (first msg = "ok")[

      set messages-received_ok messages-received_ok + 1
      ok xj dj ]
        if (first msg = "nogood")[
      nogood item 1 msg
        set messages-received_nogood messages-received_nogood + 1]
end
```

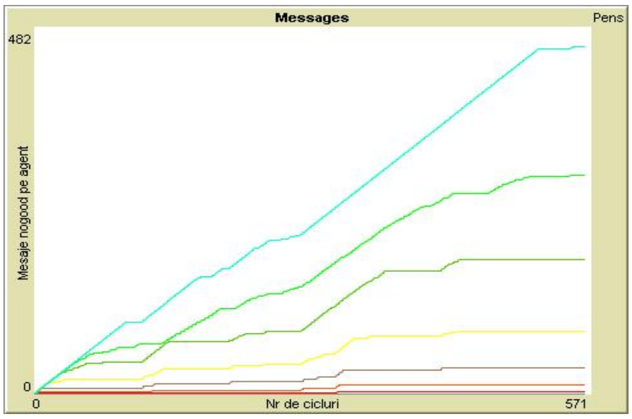
The main target is to take a message from the message queue (let's not forget that this is a FIFO-type of queue), identify the type of message and then call the procedure of processing the respective message.

The two processing procedures for the ok or nogood messages can be implemented according to the implemented algorithm. For instance, for the ABT algorithm we mentioned, the two procedures will look like this:

```
to ok [Qj Vj]
   set current-view replace-item Qj current-view Vj
   check-agent-view
end

to nogood [ beef ]
…
end
```

Another important thing that can be achieved in NetLogo is related to the evaluation of the asynchronous algorithms [1,2]. We are interested to know how we can count the costs of the implemented techniques. NetLogo allows the counting of the various types of messages that arise in the respective technique. For the algorithms in the ABT family, it is interesting to count the message flow such as the number of ok and nogood messages. This can be done using some global variable, attached to the agents. It is achievable for the model we are introducing, by means of a propriety variable for each agent, variable that has to be incremented at the moment one type of message is generated and sent. For our model, this can be done within the ***handle-message*** routine. Moreover, NetLogo also allows the graphic display of the variation of the number of messages for each agent, which is very useful in analyzing the influence of the order of agents on the message flow or on the behavior of the agents with respect to the unit of measurement. This can be achieved using a ***plot*** type graphic object, within which one can draw, for each agent, the number of values handled at each step. For instance, in the figure 1. given below, it is shown how the number of nogood messages per agent is counted for each cycle:



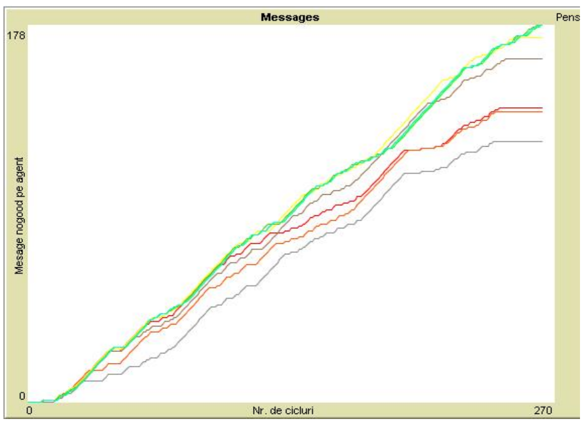
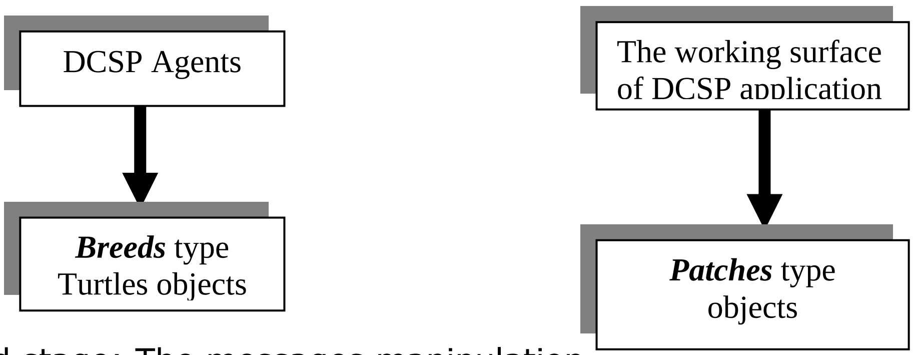


Figure 1. For the ABT and AWCS algorithm the evolution of nogood message flow for the problem of the n queens (n=8)

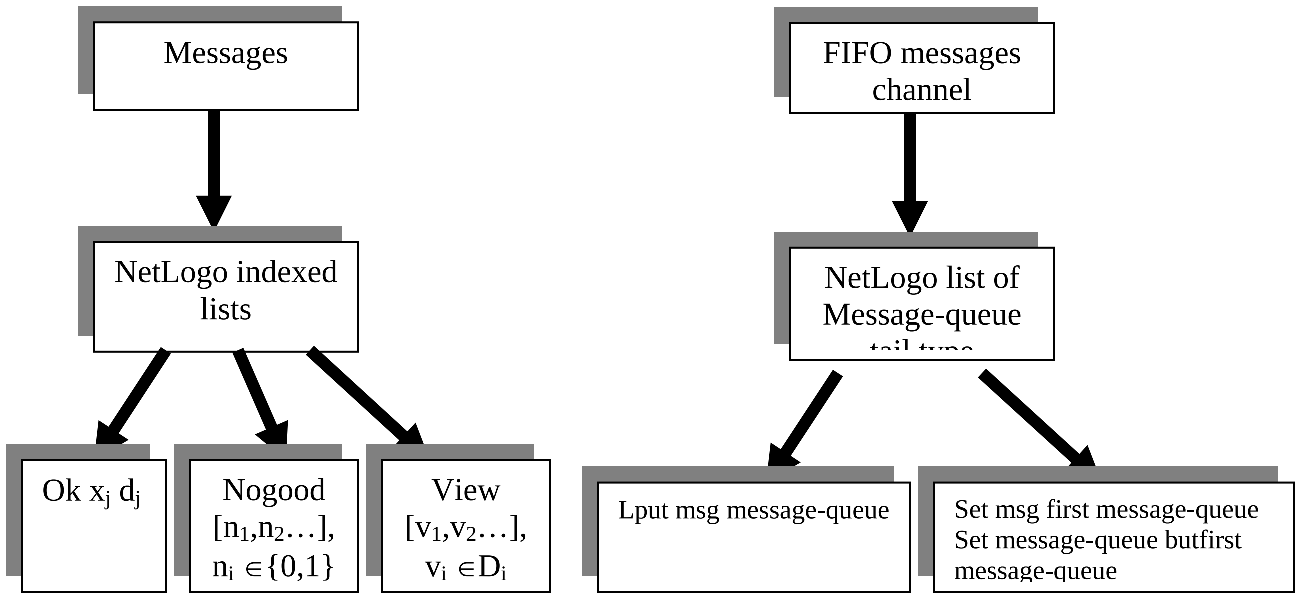
2.2. Model Of Implementation And Evaluation For the Asynchronous Techniques, to Be Applied In Solving A DCSP Problem.

In short, starting from the previous analysis and taking into account the particularities of the various asynchronous techniques, we give hereinafter a diagram of asynchronous algorithm implementation for NetLogo, diagram that can be used in implementing some educational software for the study of asynchronous techniques.

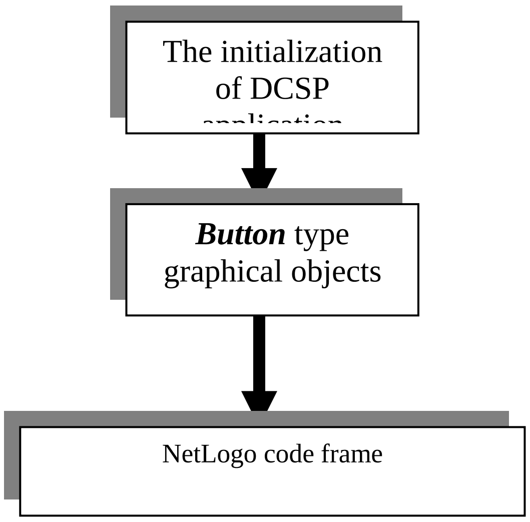
The first stage: Identifying the objects of DCSP application

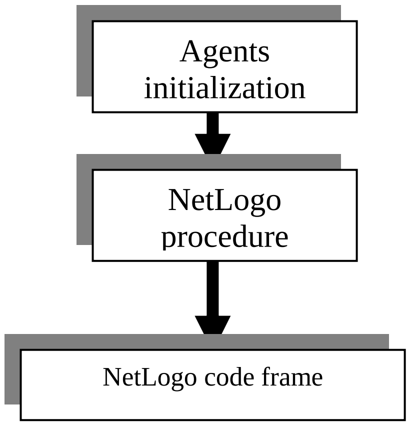

DCSP Agents
Breeds type Turtles objects
The working surface of DCSP application
Patches type objects

The second stage: The messages manipulation.

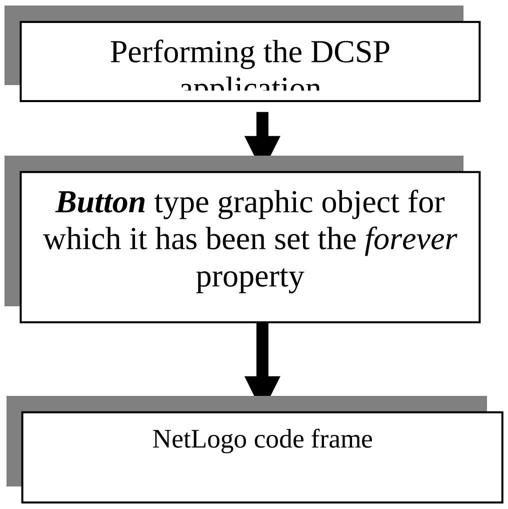

Messages
NetLogo indexed lists
Ok x_j d_j
Nogood $[n_1,n_2\ldots]$, $n_i \in \{0,1\}$
View $[v_1,v_2\ldots]$, $v_i \in D_i$
FIFO messages channel
NetLogo list of Message-queue
Lput msg message-queue
Set msg first message-queue
Set message-queue butfirst message-queue

The third stage: Initializing the application and each agent. The main program of DCSP application.

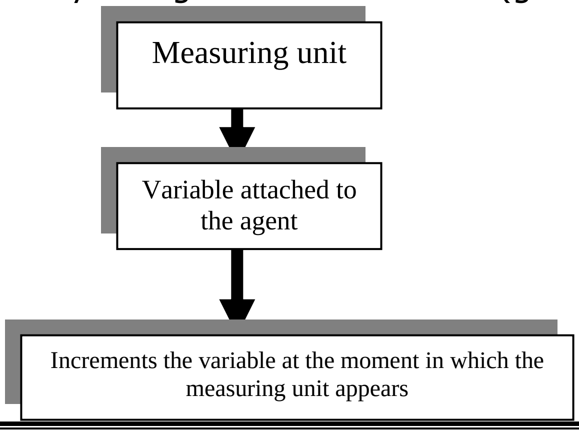

The initialization of DCSP
Button type graphical objects
NetLogo code frame



Proceeding with the application applies the introduction of a graphic button type object and setting the *forever* property. This way the code that will be attached in the form of a NetLogo procedure (that applies to each agent) in continuous action, until the emptying of messages tails and encountering the *stop* command (which, in NetLogo, stops the execution of an agent).


Performing the DCSP
Button type graphic object for which it has been set the *forever* property
NetLogo code frame

The fourth stage:The countdown of the asynchronous technique costs. This thing could be done by using certain variables (globals eventually) attached to agents.


Measuring unit
Variable attached to the agent
Increments the variable at the moment in which the measuring unit appears

3. CONCLUSIONS

The NetLogo Medium is good for building software packages (simulators) meant for testing and evaluating the asynchronous techniques. The Medium and the language backing it have enough resources to implement any DCSP technique, for any problem that has to be solved.

The model we suggested supposed the identification of the NetLogo objects needed for the implementation of the asynchronous techniques (agents, messages, message queues, agent order), as well as of the user interface. Also, we tried to count the various units of measurement in order to estimate and assess the performances of the asynchronous techniques. We consider this to be the first model for the NetLogo techniques, allowing evaluation according to several evaluation criteria. At the same time, the model enables the study of the agent behavior for various techniques, and the study of the costs for each agent.

As a general conclusion, we think that the model we achieved can be used for the study and analysis of the asynchronous techniques, the model allowing their complete evaluation.

4. BIBLIOGRAPHY

[1] MUSCALAGIU Ionel: *A comparative study on the efficiency of Asynchronous Search Techniques in the DISTRIBUTED CONSTRAINT SATISFACTION PROBLEM*. International Conference on Automation, Quality and Testings, Robotics , 2004, Cluj-Napoca.

[2] MUSCALAGIU Ionel, PANOIU M., OSACI M. *The analisys of the evaluating criteria in the asynchronous techniques efficiency*. Peridodica Politechnica, Transactions on Automatic Control and Computer Science, Vol.49 (63), 2004.

[3] YOKOO M., E.H. DURFEE, T. ISHIDA, K. KUWABARA (1998): *The distributed constraint satisfaction problem: formalization and algorithms*. IEEE Transactions on Knowledge and Data Engineering 10 (5).

[4] YOKOO Makoto (2001): *Distributed Constraint Satisfaction-Foundation of Cooperation in Multi-agent Systems*. Springer.

[5] http://ccl.northwestern.edu/netlogo/docs/.

[6] WILENSKY, U. *NetLogo itself:* NetLogo. http://ccl.northwestern.edu/ netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.