A Distributed Constraint Reasoning approach towards intelligent marketplace environment

Zakarya Erraji[1], Amal Hakkou[1], Amine Benamrane[1], Imade Benelallam[1,2], and El Houssine Bouyakhf[1]

[1]LIMIARF, FSR,University Mohammed V, Rabat, Morocco
[2]SI2M Laboratory, INSEA, Rabat, Morocco
{zakarya.erraji.e,imade.benelallam}@ieee.org,
{hk.amal221,benamraneamine}@gmail.com,
bouyakhf@mtds.com

Abstract. With the growing success of e-marketplace adoption, the need for new intelligent approaches to support both buying and selling goods or services become a fundamental requirement. In the recent past, several techniques have been proposed, in different research areas, allowing the simulation of a market with a set of sellers proposing products and buyers who have a list of interests. In this paper, we propose a new problem formulation and Distributed Constraint Reasoning protocol to deal with e-marketplace issues. More specifically, our attention is focused on constraint-based multi-agent approach offering a flexible and confidential multi-lateral negotiation mechanism, namely, ABT-Trader. The satisfaction of strict preferences offers the best negotiation to buy all the wanted products, otherwise, constraint and variable relaxations are adopted to look for feasible solutions. This approach has been implemented and tested on JChoc Platform using preliminary generated problems. The experimental results show that our approach is of practical interest: it proposes feasible time-tested solutions.

Keywords: Intelligent Marketplace, Constraint Programming (CP), Multi-Agent Systems, Distributed Problem Solving, Agent Models and Architectures, Distributed Constraint Reasoning.

1 Introduction

Electronic marketplace is a place where customers and suppliers can meet, negotiate, make decision and transact as in a traditional marketplace. With the growing need of e- marketplace in our daily life, the need for new innovative techniques to support both customers and suppliers in buying and selling goods or services is increasing quickly.
In the recent past, several platforms [8] where people can look for a given good or service has changed the traditional ways of negotiating and doing business. These various forms of e-marketplaces are considered as revolutionary vectors of e- marketplace technology. However, most of them only serve a small part of the

transaction process and need human intervention before or during negotiation process. Nowadays, autonomous and intelligent multi-agent systems make the business processes in e-market more efficient and revolutionary [11] and citeHemaissia09. These new sophisticated systems may replace both buyers and sellers on decision making in order to reach a negotiated agreement. In particular, virtual sellers and buyers can be provided with a negotiation protocol that help them to specify constraints and then to look for the optimal/pseudo-optimal decision regarding these specified preferences (constraints). When the combinations of multi-lateral negotiation become complicated, and the constraints both on the customers and suppliers sides become vague and complex, intelligent negotiation gives rise to new challenges for developers of architecture and software technologies underlying e-marketplaces. Hence, intelligent agents should be initially created with their complete set of strategies and should be able to learn and make decision; rather than having an automation behaviour, they should have the ability to acquire experience from previous negotiations they've conducted. Although a completely uncontrolled e-marketplaces are still rather a vision than reality.
To achieve this challenge and remove these limitations, some projects have been initiated, such as SICS MarketSpace [8]. Afterwards, many negotiation mechanisms have been proposed in the literature [6], [7], [11] and [9]. The most important work has been done on negotiation process and the formulation model still unripe and is not applicable in realistic use environments.
One of the most promising answer to the new challenges posed by arising e-marketplaces is "Constraint Programming" paradigm. Several papers have proposed Distributed Constraint Reasoning (DCR) as a paradigm for problem modeling and solving in framework of Multi-agent Systems. One of the main features of DCR approaches is its distributed nature in which a set of intelligent agents can each make local decisions and communicate in order to improve global decision problem. To the best of our knowledge, few works have been interested to the formulation of e-marketplace environment using Distributed Constraint Reasoning techniques [1] [10].
In this paper, we propose a new problem formulation and DCR protocol to deal with e-marketplace issues. More specifically, our attention is focused on constraint-based multi-agent approach offering a dynamic and privacy multi-lateral negotiation mechanism, namely, ABT-Trader. Also, the framework is based on two resolution phases, the registration and building network phase, in which each agent builds its distributed constraint network, and the negotiation phase, in which the sellers and buyers agents negotiate using the distributed constraint based techniques reasoning. The satisfaction of strict preferences offers the best negotiation to buy all the wanted products, otherwise, constraint and variable relaxations are adopted to look for a feasible solution.
The remainder of this paper is organized as follows. Section 2 presents the background of Distributed Constraint Reasoning. Section 3 describes the main problem of e-marketplace environment, and the important issues to fully automated negotiation. Section 4 addresses the distributed constraint reasoning model us-

ing privacy and relaxation concepts. Section 5 focus on constraint resolution protocol as a negotiation mechanism. Section 6 discusses preliminary experiment evaluation, and section 7 gives a brief overview of related works. Finally, section 8 provides a conclusion and areas for further research.

2 Preliminaries

Constraint Programming distinguishes between the description of the constraints involved in a problem on the one hand, and the algorithms and heuristics used to solve the problem on the other hand. Modelling and solving problems is through a very elegant mathematical formalism, based on Constraint Satisfaction Problems CSPs.
A Distributed Constraint Satisfaction Problem (DisCSP) is represented by a constraint network CSPs where variables and constraints are distributed among multiple automated agents.

Definition: A DisCSP (or a distributed constraint network) has been formalized as a tuple(A, X, D, C, ψ), where:

- $A = \{A_1, ..., A_p\}$ is a set of p agents.
- $X = \{x_1, ..., x_n\}$ is a set of n variables such that each variable x_i is controlled by one agent in A.
- $D = \{D(x_1), ..., D(x_n)\}$ is a set of current domains, where $D(x_i)$ is a finite set of possible values for variable x_i.
- $C = \{C_1, ..., C_m\}$ is a set of m constraints that specify the combinations of values allowed for the variables they involve. We note that the constraints are distributed among the automated agents. Hence, constraints divide into two broad classes: inter-agent (Totally Known Constraints and Partially Known Constraints) and intra-agent. agents.
- $\psi : X \longmapsto A$ is a function that maps each variable to its agent.

Also in a dynamic environment a DisCSP may change over time, these changes could be due to addition and/or deletion (relaxation) of variables, values, constraints, or agents. The Distributed and Dynamic Constraint Satisfaction Problems (DDisCSPs) can be described as a five tuple (X, D, C, A, δ) where :

- A, X, D, C, and ψ remain as described in DisCSP

- δ is the change function which introduces changes.

Many DDisCSPs approaches (e.i : DynABT [13])are proposed to solve such type of problems, and can be easily implemented in This platform.

A solution to a DisCSP is an assignment of a value from its domain to every variable of the distributed constraint network, in such a way that every constraint is satisfied. Solutions to DisCSPs can be found by searching through the possible assignments of values to variables such as ABT algorithm [2].

Regarding constraints privacy, we adopte the Partially Known Constraints (PKC) [12], when the scope $variables(C_{ij})$ of each constraint is known by every related agent, but the relation $relation(C_{ij})$ is partially known. The model is as follows. A constraint C_{ij} is only partially known by its related agents. From $C_{i(j)}$ with agent j as,

$$variables(C_{i(j)} = \{x_i, (x_j)\} \qquad relation(C_{ij} \subseteq relation(C_{i(j)})$$

Where $(x_j$ in $variables(C_{i(j)}$ means that agent i knows little about the other variable of the constraint. From constraint C_{ij}, agent j knows the constraint $C_{(i)j}$,

$$variables(C_{(i)j} = \{(x_i), x_j\} \qquad relation(C_{ij} \subseteq relation(C_{(i)j})$$

It is required that,

$$relation(C_{ij}) = relation(C_{i(j)}) \cap relation(C_{(i)j})$$

3 Problem statement

Electronic marketplace is a place where customers and suppliers can meet, negotiate, make decision and transact as in a traditional marketplace. According to the literature, the classification of multi-agent e-marketplaces are based on three characteristics:

- Character of negotiation: On either side - on the buyers' and on the sellers' side - one or more participants may be negotiating; multilateral negotiation or bilateral negotiation.
- Number of issues: This characteristic represents the number of negotiation issues. In the simplest case, the negotiation can be reached over one-dimensional issue (price). In more complicated cases, the negotiation can be reached over multi-dimensional issues (related to price, quality, terms and conditions, etc.).
- Level of preferences : the preferences regarding the negotiation issues may be crisp or fuzzy.

In this paper, we focus on intelligent electronic marketplaces and furthermore, on the special case of multi-agent systems negotiation. We assume that :

- A multilateral negotiation space;
- A multi-dimensional issues(for simplification assumption we consider only the price attribute and for the other attributes quality, terms and conditions, etc. can be considered in an extension formulation);
- A strict preference model that can be relaxed during negotiation.

The problem can be described as bellow :

- An actor in the market can be whether seller or buyer.

- An actor can trade many products (i.e. goods or services).
- Each seller can not sell a product less than his corresponding lowest price.
- Each buyer can not buy a product more than his corresponding highest price. We assume that this assumption can we relaxed during resolution.
- Each buyer have his fixed budget that can not exceeded.
- Each buyer can buy a limited number of products depending on his budget and the priority of products those he wants to buy.
- Each buyer can choose the best received offer for a product using different criterias (e.g. warranty, quality, delivery, price, etc). In our model, we assume that except the price all criterias are equi priority. Thus, the offer evaluation is based only on the price.
- Each agent preference is considered as private.

4 Distributed constraints reasoning model

The first contribution of this paper is to design the problem already explained as a Distributed Constraint problem, to do this, we will translate the different components of the problem as a set of <Agents, Variables, Domains, Constraints> such that the model describes exactly our problem. The model can be done in various ways, therefore we propose the one that we see the clearest, the simplest and the most efficient.

4.1 Agents

$A = \{Seller_1, ..., Seller_p, Buyer_1.Buyer_q\}$ is a set of p sellers and q buyers.

4.2 Variables

We assume that :

- $X_j^{Buyer_i}$ is the j^{th} product needed by the buyer i.
- $X_l^{Seller_k}$ is the k^{th} product offered by the seller k.

X is the set of n variables $X_j^{Actor_i}$ that correspond to the j^{th} product of the i^{th} actor.

A variable is the product that can be a service or a good.

4.3 Domains

Each product has a set of possible prices. The price is not always fixed or known. As a variable, the product in the seller side can take a value from a set of prices those present his domain. Each buyer has an idea about the possible prices for each product, Those prices present the domain of a product for a buyer. So, the domain can be whether an interval value or a singleton.

4.4 Constraints

In our model the set of the constraints is defined as bellow:

$$C = \{C^{seller}_{intra}, C^{seller}_{inter}, C^{buyer}_{intra}, C^{buyer}_{inter}\}$$

– We assume that there are no intra constraints for a seller agent, means that, he can sell any product independently, thus

$$C^{seller}_{intra} = \varnothing$$

– To preserve the seller confidentiality, the inter constraints for a seller are partially known constraints and buyers can not see these constraints, says that the seller can not sell a product with a price lower than the smallest value in his domain thus

$$\begin{gathered} C^{seller}_{inter} = \{C_{seller_i, buyer_j} = \{X_k^{Buyer_j} \geq min(D(X_k^{Seller_i}))\} \\ \diagdown \text{ for each seller i offering the product k interesting the buyer j}\} \end{gathered}$$

– The buyer has to respect his budget. For each buyer j we assume the intra constraint as bellow:

$$\begin{gathered} C^{buyer}_{intra} = \{\sum_{i=1}^{k} X_i^{buyer_j} \leq \text{Budget}(Buyer_j)\} \\ \diagdown \text{ for each buyer j} \end{gathered}$$

– For the same reasons of confidentiality (constraint's privacy), we assume the inter constraints those avoid the sellers to sell a product with a price lower than the smallest value in his domain, thus

$$\begin{gathered} C^{buyer}_{inter} = \{C_{buyer_i, seller_j} = \{X_k^{seller_j} \geq max(D(X_k^{Buyer_i}))\} \\ \diagdown \text{ for the seller i offering the product k interesting the buyer j}\} \end{gathered}$$

4.5 Agent priority

In our model, there is two levels of priority 1 and 2. Each buyer is in the level 2 of priority, and has a set of parents, where each parent is a seller. All sellers have priority 1. Sellers can sell products to different buyers in parallel, and the buyer can negotiate with different sellers in parallel too.

4.6 Value Ordering heuristics

While the negotiation process, sellers select the values from their domains in descending order. However, the buyers choose the lowest proposed offers (i.e. prices) sent by the sellers for each product.

4.7 Variable and constraint relaxation

In case of unsolved problem, a buyer could remove variables (i.e. products) according to a variable priority heuristic to look for possible products with highest priority, and those satisfy the budget constraint. In this case, the inter constraints of the buyer could be relaxed.

5 Resolution protocol

In the resolution protocol we can observe two phases, the first phase is the building of the Distributed Constraint Satisfaction Problem network, and the second phase is the negotiation process. The registration and negotiation phases presented in the figures 1 and 2 show two types of agents: buyer and seller.

5.1 Registration process

The first phase of the solving process is divided into two parts: the registration and the network building. Every trader in the marketplace (seller or buyer) must be registered. This registration allows the manager of the marketplace to know the needs of each buyer, and the products proposed by each seller, thus, the building of the DisCSP network by linking between them.

For example, the buyer 1 registers himself to the marketplace and communicates his needs (e.g. product 1, product 3 and product 9), thereafter the manager sends to the buyer 1 a list of sellers for product 1, 3, and 9.

5.2 Negotiation process

The negotiation process starts when the marketplace manager sends for each buyer a list of sellers offering the needed products.

To simplify the explanation of our negotiation process let's consider a buyer 1 that wants to trade asynchronously with two sellers for 2 products (i.e. product 1 and 9) with a fixed budget b where price(product 1) + price(product 9)$\leq$b.

The first step for the buyer 1 is to ask all received sellers to send their offers, then he chooses the best offer for each product. If all constraints are satisfied then the buyer terminates his negotiation. Otherwise, for each product, he send a message to the best seller to propose a better offer while he has not sent a fixed price, in the other case (i.e. the seller has already sent a fixed price for this product), he chooses another seller who proposes the best offer and who have not sent a fixed price to resend a better offer for this product. While the solving, if all sellers send their fixed price and the buyer can not satisfy all his constraints then he tries to buy only the products those have a high priority for him and those satisfy his constraints.

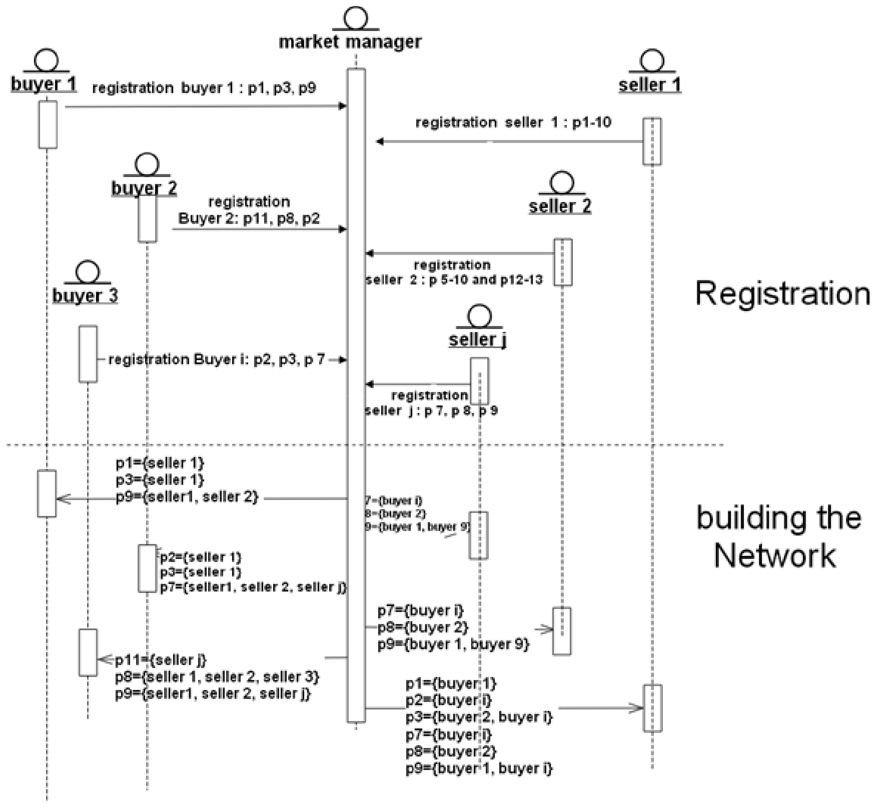


Fig. 1. Registration and building the DCSP network

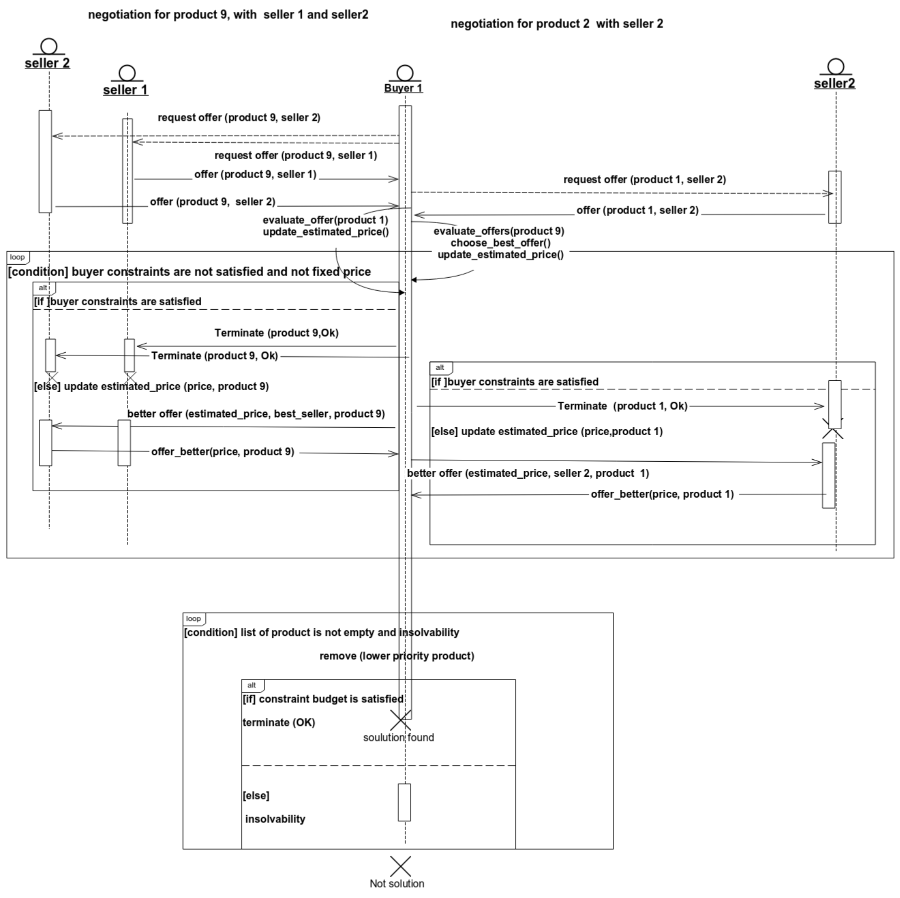

negotiation for product 9, with seller 1 and seller2
negotiation for product 2 with seller 2
seller 2
seller 1
Buyer 1
seller2
request offer (product 9, seller 2)
request offer (product 9, seller 1)
offer (product 9, seller 1)
offer (product 9, seller 2)
request offer (product 1, seller 2)
offer (product 1, seller 2)
evaluate_offer(product 1)
update_estimated_price()
evaluate_offers(product 9)
choose_best_offer()
update_estimated_price()
loop
[condition] buyer constraints are not satisfied and not fixed price
alt
[if]buyer constraints are satisfied
Terminate (product 9,Ok)
Terminate (product 9, Ok)
[else] update estimated_price (price, product 9)
better offer (estimated_price, best_seller, product 9)
offer_better(price, product 9)
alt
[if]buyer constraints are satisfied
Terminate (product 1, Ok)
[else] update estimated_price (price,product 1)
better offer (estimated_price, seller 2, product 1)
offer_better(price, product 1)
loop
[condition] list of product is not empty and insolvability
remove (lower priority product)
alt
[if] constraint budget is satisfied
terminate (OK)
soulution found
[else]
insolvability
Not solution

Fig. 2. Negotiation process

5.3 Algorithm description

The ABT-Trader is based on ABT [2] and used by two types of agents: buyer and seller. The pseudo code of ABT-Trader represented in Figures 3 and 4 says that the seller stores one nogood per removed value in the NogoodStore for each negotiation till the end of the resolution. All agents start the search by calling the procedure *Setup*() in which they initialize their nogoodStores, agentViews, finalAgentViews, products, etc. Then each agent checks his type (buyer or seller). If the agent is a buyer, in this case, he asks all available sellers of each product to be his parents and to negotiate with him about the price of this product, after that, he calls the main procedure named Buyer(), if not (i.e. the agent is a seller) he runs a loop where he waits for possible buyers, and when a buyer arrives (i.e. *sell*? message received), the agent seller creates a copy of all data (i.e. domains, nogoodStores, etc) and calls a new Seller() procedure to negotiate with this buyer about the product carried in the message *sell*? .
In the first time, the buyer waits for all sellers to propose their prices within a time out, then he tries to choose the best proposed values/prices and try to assign them. The buyer agent assign all values only when all constraints are satisfied, in this case, a solution is found. Otherwise, when there is a problem in this phase (i.e. the buyer can not assign all variables), he tries to send a nogood per product to the seller who have not sent a finalOK? message, and who have proposed the best price for this product, after that, he waits with a time out for the new propositions, in this part, he does not wait for all new proposition to arrieve, he tries to find a solution when a new proposition is received. In case of all sellers have proposed their final prices for all products (i.e. the size of the agentView equal the size of FinalAgentView) and there are some not satisfied constraints, the buyer uses the priority between products to buy just products who have a high priority and those satisfy budget constraint only.
While the negotiation, when the agent seller receives a nogood, he tests the size of values coherent with the current nogoodstore (the nogoodstores and domains are a copy of the initial data and will be removed in the end of the current negotiation), if the size is 1, he send a final price using the *FinalOk* message, if not, he decreases the chosen value and sends it using the message ok?. The agent seller starts the negotiation with choosing the biggest value and when he receives a nogood he decreases it. For the first time the buyer can propose the lowers prices found in his products domains, and the seller remove all possible values bigger then the proposed values from his domain while the size of value is greater than 1.

6 Experiment

6.1 Experimental platform

JChoc [3] is a JADE-based [4] platform. This platform is a distributed constraint multi-agent system, It can also be used to analyze and test algorithms proposed

Algorithm 1 ABT_{Trader} Part1

```
procedure Setup()
initialize products, domains, myAgentViews, myFinalAgentView, nogoodStores, and type;
Register with all products and type.
if type=buyer then
    for all $product \in products$ do
        for all $seller \in sellers(product)$ do
            sendMsg : sell?(seller,product);
            add seller to the list of sellers
        end for
    end for
    Buyer()
else
    while true do
        $msg \leftarrow getMsg()$
        if msg.type=sell? then
            Seller(msg.Sender, msg.product);
        else
            msg.isRead$\leftarrow$ false
        end if
    end while
end if

procedure $Buyer()$
$myvalues \leftarrow enmpty$ ; $end \leftarrow false$;
while $\neg end$ do
    $msg \leftarrow getMsg()$
    switch (msg.type) do
        ok? : ProcessOk(msg);
        finalOk :ProcessFinalOk(msg);
end while

procedure $Seller(buyer, product)$
$ChooseValue(buyer, product)$ ; $end \leftarrow false$; copy nogoodStores and domains;
while $\neg end$ do
    $msg \leftarrow getMsg()$;
    switch (msg.type) do
        nogood : ResolveConflict(msg);
        stop : $end \leftarrow true$; delete current nogoodStores and domains;
end while

procedure $ProcessOk(msg)$
Update(myAgentView,msg.Assig);
CheckAgentView();

```

Fig. 3. Algorithm part 1

Algorithm 2 $ABT_{Trader}\ Part2$

```
 1: procedure $ProcessFinalOk(msg)$
 2: Update(myAgentView,msg.Assig);
 3: Update(myFinalAgentView,msg.Assig);
 4: CheckAgentView();
 5:
 6: procedure ResolveConflict(msg)
 7: if The size of values those not elimunated by the current nogoodStore $>1$ then
 8:    Add(msg.Nogood,myNogoodStore);
 9: end if
10: ChooseValue(buyer,product);
11:
12: procedure $ChooseValue(buyer, product)$
13: get all values $\in$ D(product) and not eliminated by the current goodStore
14: if values.size=1 then
15:    sendMsg: finalOk(buyer, values.getvalue, product);
16: else
17:    sendMsg: Ok?(buyer, values.getBiggestValue, product);
18: end if
19: procedure $CheckAgentView()$
20: if AgentView.size $\leq$ sellers.size and waiting_time $<$ time_out then
21:    Exite this Procedure;
22: else
23:    Choose the best prices from the agentView and try to assign them to myValues;
24:    if myvalue are not assigned then
25:       Choose the best price for each product and not in finalAgentView and send
          nogood with the my proposed price to its seller;
26:       if No nogood sent then
27:          Try to assign variables or products who have the high priority and send
             msg stop to all sellers;
28:          end $\leftarrow$ true;
29:       end if
30:    else
31:       Send msg stop to all sellers;
32:       end $\leftarrow$ true;
33:    end if
34: end if
```

Fig. 4. Algorithm part 2

by constraints programming community, and it allows the use of real communication channels. We have implemented the ABT-Trader in this platform, where all services are managed by the services management unit and the messages are transported between agents by the communication management unit.

6.2 Experimental Settings

We use XML files to describe the sub-problems for each agent (i.e. buyer or seller). The figure 5 shows an example for an agent seller and the figure 6 shows an example for an agent buyer before the network building. The inter constraints will be added during registration process. We have used in this experimentation two buyers named C1 and C2 and three sellers named F1, F2 and F3. The agent C1 wants to buy products X, Y, and Z. C1 has 115 as a budget, and has an estimated price about each product: D(X) = [100, 200], D(Y)=[150, 300], D(Z)=[30, 70]. In case of unsolved problem, he tries to buy the products X then Y and then Z(i.e X is more prioritized than Y, and Y is more prioritized than Z). The agent C2 wants to buy two products X and Z. C2 has 30 as a budget, and has an estimated price about each product: D(X) = [50, 100] and D(Y)=[10, 70]. Like C1 In case of unsolved problem, he tries to buy products X then Y (i.e X is more prioritized than Y). As a seller, F1 proposes two products X and Y, he can sell X from 25 to 50 and Y from 70 to 100. F2 proposes the three products X with a fixed price: 70, Y for 70 to 100, and Z for 40 to 60. Finally, F3 proposes X and Y as products, 80 to 120 for X, and 60 as a fixed price for Y.
We assume that there are no constraints between sellers or between buyers. And for a seller there are no intra constraints. Each agent was launched in a machine core i7 with 8Go of Ram, and they can communicate between them using the http protocol (e.i. Each machine is connected to internet).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<instance>
<presentation name="MSP" type="DisCSP" model="Complex" constraintModel="PKC"
     format="XDisCSP 1.0" />
<domains nbDomains="2">
        <domain name="Dx" nbValues="">25..50</domain>
      <domain name="Dy" nbValues="">70..100</domain>
</domains>
<variables nbVariables="12">
    <variable name="Xf1" id="1" domain="Dx" description="X" />
   <variable name="Yf1" id="2" domain="Dy" description="Y" />
</variables>
   <constraints nbConstraints="0">
   </constraints>

   <predicates nbPredicates="0">
   </predicates>

   <agents_neighbours>
   </agents_neighbours>
   <services service="Provider-X Provider-Y"></services>
</instance>
```

Fig. 5. seller F1 xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<instance>
<presentation name="MSP" type="DisCSP" model="Complex" constraintModel="PKC"
    format="XDisCSP 1.0" />
  <domains nbDomains="1">
      <domain name="Dx" nbValues="">100..200</domain>
    <domain name="Dy" nbValues="">150..300</domain>
    <domain name="Dz" nbValues="">30..70</domain>
  </domains>
  <variables nbVariables="12">
    <variable name="Xc1" id="1" domain="Dx" description="X" />
    <variable name="Yc1" id="2" domain="Dy" description="Y" />
    <variable name="Zc1" id="3" domain="Dz" description="Z" />
  </variables>

  <constraints nbConstraints="3">
    <constraint name="C1" reference="sum" scope="Xc1 Yc1 Zc1 115" arity="4"
        budget="115"/>
    <constraint name="C2" reference="priority"   scope="X Y Z" arity="3"/>
  </constraints>

  <predicates nbPredicates="0">
  </predicates>

  <agents_neighbours>
  </agents_neighbours>

  <services service="Customer-X Customer-Y Customer-Z"></services>
</instance>
```

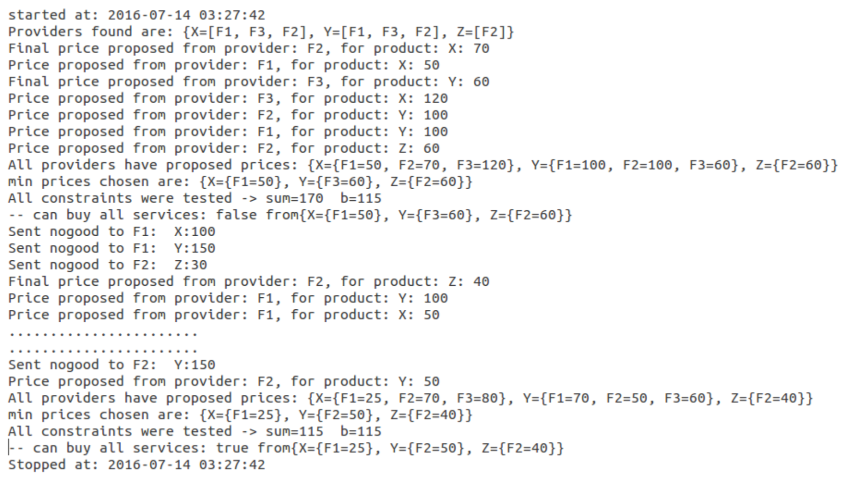
Fig. 6. buyer C1 xml file

6.3 Experimental Results

The solving phase is stopped by the agent buyer when finishing all negotiations. The figure 7 shows the start and the end of the buyer C1 negotiations, where he can buy all products: X from F1 for 25, Y from F2 for 50, and Z from F2 for 40, And the figure 8 shows the start and the end of the buyer C2 negotiations, as a result, he can buy only the product X from the seller F1 for 25. These preliminary results show the effectiveness of our approach, since the negotiation perform a relevant negotiated price. In a time tested performance.

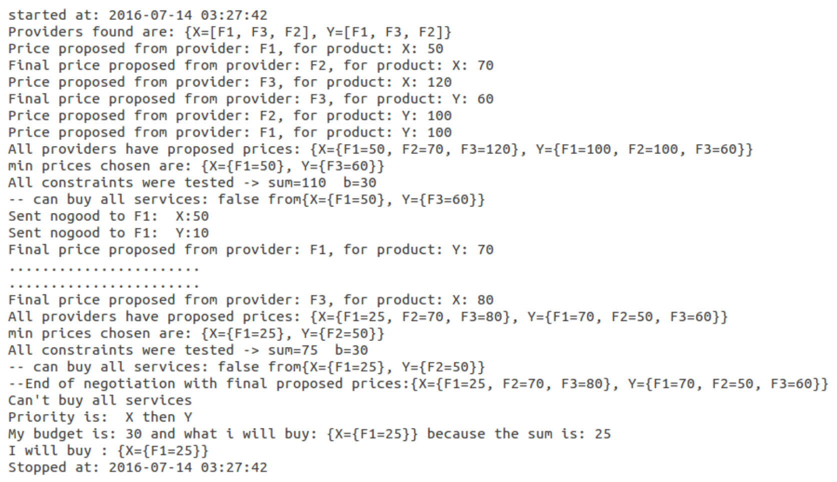
7 Related works

MarCon [1](market-based constraints) aims to support a mix of human and artificial agents by offering a systematic method for applying markets to a wide array of problems.
A MarCon configuration is a network of alternating variables and constraints, each variable and each constraint is a separated agent. The initial network construction must have at least two constraint and one variable, in other definition, it must have buyer, a seller which are agents constraints and a variable which is the environment where calculation happens.
Although the platform uses the notion of agent, but it does not respect the literature of distributed constraint problems in the side as the constraints and variables are separated agents rather than having each agent with a set of variables, domains, constraints.
On the other hand, the converge toward the solution is not always valid because the shrinking of range price. However, if the participants price or assignment



```
started at: 2016-07-14 03:27:42
Providers found are: {X=[F1, F3, F2], Y=[F1, F3, F2], Z=[F2]}
Final price proposed from provider: F2, for product: X: 70
Price proposed from provider: F1, for product: X: 50
Final price proposed from provider: F3, for product: Y: 60
Price proposed from provider: F3, for product: X: 120
Price proposed from provider: F2, for product: Y: 100
Price proposed from provider: F1, for product: Y: 100
Price proposed from provider: F2, for product: Z: 60
All providers have proposed prices: {X={F1=50, F2=70, F3=120}, Y={F1=100, F2=100, F3=60}, Z={F2=60}}
min prices chosen are: {X={F1=50}, Y={F3=60}, Z={F2=60}}
All constraints were tested -> sum=170  b=115
-- can buy all services: false from{X={F1=50}, Y={F3=60}, Z={F2=60}}
Sent nogood to F1:  X:100
Sent nogood to F1:  Y:150
Sent nogood to F2:  Z:30
Final price proposed from provider: F2, for product: Z: 40
Price proposed from provider: F1, for product: Y: 100
Price proposed from provider: F1, for product: X: 50
.......................
.......................
Sent nogood to F2:  Y:150
Price proposed from provider: F2, for product: Y: 50
All providers have proposed prices: {X={F1=25, F2=70, F3=80}, Y={F1=70, F2=50, F3=60}, Z={F2=40}}
min prices chosen are: {X={F1=25}, Y={F2=50}, Z={F2=40}}
All constraints were tested -> sum=115  b=115
-- can buy all services: true from{X={F1=25}, Y={F2=50}, Z={F2=40}}
Stopped at: 2016-07-14 03:27:42
```

Fig. 7. C1 log sample



```
started at: 2016-07-14 03:27:42
Providers found are: {X=[F1, F3, F2], Y=[F1, F3, F2]}
Price proposed from provider: F1, for product: X: 50
Final price proposed from provider: F2, for product: X: 70
Price proposed from provider: F3, for product: X: 120
Final price proposed from provider: F3, for product: Y: 60
Price proposed from provider: F2, for product: Y: 100
Price proposed from provider: F1, for product: Y: 100
All providers have proposed prices: {X={F1=50, F2=70, F3=120}, Y={F1=100, F2=100, F3=60}}
min prices chosen are: {X={F1=50}, Y={F3=60}}
All constraints were tested -> sum=110  b=30
-- can buy all services: false from{X={F1=50}, Y={F3=60}}
Sent nogood to F1:  X:50
Sent nogood to F1:  Y:10
Final price proposed from provider: F1, for product: Y: 70
.......................
.......................
Final price proposed from provider: F3, for product: X: 80
All providers have proposed prices: {X={F1=25, F2=70, F3=80}, Y={F1=70, F2=50, F3=60}}
min prices chosen are: {X={F1=25}, Y={F2=50}}
All constraints were tested -> sum=75  b=30
-- can buy all services: false from{X={F1=25}, Y={F2=50}}
--End of negotiation with final proposed prices:{X={F1=25, F2=70, F3=80}, Y={F1=70, F2=50, F3=60}}
Can't buy all services
Priority is:  X then Y
My budget is: 30 and what i will buy: {X={F1=25}} because the sum is: 25
I will buy : {X={F1=25}}
Stopped at: 2016-07-14 03:27:42
```

Fig. 8. C2 log sample

ranges do not intersect, the variable agent recommends human negotiation between the buyer and seller, so the platform became invalid.

In [10], authors present a DisCSP framework for one to many negotiation by means of conducting a number of concurrent coordinated one to one negotiation, and assume that this framework can be extends to be able to solve the Many-To Many negotiation problem. Also, the framework is based on two negotiation levels , the agent negotiation level in which each agent use the constraint based techniques reasoning, and the coordination level in which the coordinator agent evaluates how well subordinates agent has done, and issues the new instructions. However, negotiations are focused on one product with multi criterias, and the DisCSP formulation is not presented.

Moreover, other several works have been condacted abroad Constraint Programming. The model in [6] performs optimal negotiation process, but it works only in the environment with a fixed number of agents. The model in [7] is a multi-issues negotiations between two agents. However, issues of multi-lateral model still not taken into consideration.

8 Conclusion and future works

Intelligent multi-agent electronic marketplaces are promising, and they will play an essential role in e- commerce and e-business activities. With the growing success of e-marketplace adoption, the need for new intelligent approaches to support both buying and selling goods or services become inevitable. With these approaches, the discovery procedures of products, the issues negotiation, and the solution search must use a real communication challenge.
In this paper, advances of distributed constraint reasoning approaches was operated. We used agent concept to describe the stakeholders to respect all their proprieties. We have proposed a new problem formulation and DCR protocol to deal with e-marketplace issues. More specifically, our attention was focused on constraint-based multi-agent approach offering a dynamic and privacy multilateral negotiation mechanism, namely, ABT-Trader.
Our approach has been implemented and tested using preliminary generated problems. The opening experimental results are promising and further investigations will be conducted. We have just explored the offers destined to satisfy the continuing expectations of market, however we are convinced that all those offers should be treated by our approach. Particularly, fuzzy and dynamic preferences, real case experimentations and learning process.

References

1. Parunak, H. Van Dyke, Allen C. Ward, and John A. Sauter. "A systematic market approach to distributed constraint problems." Multi Agent Systems, 1998. Proceedings. International Conference on. IEEE, 1998.

2. Bessiere, Christian, Arnold MAESTRE, and Pedro MESEGUER. "La famille ABT." Journes nationales sur la rsolution pratique de problmes NP-complets. 2002.
3. Benelallam, Imade, et al. "Dynamic JChoc: A Distributed Constraints Reasoning Platform for Dynamically Changing Environments." International Conference on Agents and Artificial Intelligence. Springer International Publishing, 2015.
4. Bellifemine, Fabio Luigi, Giovanni Caire, and Dominic Greenwood. Developing multi-agent systems with JADE. Vol. 7. John Wiley & Sons, 2007.
5. Benamrane, Amine, et al. "Modeling trainings in Faculty of Medicine and Pharmacy of Casablanca as Constraint Satisfaction Problem."
6. Fatima, S., Wooldridge, M., Jennings, N. "Approximate and Online Multi-Issue Negotiation." In: Proc. of 6th Int. Conf. on Autonomous Agents and Multi-Agent Systems, pp. 947-954 (2007)
7. Lai, G., Sycara, K., Li, C.: A Pareto Optimal Model for Automated Multi-attribute Nego- tiations. In: Proc. of 6th Int. Conf. on Autonomous Agents and Multi-Agent Systems, pp. 10401042 (2007)
8. Eriksson, J., Finne, N. and Janson, S. "To each and everyone an agent: Augmenting web-based commerce with agents", Proceedings of the Inter- national Workshop on Intelligent Agents on the Internet and Web, World Congress on Expert Systems, Mexico City, March 1998, http://www.sics.se/∼market.
9. Hemaissia, M., Seghrouchni, A., Labreuche, C., Mattioli, J.: A Multilateral Multi-Issue Negotiation Protocol. In: Proc. of 6th Int. Conf. on Autonomous Agents and Multiagent Systems, pp. 939-946 (2007)
10. I. Rahwan, R. Kowalczyk and H. H. Pham (2002). Intelligent Agents for Automated One-to-Many E-Commerce Negotiation. Proceedings of the 25th Australasian Computer Science Conference (ACSC), Melbourne, Australia, pp. 197-204.
11. Ren, F., Zhang, M., Sim, K.: Adaptive Conceding Strategies for Automated Trading Agents in Dynamic, Open Markets. Decision Support Systems 46(3), 704-716 (2009)
12. Brito, Ismel, and Pedro Meseguer. "Distributed forward checking."International Conference on Principles and Practice of Constraint Programming. Springer Berlin Heidelberg, 2003.
13. Omomowo, Bayo, Ins Arana, and Hatem Ahriz. "DynABT: Dynamic asynchronous backtracking for dynamic disCSPs." International Conference on Artificial Intelligence: Methodology, Systems, and Applications. Springer Berlin Heidelberg, 2008.