Autonomous Agents and Multi-Agent Systems, 8, 5–45, 2004


# An Extended Multi-Agent Negotiation Protocol

SAMIR AKNINE samir.aknine@lip6.fr
*LIP6, Université Paris 6, 8, rue du Capitaine Scott, 75015 PARIS Cedex 15, France*

SUZANNE PINSON pinson@lamsade.dauphine.fr
*LAMSADE, Université Paris Dauphine, Place du Maréchal De Lattre De Tassigny, 75775 Paris 16, France*

MELVIN F. SHAKUN mshakun@stern.nyu.edu
*Leonard, N. Stern School of Business, New York University, 44, West 4th Street, New York, NY 10012-1126, USA*

**Abstract.** This article presents a task allocation protocol that is efficient in time and tolerates crash failures in multi-agent systems. The protocol is an extension of the negotiation protocol defined by Smith and Davis [25, 26] for task allocation. Our extension of the Contract Net Protocol (1) enables an agent to manage several negotiation processes in parallel; (2) optimizes the length of the negotiation processes among agents; (3) reduces the contractors' decommitment situations; (4) enables the detection of failures of an agent participating in a negotiation process and prevents a negotiation process with blocked agents.

**Keywords:** negotiation protocol, contract net protocol, multi-agent systems.

## 1. Introduction

Negotiation between intelligent agents is one of the fundamental issues of research in Distributed Artificial Intelligence and Multi-Agent Systems. A negotiation process aims at modifying the local plans of each agent in order to achieve agreement among a subset of agents in the system. The well-known Contract Net Protocol (CNP) defined by Smith and Davis [26] for decentralized task allocation is a distributed negotiation model based on the notion of call for bids on markets. The relation between clients (managers, customers, buyers) and suppliers (bidders, contractors, sellers) is created in a call for bids and an evaluation of the proposals submitted by the bidders to the managers. For instance, in the transportation domain, the bid will be given a value which corresponds to the duration of the transportation tasks. The negotiation process is carried out in four steps.

(1) In the first step, the manager sends calls for bids to the agents it knows are able to perform certain tasks;
(2) In the second step, the bidders use the description of the task to build a proposal they send to the manager;
(3) In the third step, the manager receives and evaluates the proposals and assigns the task to the best bidder;
(4) Finally, in the fourth step, the bidder to which the task is assigned sends the manager a message to confirm its intention to do the requested task.

Initially, Smith applied the protocol to simulate the operating of sensors in a distributed acoustic net. In this application [25], the agents of the system are totally cooperative. The selection of an agent for the execution of a task is based on several factors such as the position of the agent in its environment, its capacity to process information, etc.

The CNP model has several limits. First, as a multi-agent system is distributed, several managers can concurrently call for bids, so an agent may have to manage several negotiation processes in parallel in order to reduce the length of its negotiation processes. Some applications of the CNP force the contractors to sequence their negotiation processes, i.e. to answer with a single bid at a time. Sequencing the processing of contractor answers to the calls emitted by the various managers may make the contractors miss some contracts, as the following example shows (cf. section 2.1). Moreover when the multi-agent system is equipped with delay failure detectors, it may also force the managers to consider these contractor agents as failures, which perhaps is not the case. Second, CNP-based applications enable the agent to break its commitments when an agent receives an offer for a better task in comparison with those for which the agent is committed. However, breaking the commitments is not always a good solution because it makes managers call again for bids to find new contractors for their tasks.

In our work, the second original improvement for negotiation protocols is in enabling the detection of failures of agents participating in a negotiation process and in preventing a negotiation process with blocked agents [2–4]. Fault tolerance is a fundamental problem of research in distributed systems. Neither the dynamism of the agents nor the collective behaviors is normally taken into account by the approaches of distributed systems. As in distributed processes, a negotiation for task allocation may fail. An agent (contractor or manager) in a blocked state or in a bad operating state can make the system fail. However, with the growing development of Internet there is a real need for reliable intelligent negotiation tools. The previous solution proposed to this problem of failure at manager level consists in introducing a time limit for the reception of the proposals submitted by the contractors to the managers. Beyond this limit, the manager only evaluates the proposals it has received. Of course, we can re-apply the same method at the level of the contractors to enable them to react to the managers' failures, but it is obvious that this solution leads to many problems. With a short waiting time, the contractor may lose some contracts and with a long waiting time, the contractor may wait unnecessarily for an answer which is probably negative and may therefore lose interesting contracts. To solve this problem, we propose a solution based on a termination process for the negotiation initiated by a contractor which suspects a failure in the manager operation when the time exceeds the answering time of the messages addressed by the contractors. The answering time takes into account the time necessary for messages to be processed and the time necessary to transfer those which have been sent and received. We assume that an approximation of the answering time is known by all the agents before starting their execution in the environment and that the agents follow their protocol.

The protocol proposed is adjusted for cooperative and self-interested agents. It (1) gives a contractor the possibility to run in parallel several negotiation processes; (2) avoids penalizing the contractors when they break their commitments; (3) reduces the length of the global negotiation processes between agents; (4) reduces the situations of decommitment of the contractors; (5) enables the detection of failures of an agent participating in a negotiation process and prevents a negotiation process with blocked agents.

In previous work on the Contract Net Protocol, the effectiveness of this protocol has been evaluated through simulations. However, as effectiveness is a critical aspect of a protocol, we present a formal analysis of our protocol as well as a simulation.

This article is organized as follows. Section 2 informally presents our extension of the CNP. Section 3 formally analyzes our protocol. Section 4 presents our experimental results. Section 5 analyzes related work. Section 6 draws a general conclusion from this work.

## 2. Contract net protocol extension

To enable an agent to manage concurrently several negotiation processes, we have introduced two phases of proposal and allocation by replacing the two phases *Bid* and *Assignment* of the original CNP with four new phases: *PreBid*ding, *DefinitiveBid*ding, *PreAssignment* and *DefinitiveAssignment*. Thus, we obtain a new protocol described in Figure 1 in a simplified form.

The communication primitives used by the managers and the contractors are *Announce*, *PreBid*, *PreAccept*, *PreReject*, *DefinitiveBid*, *DefinitiveAccept* and *DefinitiveReject*. Table 1 proposes the semantics of each operator.

The transitions of the negotiation states between a manager and its contractors are depicted in Figure 2. In our work, we assume that each task is elementary, i.e. it can be performed by a single agent. When a manager announces a task to the contractors (state 1), it receives *PreBid*s, i.e. temporarily bids from the contractors (state 2). These *PreBid*s evaluate the contractors' capacities to carry out the task announced when they receive the announcement. These *PreBid*s are computed according to schedules that the agents make for the tasks that they receive. For each agent, a schedule of tasks forces it to favor the execution of one task compared to another and thus to increase the *PreBid* of the favored task compared to that of the other. The manager sends the best contractor a *PreAccept*

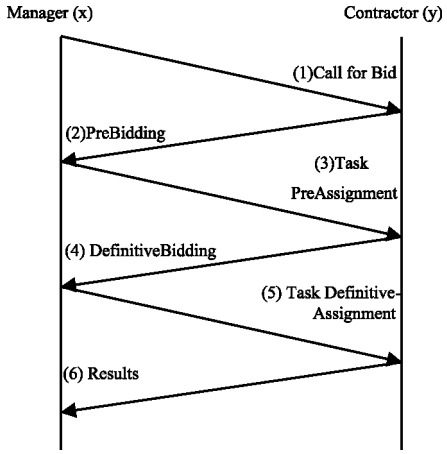



*Figure 1.* Phases of the protocol.

*Table 1.* Semantics of the conversational primitives.

| Primitives | Semantics |
|---|---|
| *Announce* | An agent proposes the execution of a task. |
| *PreBid* | An agent makes a temporarily bid for the execution of a task. |
| *PreAccept* | An agent is temporarily accepted for the execution of a task. |
| *PreReject* | An agent is temporarily rejected for the execution of a task. |
| *DefinitiveBid* | An agent makes a final bid for the execution of a task. |
| *DefinitiveAccept* | An agent is definitively accepted for the execution of a task. |
| *DefinitiveReject* | An agent is definitively rejected for the execution of a task. |

(state 3), i.e. a temporarily positive answer and sends all the other contractors *PreReject* messages (state 4). If the contractors' situations evolve, they can submit new *PreBid*s (state 2). For instance, when an agent is pre-rejected for a task, it will downgrade this task and thus increases the *PreBid* of the other tasks it preceded in the previous schedule.

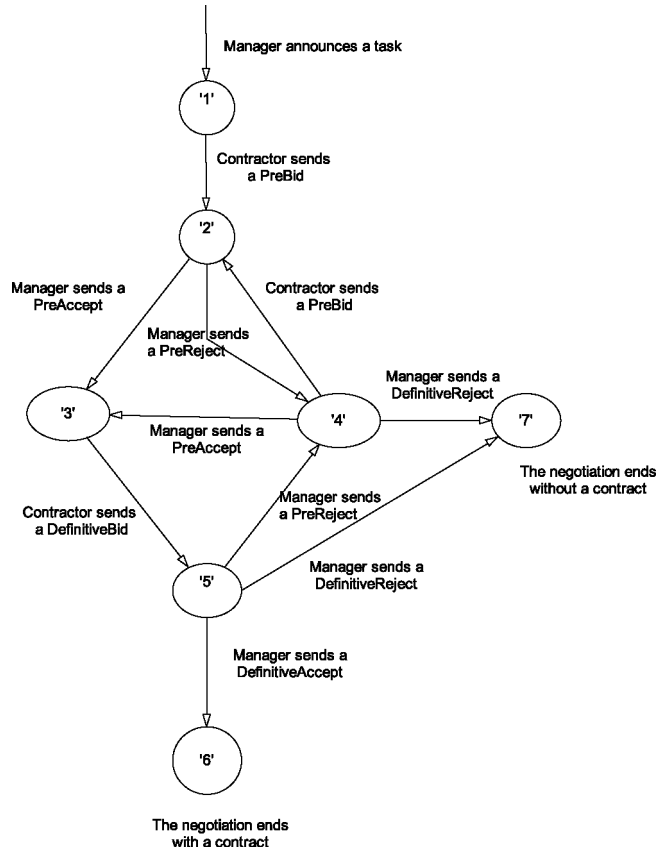



*Figure 2.* State transition graph of a negotiation between a manager and a contractor.

Each time the situation of the contractors changes positively and until the manager of the task sends them a definitive reject concerning the execution of this task, these contractors can bid again but only with better *PreBid*s (state 2, Figure 2).

At this level, the negotiation process between a contractor and a manager can evolve to a *PreAccept* state (state 3) or stay in a *PreReject* state (state 4). On receiving a *PreAccept* message, the potential contractor can send its *DefinitiveBid* (state 5), i.e. its definitive proposal for the task execution. The manager may question this *DefinitiveBid* (states 4 and 7) either if, during the *PreBid*ding phase, a *Pre-rejected* contractor has sent a better *PreBid*, the value of which exceeds the potential contractor's *DefinitiveBid* or if this *DefinitiveBid* is less than the *PreBid* of one of the pre-rejected contractors.

As we assume that the agents can be cooperative or self-interested, we have added a negotiation strategy for self-interested agents in order to prevent an agent from bidding very high in the *PreBid*ding stage in order to scare off competition and then rebidding much lower in the *DefinitiveBid*ding stage. A contractor agent which has been pre-accepted for the execution of a task, and which tries to devaluate its temporarily proposal afterwards will be definitively rejected if there is another agent whose preceding temporarily bid becomes relatively more significant after devaluation. The *DefinitiveBid* sent by the potential contractor can also result in the signing of the contract (state 6) and in a definitive rejection of all other agents (7). The negotiation then ends with the execution of the task by the selected contractor.

Section 2.2 describes in detail the algorithmic specifications of manager and contractor behaviors during a negotiation process.

### *2.1. Interest of two proposal and allocation phases in the protocol*

To make our ideas clearer and highlight the advantages of this protocol, we present here several examples. The first example shows that the protocol ensures an efficient assignment of the tasks to the agents.

**Example 1.** Let us consider a task allocation situation in which two manager agents $M_1$ and $M_2$ want two tasks to be performed rapidly by two contractor agents $C_1$ or $C_2$. Agent $M_1$ would like to propose task $t_1$ whose execution length is 30 mns. Agent $M_2$ wants to propose task $t_2$ whose execution length is 40 mns. Each contractor agent $C_1$ and $C_2$ is currently executing its tasks. Agent $C_1$ will be free in 20 mns, agent $C_2$ in 95 mns.

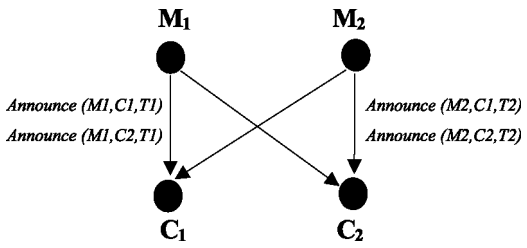



*Figure 3.* Task announcement phase.

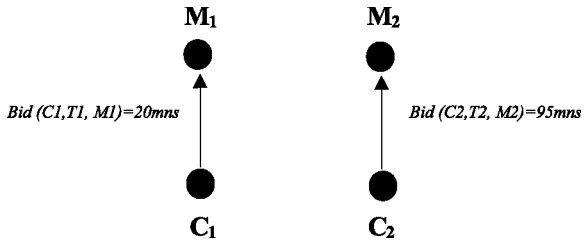



*Figure 4.* Bidding phase.

Negotiation using the CNP will be as follows. Agents $M_1$ and $M_2$ will announce their tasks ($t_1$ and $t_2$) to the agents $C_1$ and $C_2$. Once these agents receive these announcements, they will bid for these tasks in an order which interests them (cf. Figure 3).

Let us assume that agent $C_1$ wants begin task $t_1$ before task $t_2$ and that $C_2$ wants to perform task $t_2$ then task $t_1$. The agents will answer with the following bids. Agent $C_1$ starts a conversational process with $M_1$. It proposes for $t_1$ a bid for 20 mns which corresponds to the time at which agent $C_1$ will be free. Agent $C_2$ sends a bid for 95 mns, time at which $C_2$ will be free concerning $t_2$ to $M_2$. The definition of the CNP does not enable $C_1$ to bid for task $t_2$ until it has received an answer from $M_1$ for the execution of $t_1$. Also, agent $C_2$ cannot be a candidate for executing task $t_1$ before receiving an answer from agent $M_2$. So each contractor agent must sequence its negotiations processes with the two managers (cf. Figure 4).

The two agents $M_1$ and $M_2$ will wait respectively for the bids of $C_2$ and $C_1$ for the execution of the two tasks $t_1$ and $t_2$. If there is a deadline for sending answers, the agents $M_1$ and $M_2$ will answer the bids that they have already received, assuming that agents $C_1$ and $C_2$ respectively are not interested in tasks $t_2$ and $t_1$. So agent $M_1$ will offer its task to $C_1$ and it will cancel its announcement for $C_2$; agent $M_2$ will accept agent $C_2$ and will also cancel its announcement for $C_1$. Otherwise, i.e. if there is no deadline, the multi-agent system will be blocked because the agents will still be waiting for the bids of the contractors (cf. Figure 5).

However, the allocation of the tasks is not efficient because agent $C_1$ will be free in just 50 mns, i.e. 20 mns plus the 30 mns execution time for $t_1$, but task $t_2$ assigned to agent $C_2$ will not be executed before 95 mns are up.

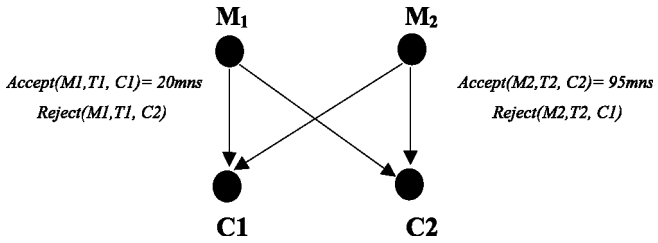



*Figure 5.* Allocation phase.

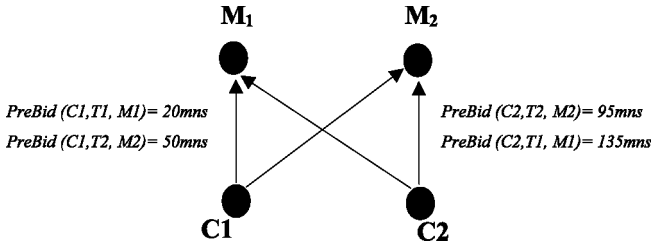



*Figure 6.* Temporarily bedding phase.

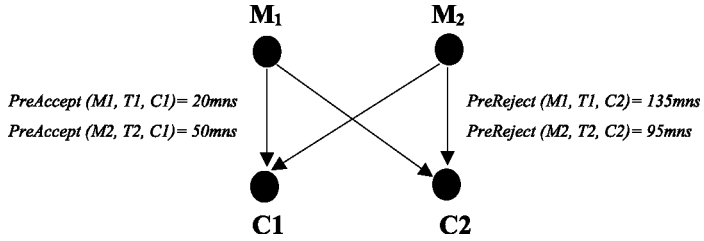



*Figure 7.* Temporarily answering phase.

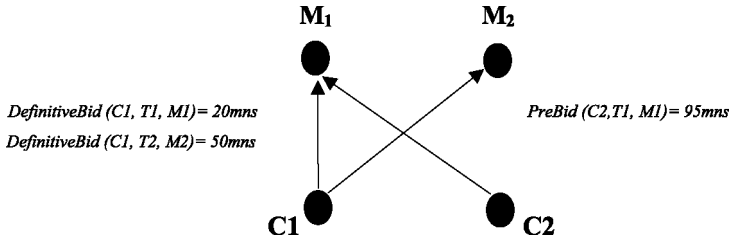



*Figure 8.* Definitive bidding phase.

• *How does the new protocol solve this problem?*

Once the manager agents have announced the tasks, the contractors $C_1$ and $C_2$ order their tasks and submit temporarily bids. Agent $C_1$ sends a *PreBid* of 20 mns to $M_1$ concerning the execution of task $t_1$. At the same time it sends another *PreBid* of 50 mns to manager $M_2$ concerning the execution of task $t_2$. Agent $C_2$ transmits a temporarily bid of 95 mns for task $t_2$ managed by $M_2$ and another temporarily bid of 135 mns for task $t_1$ managed by $M_1$ (cf. Figure 6). Thus each contractor agent can parallelize its negotiation processes with the two managers $M_1$ and $M_2$.

The resulting answers from each manager $M_1$ and $M_2$ are temporarily acceptance of agent $C_1$ for the execution of the tasks $t_1$ and $t_2$, and temporarily rejections of $C_2$ (cf. Figure 7).

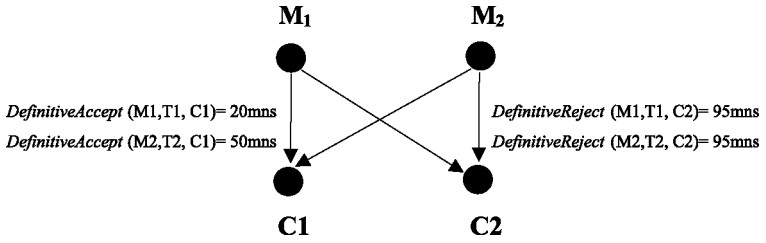



*Figure 9.* Definitive allocation phase.

If the managers can send out the information concerning the *PreBid*s that they have received from the contractors, they do so through *PreReject* messages in which they indicate the *PreBid* of the accepted contractor. If $C_2$ knows this information, it can leave the negotiation process because $C_2$ cannot make a better bid either to $M_1$ or to $M_2$.

Otherwise, if the managers are not authorized to send out the *PreBid*s received, agent $C_2$ will try to reorder its tasks, i.e. it will try to choose the execution of task $t_1$ before the execution of task $t_2$. Agent $C_2$ will send its bid of 95 mns for task $t_1$ to manager $M_1$ again, considering that this bid is better than the one it had previously sent (cf. Figure 8).

This new bid is lower than to that sent by agent $C_1$. Consequently, this message will be followed by a negative definitive answer for $C_2$ and by a positive definitive answer for $C_1$ as soon as $C_1$ confirms its temporarily bid by sending a *DefinitiveBid* which is equal to its temporarily bid in this example. Contrary to the results observed when applying the Contract Net Protocol, the negotiation between agents with this new protocol gives a more efficient allocation of the tasks (cf. Figure 9).Using the Contract Net Protocol, the execution of the two tasks would take 135 mns and using the suggested protocol, the execution of the tasks would take 90 mns.

Over and above the advantage of efficient allocation of tasks, there are different reasons for using two phases of proposal and allocation to negotiate the execution of tasks. These advantages are the following.

(1) It enables M-N negotiations, i.e. a contractor agent can manage concurrently several negotiation processes with M manager agents and a manager agent can manage concurrently several negotiation processes with N contractor agents, thus reducing the global length of the negotiation between the agents. Sending the managers (resp. contractors) proposal (resp. answer) messages in packages means messages can be processed in parallel, which reduces the waiting phase of the senders and the receivers.
(2) An agent can propose itself with a temporarily bid (*PreBid*) to perform a task as soon as it receives an order. If the agent subsequently receives new orders, it can modify its previous offer as long as it has not sent a *DefinitiveBid*.
(3) The length of the *PreBid*ding phase enables an agent to make the best choice from among the tasks proposed before bidding definitively for the tasks, so that once acceptance to carry out the tasks has been given, the protocol reduces the risk of decommitment.

(4) The fact that the contractors inform the managers of the other contracts they are negotiating at the same time enables the managers to control the situations better. Until the manager has received a *DefinitiveBid* from the potential contractor confirming its intention to perform the task both parties agreed on, it does not definitively reject the other contractors but informs them of the existence of this potential contractor.
(5) Reducing the contractors' decommitment means the manager may not have to announce the task again when a contractor rejects its contract and, until the contract has been definitively signed, will wait for better *PreBid*s from the pre-rejected contractors.

### *2.2. Negotiation algorithms*

This section presents the algorithms which describe the behaviors of the managers and the contractors which participate in a negotiation process as well as the methods of ending a negotiation process between agents when there is a failure within the system. An agent is suspected of being in a failure state if it is not able to answer the messages which are sent to it. Failures in an agent are detected using a supervisor system introduced in each agent. This system informs the other agents about these failures.

The details of the contractor behaviors are given in Figure 10.

1. Let $C_i$ be a contractor agent which receives an offer for the execution of a task $T_k$, $C_i$ sorts task $T_k$ in its list of tasks. It computes, saves and sends the *PreBid* of $T_k$.
2. When contractor $C_i$ receives a *PreReject* message from manager $M_j$ of task $T_k$, $C_i$ sorts its task $T_k$. After this new ordering of its tasks, $C_i$ computes, saves and sends the *PreBid*s and *DefinitiveBid*s which have changed for all the tasks. $C_i$ waits for a definitive answer from manager $M_j$.
3. When $C_i$ receives a *DefinitiveReject* message, if its plan has changed while including task $T_k$, $C_i$ restores its previous plan without $T_k$. It computes, saves and sends the *PreBid*s and *DefinitiveBid*s of all the tasks after this new ordering of its tasks. $C_i$ deactivates the timeout of task $T_k$.
4. When $C_i$ receives a *PreAccept* message, $C_i$ computes, saves and sends the *DefinitiveBid* of task $T_k$.
5. When $C_i$ receives a *DefinitiveAccept* message, $C_i$ may start the execution of its task.
6. When $C_i$ receives a new announcement for the execution of another task $T_{k'}$, $C_i$ sorts task $T_{k'}$ in its list of tasks. It computes, saves and sends the *PreBid* of $T_{k'}$ and the *PreBid*s or *DefinitiveBid*s which have changed for all the tasks after this new ordering of its tasks. $C_i$ waits for an answer.
7. If manager $M_j$ is suspected of failure, $C_i$ initiates a negotiation termination process for task $T_k$.

*Figure 10.* Contractor behaviors during a negotiation process.

The detail of the manager behaviors are given in Figure 11.

1. For a manager agent $M_j$ which offers one of its tasks $T_k$ to other agents, $M_j$ sends announcement messages to obtain bids from the contractors $C_i$. $M_j$ waits for the bids from contractors that are not suspected of failure.
2. When manager $M_j$ receives a *PreBid* message from a contractor $C_i$ during the *PreBid*ding phase, $M_j$ looks in its contractor list for a potential contractor whose *PreBid* for task $T_k$ is greater than that of $C_i$. If this is the case, $M_j$ answers with a *PreReject* message to contractor $C_i$. If this is not the case, $M_j$ considers contractor $C_i$ as a potential one for the execution of task $T_k$. If all the contractors have sent their bids, manager $M_j$ sends the potential contractor a *PreAccept* message.
3. When manager $M_j$ receives a *PreBid* message from a contractor $C_i$ during the DefinitiveAssignment phase, $M_j$ compares this *PreBid* with the *PreBid* of the potential contractor; if it is not better, $M_j$ *PreReject*s $C_i$. If the *PreBid* is better, $M_j$ *PreReject*s the potential contractor and sends contractor $C_i$ a *PreAccept* message.
4. When manager $M_j$ receives a *DefinitiveBid* message for the execution of task $T_k$ and if this *DefinitiveBid* is greater than the *PreBid* of the potential contractor, or all the *PreBid*s are less than the *DefinitiveBid*, manager $M_j$ answers with a *DefinitiveAccept* message to the potential contractor and sends all the other contractors $C_i$ *DefinitiveReject* messages. In the other cases, $M_j$ selects another potential contractor and sends it a *PreAccept* message if this has not already been done and consequently sends the old potential contractor a *PreReject* message.
5. If $M_j$ suspects that a contractor has failed, $M_j$ considers this contractor which has not answered the message as uninterested in task $T_k$.

*Figure 11.* Manager behaviors during a negotiation process in a multi-agent system.

Our method, which consists in solving a blockage in the case of multi-agent systems, is illustrated in Figure 12. This method is adopted by each contractor which suspects a blockage in the manager of the negotiation process. It uses two variables: Manager_Decision and Contractor_Decision.

The first variable enables a contractor to inform other agents about the last answer *PreReject*, *PreAccept*, *DefinitiveReject* or *DefinitiveAccept* received from the manager about its refusal or acceptance to perform the announced task. The second variable indicates locally (for each contractor) the decision of the manager, which is deduced by the contractor for the attribution of the task. If the manager fails, unanimous decisions enable the agents of the system to unblock the negotiation process. Ending the negotiation process saves the contractors from waiting uselessly for an answer which is probably negative and therefore from losing interesting contracts.

In the first phase of this procedure, the contractor sends all the agents an Inform Manager_Decision message (lines 1 and 2). By default, its variable contains an "Unknown" value which indicates that the contractor is waiting for a temporarily or a

```
Procedure initiation of a termination process for negotiation in a multi-agent system
Begin
                    /*Let C be the set of contractors of manager Mj */
(1)     For each aj ∈ C
(2)             Send Message Manager_Decision for task Tk
(3)     Wait (      (a delivery of a PreReject (Tk) from each aj ∈ C)
(4)             or (a delivery of a DefinitiveReject (Tk) from each aj ∈ C)
(5)             or (a delivery of a PreAccept from a contractor)
(6)             or (a delivery of a DefinitiveAccept from a contractor))
(7)             or (a contractor is suspected));
(8)     Case
(9)             A PreAccept has been delivered from a contractor
(10)                    Contractor_Decision ← PreReject (Tk)
(11)            A DefinitiveAccept has been delivered from a contractor
(12)                    Contractor_Decision ← DefinitiveReject (Tk)
(13)            All answers are PreRejects (Tk)
(14)                    Contractor_Decision ← PreAccept(Tk)
(15)            All answers are DefinitiveRejects (Tk)
(16)                    Contractor_Decision ← DefinitiveAccept(Tk)
(17)            ∃ aj ∈ C, Suspected(aj) and ∃ ap ∈ C, ap ≠ aj / Answer (ap) ∈ {DefinitiveReject}
(18)                    Contractor_Decision ← DefinitiveReject (Tk)
(19)            ∃ aj ∈ C, Suspected(aj) and ∀ ap ∈ C, ap ≠ aj / Answer (ap) ∉ {DefinitiveReject}
(20)                    Contractor_Decision ← PreReject (Tk)
(21)    EndCase
(22)    For each aj ∈ C
(23)            Send Message Contractor_Decision for task Tk
End
```

*Figure 12.* Algorithm for ending a multi-agent negotiation process.

definitive answer from the manager. This message reminds the manager and the contractors about the last manager decision concerning the allocation of a task.

In the second phase, the contractor enters a waiting phase until (1) it receives a *PreAccept* (resp. *DefinitiveAccept*) message from the agents (lines 9 and 12) to infer the *PreReject* (resp. *DefinitiveReject*) decision of the manager; (2) it receives a *PreReject* or a *DefinitiveReject* message from all the agents (lines 13 and 16) to infer the *PreAccept* (resp. *DefinitiveAccept*) decision; (3) the agent suspects a failure in another agent of the system and leaves the negotiation process (lines 17 and 20). In the third phase, the contractor sends all the agents its final decision (lines 22 and 23).

This method of unblocking a negotiation process by the contractors is significant because it always unblocks a contractor agent that takes the decision to execute or not the announced task. If the agent infers a manager's decision of the type *DefinitiveAccept* or *PreAccept* concerning this task, it can start its execution if during the previous negotiation steps it has received all the information concerning this task. If it infers a manager's decision of the type *DefinitiveReject*, it leaves the negotiation process. An inferred *PreReject* decision for a task enables the contractor to reorder its list of tasks and to outclass this task by the other tasks appearing in its plan. So in all the cases, the multi-agent system is necessarily unblocked with this method. It is useful because it enables the agents to solve autonomously the problem of failures.

## 3. Theoretical analysis of the negotiation protocol

Our protocol aims at minimizing the length of the negotiation processes and at taking into account faulty cases. At this stage of the presentation of our negotiation protocol, we have informally defined our protocol. We have introduced the algorithms and communication primitives which enable the synchronization of conversational processes between the agents which participate in the negotiation process. In the following section, we will introduce the formal definitions we have used in our negotiation protocol. Following [14], we define a negotiation with a manager and a contractor as follows:

**Definition 1** (*Bipartite negotiation*) Let A be a set of agents in a multi-agent system S, T be a set of tasks announced by the agents of S. A bipartite negotiation process between a manager agent $M$ and a contractor $a_i$ for the execution of a task $T_k$ is a 6-uple:

$$\mathrm{Neg}_{i,k} = (a_i, M, T_k, P, s, v)$$

- $a_i \in A$, where $a_i$ is a contractor agent engaged by M in the process $\mathrm{Neg}_{i,k}$.
- $M \in A$, where M is a manager agent responsible of the negotiation process $\mathrm{Neg}_{i,k}$.
- $T_k \in T$, where $T_k$ is a task for which the negotiation process $\mathrm{Neg}_{i,k}$ has been started.
- P is a set of necessary communication primitives for M and $a_i$ to negotiate the execution of $T_k$, where $P = \{$*PreBid*, *PreAccept*, *PreReject*, *DefinitiveBid*, *DefinitiveAccept*, *DefinitiveReject*$\}$.
- $s \in P$, where s is the current state of the negotiation $\mathrm{Neg}_{i,k}$ between $a_i$ and M.
- v is the current value proposed by $a_i$ to M in order to execute task $T_k$.

### 3.1. *Termination of the negotiation process*

In the previous section, we have shown that the negotiation protocol between the manager and its contractors specifies the actions which can be followed by each member to negotiate a contract about the execution of a task. However, the protocol does not enable the contractor to build a deterministic plan ending with the signing of the contract by the manager. At each negotiation step, the contractor must make a choice from among several possibilities to keep the negotiation process going. We have defined in the contractors' behaviors a strategy that rearranges the tasks which caused the rejection of the agent. In our approach, the decision about the adoption of a behavior by any contractor $a_i$ during a negotiation process $\mathrm{Neg}_{i,k}$ with a manager M concerning the execution of a task $T_k$ does not only depend on the manager's behaviors but also on the states of the other negotiation processes $\mathrm{Neg}_{i,p}$, started concurrently by $a_i$ about the execution of tasks $T_{p/p\neq k}$.

The basic idea that we have adopted in order to guarantee the ending of a negotiation process is to enable a contractor to change its preference order of tasks announced if some of its contracts are not signed. Therefore, it withdraws from the tasks that it has favored in the preceding cycles and from which it has been rejected, and it improves its offers for the tasks it had neglected in the preceding cycles. The following definitions, lemmas and theorems show these properties of the protocol, in particular the termination property of

the negotiation processes between agents in all possible cases, i.e. even if a failure occurs in one or more agents in the system.

When an agent receives several offers at the same time from one or more managers, it sorts these offers. This shows the need to define a task-ordering model.

**Definition 2** (*Task order*) Let A be a set of 'n' agents, $a_i$ an agent of A and T a set of 'm' tasks announced to $a_i$ by the agents of A. Let WantExec ($a_i$, $T_k$, $\delta_k$) be a predicate which is true if $a_i$ wants to perform the task announced $T_k \in T$ with a bid $\delta_k$ and T-ord$_i$($T_k$) the order of task $T_k$ in the list of tasks of $a_i$. Tasks are ordered by $a_i$, denoted:

$$\text{T-Ord}_i(T) = (T_1, .., T_r)$$

such that:

$$1 \leq l, r \leq m, 1 \leq r, \delta_1 > \cdot\cdot > \delta_r, \text{T-ord}_i(T_1) < \cdot\cdot < \text{T-ord}(T_r).$$

When a manager receives several offers submitted by different contractors for the execution of several tasks it has announced, it sorts the contractors in order to evaluate them.

**Definition 3** (*Agent order*) Let A be a set of 'n' agents and T a set of tasks. Let $T_k$ be a task of T announced by an agent $a_l$ for the agents $a_i$. Let WantExec($a_i$, $T_k$, $\delta_i$) as defined above and A-ord$_k$($a_p$) the order of agent $a_p$ in the list of possible contractors for the execution of task $T_k$. An order of the agents of A which bid for the execution of $T_k$ is denoted:

$$\text{A-Ord}_k(A) = (.., ap, .., aq, ..)$$

such that:

$$1 \leq p, q \leq n, \delta_p \geq \cdot\cdot \geq \delta_q, \text{A-ord}_k(a_p) < \cdot\cdot < \text{A-ord}_k(a_q)$$

The protocol proposed satisfies certain properties, in particular in the sending of the answers provided to the contractors during a negotiation process. The following lemma and theorem show that the definition of the protocol takes into account all the answers sent by each contractor agent to the manager of the tasks.

**Lemma 1.** Under the hypothesis of instantaneous message delivery,

1. A potential contractor cannot reach the *DefinitiveBid*ding (resp. Task Execution) phase until all the contractors have received a *PreReject* (resp. *DefinitiveReject*) message from the manager of the negotiation process, in order to give the rejected contractors the possibility of modifying their bids.

2. A contractor cannot reach a *PreReject* state until one of the contractors is in a *PreAccept* state, and it cannot access a *DefinitiveReject* state until one of the contractors is in a *DefinitiveAccept* state.

**Proof:**

- In order to demonstrate (1), we know that an agent enters a *DefinitiveBid*ding (resp. Task Execution) phase only if it has previously received a *PreAccept* (resp. *DefinitiveAccept*) message from the manager. With this argument and with the hypothesis of instantaneous message sending, all the contractors have necessarily already received a *PreReject* (resp. *DefinitiveReject*) message (cf. Figure 11). So they have already been through the *PreReject* (resp. *DefinitiveReject*) phase.
- For (2), as for (1), to be in a *PreReject* (resp. *DefinitiveReject*) state, a contractor must previously have received a *PreReject* (resp. *DefinitiveReject*) message, which means the existence of a potential contractor whose *PreBid* (resp. *DefinitiveBid*) is at least equal to the *PreBid* of the pre-rejected (resp. definitively rejected) contractor. For this contractor, the manager will necessarily send a *PreAccept* (resp. *DefinitiveAccept*) message. Then, this potential contractor will be in a *PreAccept* (resp. *DefinitiveAccept*) state (cf. Figure 13).

**Theorem 1** The algorithm describing the behaviors of the manager in the negotiation process (cf. Figure 11) always respects conditions (1) and (2) of lemma 1.

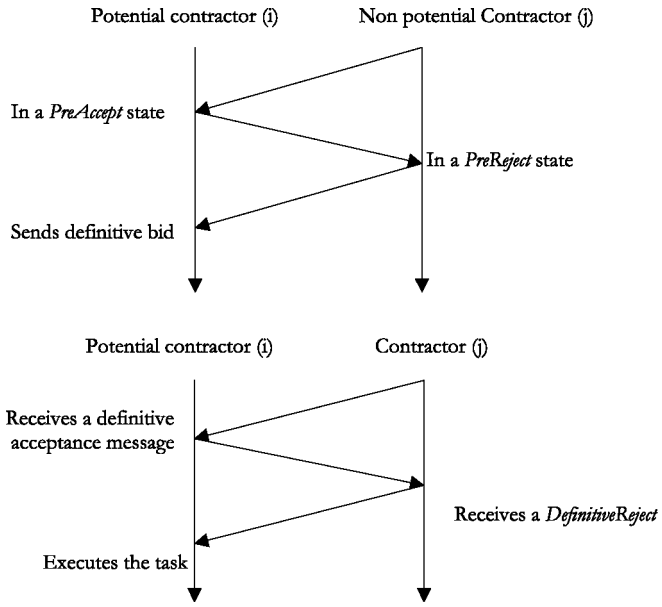



*Figure 13.* Comparison between the transition states of a negotiation with a potential contractor and with a non-potential contractor.

In all cases, condition (2) of lemma 1 cannot be violated by the algorithm that describes the manager's behavior during the negotiating process. The reason is that the manager sends a *PreReject* (resp. *DefinitiveReject*) message to a contractor ($a_j$) only after verifying the condition (*PreBid* ($a_j$) $\leq$ Max*PreBid*s) (resp., Max*PreBid*s $\leq$ *DefinitiveBid* ($a_p$)), (see Figure 11), $a_p$ being a potential contractor.

This condition requires that at least one potential contractor exists and has a *PreBid* which equals the maximum value of the *PreBid*s. Condition (1) of lemma 1 confirms that the manager can only start the second phase of the algorithm if the first phase has been completed.

The dependence between the tasks announced to an agent is translated into a dependence between the negotiation processes engaged by this agent for the execution of these tasks.

As the tasks of each agent are ordered, the bids computed for some tasks will depend on the bid computed for the others. Consequently, a temporarily or definitive reject during the negotiation of a task will involve changes in the negotiation process of the tasks which depend on it. The following definition shows the conditions of dependence of negotiation processes.

**Definition 4** (*Dependence of negotiation processes*) Let A be a set of 'n' agents and T be a set of 'm' tasks announced by the managers of A. Let $Neg_i$ be a set of negotiation processes started by $a_i$. A negotiation process $Neg_{i,k} \in Neg_i$, with i = 1..n and k = 1..m, such that $Neg_{i,k} = (a_i, M, T_k, P, s, \upsilon)$ between a contractor $a_i \in A$ and a manager $M \in A$, is dependent on a negotiation process $Neg_{i,k'} = (a_i, M', T_{k'}, P, s', \upsilon')$ between a contractor $a_i \in A$ and a manager $M' \in A$, if the order of the tasks $T_j/j = 1..m$ for the contractor $a_i$ such that:

$$T\text{-}Ord_i(T) = (.., T_{k'}, .., T_k, ..).$$

The negotiation process $Neg_{i,k'}$ is not dependent on the negotiation process $Neg_{i,k}$.This definition means that an agent which receives several offers from the managers should sort the announced tasks according to its preferences for these tasks and also according to tasks it has committed itself to perform. While ordering these tasks, the agent makes priorities between them. Consequently, the bid that it submits for a task depends on the bids that it has submitted for the tasks for which it has already committed itself and the bids for the tasks it prefers in this ordering relation.

Now we define the negotiation processes that can be positively reordered, i.e. processes which can enable the agreement of different participants to the negotiation process when the contractor agent is in a temporary reject step and the manager agent has a potential contractor to which it will allocate the task if the pre-rejected contractors do not react.

**Definition 5** (*Positively reordered bipartite negotiation*) Let A, m and $Neg_i$ be as defined above. A negotiation $Neg_{i,k} \in Neg_i$, with i = 1..n and k = 1..m, such that $Neg_{i,k} = (a_i, M, T_k, P, s, \upsilon)$ between a contractor $a_i \in A$ and a manager $M \in A$ can be positively reordered by $a_i$ if there is at least a negotiation $Neg_{i,k'} \in Neg_i$,

$$Neg_{i,k'} = (a_i, M', T_{k'}, P, s', \upsilon')$$

such that:

$$M' \in A, k' \neq k, s' \in \{Pre\text{R}eject, Definitive\text{R}eject\}$$

and $\text{T-Ord}_i(T) = (.., T_{k'}, .., T_k, ..)$.

The following lemma demonstrates that a negotiation process between agents may be arranged when this negotiation fails to find an agreement but it cannot be arranged definitively. This serves to prove the next theorem.

**Lemma 2.** Let T be a set of 'm' tasks announced to an agent $a_i$ and $\text{T-Ord}_i(T)$ the preference order of $a_i$ for the execution of the tasks T. Let $\text{Neg}_i$ be a set of negotiations started by $a_i$ for the execution of tasks $T_j$, $T_j \in T$ with $j = 1..m$. Let p be the rank of a task $T_k$ in $\text{T-Ord}_i(T)$ of $a_i$ and $\text{Neg}_{i,k}$ be the negotiation process of $a_i$ for the execution of $T_k$. Agent $a_i$ may at most $(p - 1)$ times positively rearrange the negotiation process

$$\text{Neg}_{i,k} = (a_i, M, T_k, P, s, v).$$

**Proof:** (see annexes)

The following theorem proves that the agent negotiation processes do not fail using this protocol.

**Theorem 2** Let S be a multi-agent system composed of n agents and T a set of tasks announced. A negotiation process engaged by the agents of S using the protocol proposed ends after a finite set of steps.

**Proof:** (see annexes)

Having presented the properties satisfied by the protocol proposed, in particular those which ensure an effective negotiation with all the potential contractors for a given task, we will continue our study of the protocol by analyzing the other properties it checks. The properties that interest us concern how contractor agents unblock a negotiation process.

### *3.2. Unblocking contractor and manager negotiation processes in failure cases*

In the previous sections we have presented the multi-agent negotiation protocol we propose for task allocation. This protocol is applied by agents which can take the role either of manager or of contractor. However, these agents can break down and the negotiation may block. A classical solution to solve this problem of breakdown in manager agents consists in introducing a time limit for the reception of the bids submitted by the contractors to the managers. Beyond this limit, the manager evaluates the bids it has received. Of course, we can re-apply the same method at the level of the contractors to enable them to react to the managers' failures, but it is obvious that this solution leads to many problems. With a short waiting time, the contractor may lose some contracts and with a long waiting time, the contractor may wait unnecessarily for an answer which is probably negative and may therefore lose interesting contracts.

To solve this problem we propose a solution which is inspired by work suggested in [10, 15, 19]. This solution is based on a termination process for the negotiation initiated by a contractor which suspects a failure in the manager operation when the time exceeds the answering time of the messages sent by the contractors. The answering time takes into account the time necessary for messages to be processed and the time necessary to transfer those which have been sent and received. This process will be free by exchanging messages between contractor agents which are assumed to have correctly received the messages. We assume that the messages sent by the agents are not corrupted after the failure. The problem of task allocation by the managers is thus transformed into a process of unblocking a negotiation process by the contractor agents. During this process, all those contractor agents that are not blocked propose a value (cf. section 2). Using these values, the contractor agents solve the failure. The process of unblocking a failure satisfies the following properties:

**Termination.** Each contractor agent that is not blocked decides on a value from among the set of possible answers that the manager can send it concerning the execution of a task. Formally:
Let A be a set of agents in a multi-agent system, Non-Blocked(A) an operator which returns the set of non-blocked agents in A and $T_k$ a task in current negotiation.

$$\forall a_i \in \text{Non-Blocked}(A),$$

$$\exists\ c \in \{\textit{PreAccept}, \textit{Pre}\text{Reject}, \textit{DefinitiveAccept}, \textit{Definitive}\text{Reject}\}$$

such that:

$$\text{Decision}\ (a_i, T_k) = c$$

**Agreement.** Two agents cannot give the same value of definitive acceptance for the execution of a task $T_k$.
Formally,

$$\forall a_i, a_j \in \text{Non-Blocked}(A),\ \text{if Decision}(a_i, T_k) = \text{Decision}(a_j, T_k) =$$

$$\textit{DefinitiveAccept}\ \Rightarrow a_i = a_j$$

**Integrity.** Each decision of an agent belongs to the set of possible answers, i.e. {*PreAccept*, *Pre*Reject, *DefinitiveAccept*, *Definitive*Reject}.

**Validity.** If a contractor agent $a_i$ decides with a value $v \in \{\textit{PreAccept}, \textit{DefinitiveAccept}\}$, then:

- a priori, $a_i$ has received the answer v from the manager of the task.

or

- $a_i$ has received an answer $v' \notin \{\textit{PreAccept}, \textit{DefinitiveAccept}\}$ from $\forall a_j \in A$, $i \neq j$.

Now we will prove that the results of the algorithm which unblocks a negotiation process generate consistent states for all the agents. To demonstrate the following theorems, we need to demonstrate some intermediate results.

Lemma 3 guarantees that each non-blocked agent of the system will receive answers which indicate the negotiation state of each contractor agent for the current task.

**Lemma 3.** At each cycle of the first phase of the algorithm for unblocking contractor agents (cf. Figure 12), each contractor agent $a_i$ receives a message of the type Decision_Manager ($a_p$, M, $T_k$) from each contractor $a_p$ in A. This message indicates the last answer provided by the manager for contractor $a_p$ on the execution of task $T_k$.

**Proof:** Since agent $a_i$ belongs to the set of possible contractors for the execution of a task $T_k$ and since any agent $a_p$ carries out all the cycles of the first phase of the algorithm for releasing contractor agents (cf. Figure 4), at each cycle an agent $a_i$ waits for and receives a message of type Decision_Manager ($a_p$, M, $T_k$) from agent $a_p$.

The following lemma demonstrates that the algorithm proposed avoids all possible conflict on the execution of a task by different contractor agents.

**Lemma 4.** Let $a_i$ and $a_{i'}$ be two contractor agents for the execution of a task $T_k$. $a_i$ and $a_{i'}$ cannot give the same value *DefinitiveAccept*($a_i$, $T_k$) and *DefinitiveAccept*($a_{i'}$, $T_k$) for the execution of $T_k$.

**Proof:** We must show now that at the end of the second phase of the algorithm for releasing the contractor agents, if *DefinitiveAccept* ($a_i$, $T_k$) $\in$ Decision ($a_i$) then *Definitive-Accept* ($a_{i'}$, $T_k$) $\notin$ Decision ($a_{i'}$) and inversely. Two cases must be considered:

- if *DefinitiveAccept* ($a_{i'}$, $T_k$) $\in$ Decision_Manager ($T_k$) at the end of phase 1 of the algorithm (cf. Figure 12), according to lines 9 to 12, Decision ($a_i$) $\in$ {*Definitive*R*eject*, *Pre*R*eject*};
- if Answer ($a_{i'}$, $T_k$) = {}, $a_i$ waits and then decides with *Definitive*R*eject* ($a_i$, $T_k$) or *Pre*R*eject* ($a_i$, $T_k$) (lines 17 to 20).

Lemma 5 shows that the method of a contractor agent for unblocking a negotiation process terminates.

**Lemma 5.** If a contractor agent is not in a failure state, it necessarily reaches the third phase of local decision in the algorithm for unblocking blocked agents (cf. Figure 12).

**Proof:** The only case where a contractor agent $a_i$ which is not in a failure state does not reach the third phase of local decision in the algorithm for releasing contractor agents (cf. Figure 12) is when the contractor waits indefinitely in the second phase. Indeed during the second phase, $a_i$ may wait for a message of an agent $a_{i'}$ it has not suspected; two cases must then be considered:

- Agent $a_{i'}$ fails. In this case, the deadline for answers from the contractors enables agent $a_i$ to suspect agent $a_{i'}$;
- Agent $a_{i'}$ has not failed. $a_i$ is not waiting for a message from agent $a_{i'}$ in the second phase because $a_{i'}$ has necessarily sent it in the first phase.

In both cases, $a_i$ is not blocked in the second phase of the algorithm for releasing contractor agents (cf. Figure 12); thus $a_i$ will necessarily reach the third phase of this algorithm.

Theorems 3 and 4 show that the results of the protocol proposed remain consistent even with certain failed agents.

**Theorem 3** Let S be a multi-agent system composed of n agents. Let $a_m$ be a contractor agent in S for a task $T_l$ announced by a manager M. When M fails, if the contractor $a_m$ sends a message of the type "Request_Inform" to the other contractors of M in order to be informed of the state of their negotiation processes with M for the execution of task $T_l$, the contractor $a_m$:

1. Cannot receive from two agents $a_i$ and $a_{j/i\neq j}$ two messages of the type *DefinitiveAccept*($a_i$, $T_l$) (resp., *PreAccept*($a_i$, $T_l$)) and *DefinitiveAccept*($a_j$, $T_l$) (resp., *PreAccept*($a_j$, $T_l$)).
2. Cannot receive from a contractor agent $a_i$ a message of the type *Definitive-Accept*($a_i$, $T_l$) (resp., *PreAccept*($a_i$, $T_l$)) if the manager has sent a message of the type *DefinitiveAccept*($a_m$, $T_l$), $i \neq m$ (resp., *PreAccept*($a_m$, $T_l$)).
3. Cannot receive from all the contractors $a_{i/i=1..n,m\neq i}$ messages of the type *Definitive-Reject*($a_i$, $T_l$) (resp., *Pre*R*eject*($a_i$, $T_l$)) if the manager has sent a message of the type *Definitive*R*eject*($a_m$, $T_l$) (resp., *Pre*R*eject*($a_m$, $T_l$)).

**Proof:**

1. A contractor agent $a_m$ cannot receive from the manager a message of the type *Definitive*R*eject*($a_m$, $T_l$) (resp., *Pre*R*eject*($a_m$, $T_l$)) unless there is a potential contractor $a_{p/p\neq m}$ for which the manager has necessarily sent a message of the type *DefinitiveAccept*($a_p$, $T_l$) (resp., *PreAccept*($a_p$, $T_l$)). This message is necessarily followed by other messages of the type *Definitive*R*eject*($a_i$, $T_l$)) (resp., *Pre*R*eject*($a_i$, $T_l$)) for each agent $a_{i/i\neq p}$, $a_m$ included, which demonstrates part (2) of this theorem.
2. A contractor agent $a_m$ may receive from the manager a message of the type *DefinitiveAccept*($a_m$, $T_l$) (resp., *PreAccept*($a_m$, $T_l$)) only if this agent is the potential contractor for task $T_l$. This message is necessarily preceded by messages of the type *Definitive*R*eject*($a_i$, $T_l$) (resp., *Pre*R*eject*($a_i$, $T_l$)) for each agent $a_{j/i\neq m}$, which means that the manager cannot send a message of the type *Definitive*R*eject*($a_m$, $T_l$) (resp., *Pre*R*eject*($a_m$, $T_l$)) to the contractor $a_m$, which demonstrates part (3) of this theorem.
3. In (1), we have shown that a contractor agent $a_m$ does not receive from the manager a message of the type *Definitive*R*eject*($a_m$, $T_l$) (resp., *Pre*R*eject*($a_m$, $T_l$)) unless there is one and only one potential contractor $a_{p/p\neq m}$ having received a message of the type *DefinitiveAccept*($a_p$, $T_l$) (resp., *PreAccept*($a_p$, $T_l$)). This message, sent by the manager, is followed by other messages of the type *Definitive*R*eject*($a_m$, $T_l$) (resp., *Pre*R*eject*($a_m$, $T_l$)) for each agent $a_{m/m\neq p}$, which justifies the fact that $a_m$ cannot receive at the same time from two different agents $a_i$ and $a_{j/i\neq j}$ two messages of the type *DefinitiveAccept*($a_i$, $T_l$) (resp., *PreAccept*($a_i$, $T_l$)) and *DefinitiveAccept*($a_j$, $T_l$) (resp., *PreAccept*($a_j$, $T_l$)), which demonstrates part (1).

**Theorem 4** The algorithm describing the manager's behaviors during the negotiation process (cf. Figure 11) generates consistent states for the system, in that two contractors can be temporarily accepted at the same time, but not definitively.

**Proof:** Part (1) of theorem 3 implies that the execution of the task is initiated by a potential contractor once all the other contractors have been definitively rejected, i.e. the negotiation process cannot finish with two contractors accepted definitively at the same time. However, when a contractor fails at the end of the PreAssignment phase, it can be in a *PreAccept* state after a blocking phase during which the manager should have selected another potential contractor agent for the execution of the task.

In this section, we have presented and formally analyzed the properties of the negotiation protocol proposed. In the following section, we will show how to use our protocol on a real multi-agent application.

## 4. Experimental results

We have chosen to illustrate our negotiation protocol on a transportation application. Our system is composed of two types of agents, customers and suppliers. A customer (manufacturer) produces the goods on its site (factory) and stores them in several warehouses. The goal of the customer is to deliver the goods produced in the shortest time. To achieve this goal, the customer must find a supplier (delivery agent) who undertakes the carriage of these goods from the site of production to the warehouses.
A supplier is equipped with a geographical map which indicates sites on which the customers are positioned, the site of their warehouses and the possible transportation communication channels. Each supplier owns a truck.
The customers can communicate with their suppliers and call them when they produce new goods on their sites. Each supplier must thus build its own plan for the tasks it will ask to be carried out. Its goal is to maximize its payoff by transporting a maximum amount of goods in a minimum amount of time. To understand how these negotiation algorithms run, let us consider another example.

**Example 2.** In this case, we consider two suppliers $S_1$ and $S_2$ which are initially at positions $P_1$ and $P_2$. Their customers $M_1$ and $M_2$ are respectively at factories $F_1$ and $F_2$. The goods at site $F_1$ must be forwarded to the warehouse $W_1$ and that at $F_2$ to $W_2$. The quantities of goods at $F_1$ and $F_2$ are initially 15 and 10 units. The suppliers are equipped with trucks which have a capacity of 5 units. Figures 14 and 15 show the results of the application of the negotiation protocol proposed and those of the standard CNP. The bids submitted for the execution of the tasks announced relate to the time necessary to reach the site where the customer is located. Figure 14 concerns the steps of the negotiation process between the customers and the suppliers using the new protocol. For each negotiation step, we present the various messages sent by the customers and the suppliers. As an example, in Figure 14, *PreBid* $(S_1, T_1, M_2) = 5$ is a message transmitted by supplier $S_1$ to customer $M_2$ for the execution of task $T_1$. The value of the temporarily bid is 5 time units. Figure 15 show the steps of the negotiation process according to the standard CNP.

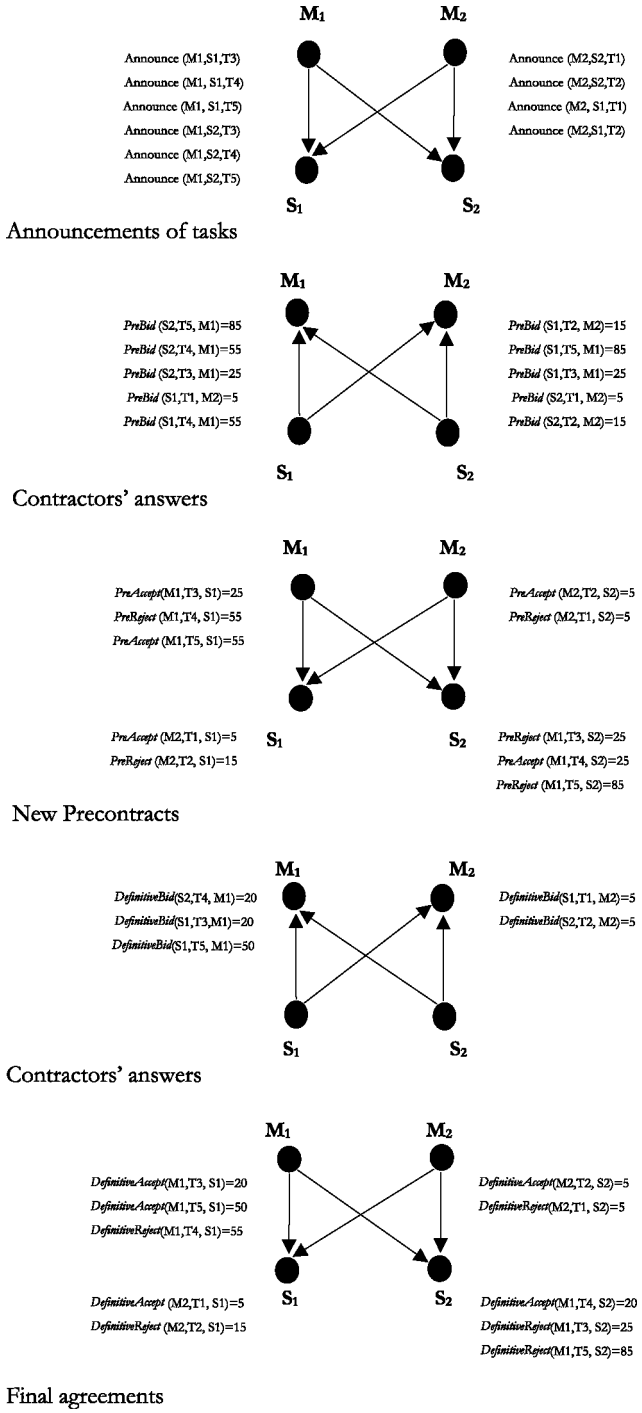



*Figure 14.* Results of the agents' negotiation process using the protocol proposed.

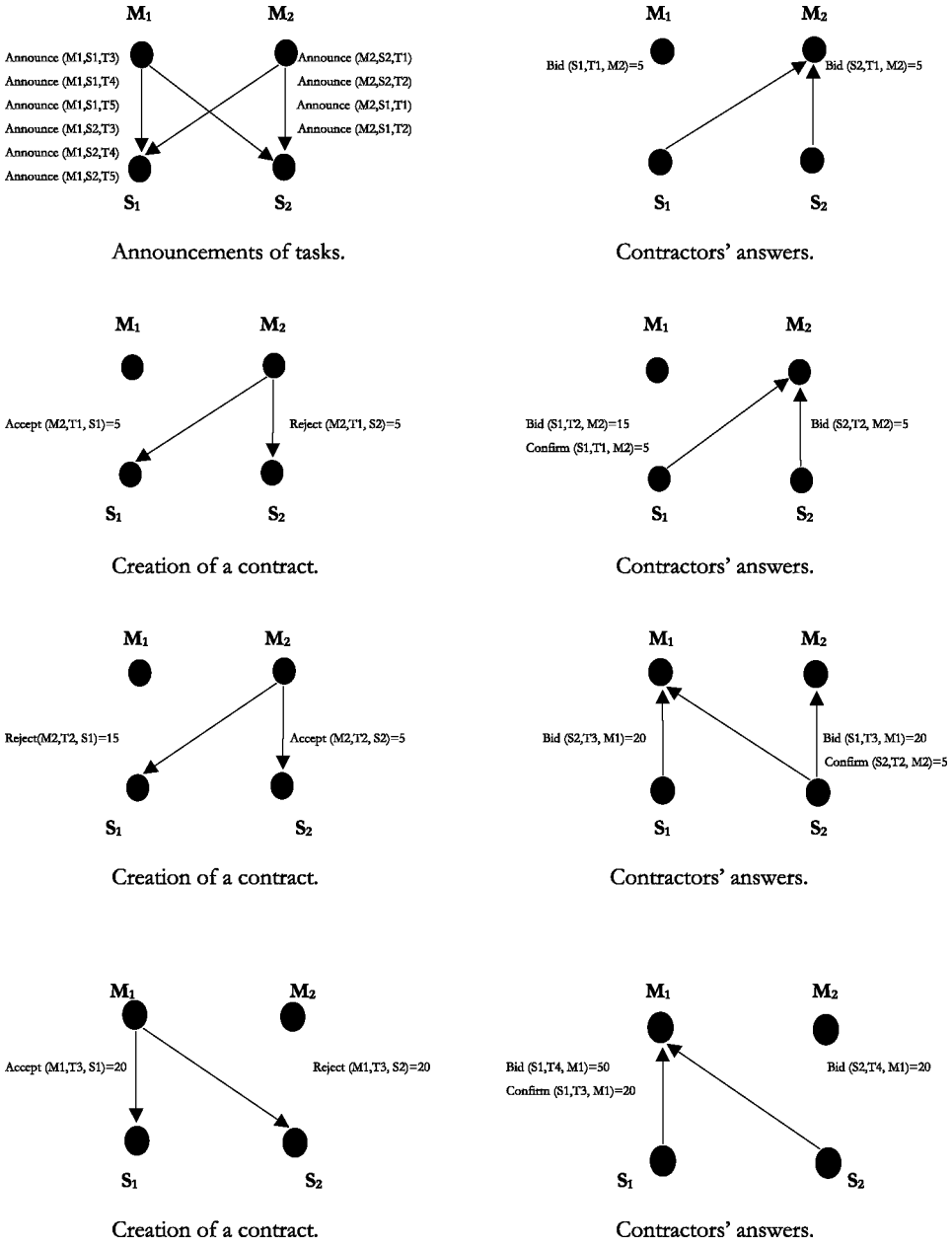



*Figure 15.* Results of the agents' negotiation process using the CNP.

In this example, the negotiation process started using the new protocol was completed performed after 5 phases. However, with the Contract Net Protocol the negotiation was completed after 12 phases. Even if we observe that more messages are exchanged in our protocol compared to the Contract Net Protocol, the additional messages do not have

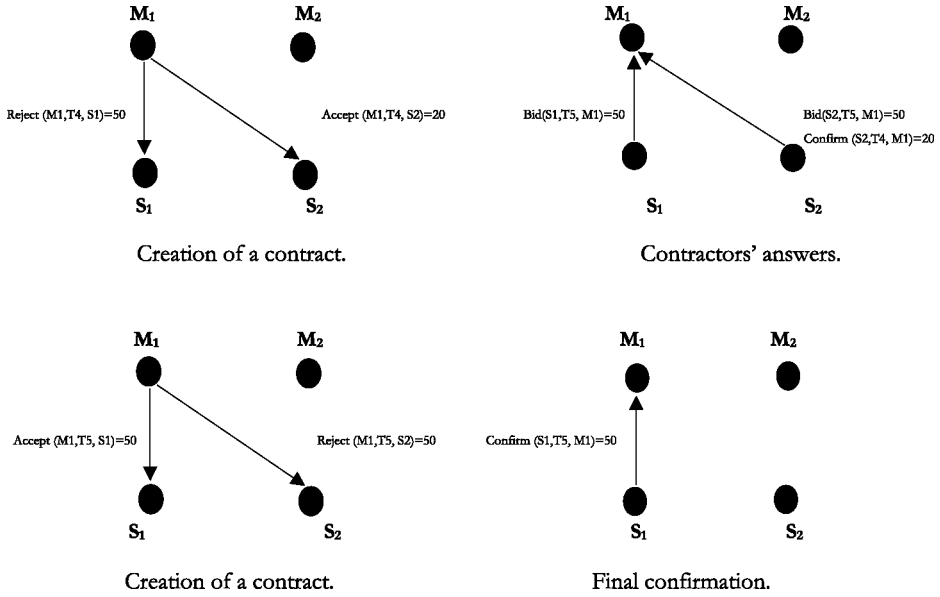



*Figure 15.* (continued).

serious consequences on the performance of our protocol. These messages are sent in packages since agents have concurrent negotiations. Consequently, there are no messages pending as in the Contract Net Protocol. This shows the advantage of our protocol in reducing the length of the negotiation processes, which has also been confirmed in the experimental results discussed below.

To evaluate the performance of our protocol, we have simulated the application described in section 3 using the Contract Net Protocol and the new suggested protocol. The agents of our system are the customers and the suppliers. This application makes it possible to measure the advantages of our protocol, i.e. shorter negotiation time and more flexibility. It makes it possible to remove or add agents and/or tasks during problem-solving process. In this sense, the multi-agent approach is preferable to an operational research approach.

In this application, the agents and their tasks have the following characteristics which are the following:

- The tasks of the agents are decomposed locally by the managers, i.e. the customers in our case. The managers propose to the suppliers to carry out tasks which correspond to the capacity of their trucks. Thus supplier agents are not concerned by the decomposition of their tasks since they are independent and it is not assumed that they know each other.
- A supplier agent is asked to respect the values of the bids which enabled it to sign contracts. Consequently, it can perform several tasks at the same time, only if it takes into account the additional loads that all the accepted tasks may induce. For instance, an agent which moves from a position $P_i$ to a warehouse $W_j$ in order to perform a task $T_k$, may accept the execution of a concurrent task $T_{k'}$. If, this task requires loading

products at a position $P_{i'}$ and/or unloading products at a warehouse $W_{i'}$, which is in the path of task $T_k$, the agent should consider in the initial bid of task $T_k$ the duration of each of these additional operations of loading and unloading goods.

- A manager agent can work with several suppliers concurrently and a supplier agent can also work with several managers concurrently.
- New tasks can be identified by the agents during the problem-solving process. Initially, tasks to be performed are not known by the suppliers. In our problem, the perception of the supplier agents is limited, i.e. they are not informed about the other suppliers which share with them the same environment but this knowledge can be gradually refined. The manager agents have precise knowledge about the supplier agents which are in their environment.
- Initially, the manager agents work independently. However, their tasks can become dependent during their execution. This can create dependence relations between these manager agents.
- The manager agents have local knowledge. In particular, they do not have knowledge about the tasks of their suppliers and other managers.
- The knowledge of the agents is exact. The communications are not disturbed and the agents are sincere, they do not communicate erroneous data.
- A supplier agent which starts the execution of a task cannot desist from this task, i.e. it cannot abandon its goods on the road.

Knowledge about the capacity of the supplier agents, the position of the production sites and the warehouses of the goods is introduced in the system at the time of its activation. In our experiments, the various transportation communication channels between the centers are memorized by the supplier agents on a map making it possible to calculate the shortest paths to be taken by the agents using the Dantzig search algorithm for the shortest path [11].
The aim of the continuation of this section is to show how we have tested the performance of the protocol proposed. Knowing that it is difficult to evaluate the complexity of the algorithms proposed in a formal way, we agree with the idea that the effectiveness of these coordination models can also be evaluated by analyzing the results of the experiments for the solving of one or more problems by the agents.

To measure the effectiveness of these models according to a time parameter, in each experiment we show the time spent by the agents to negotiate their tasks. To do so, we placed time indicators on each agent, which enabled us to follow the evolution of their plans. In both cases, i.e. using the Contract Net Protocol or the suggested extension, we used the same data.

In all 8 experiments, we varied the number of agents in the system in order to observe their behaviors while applying each of these protocols to big and small agent populations. Thus the number of agents in the system was either 4, 10, 20, 30 or 40. In each set of experiments with n agents, we also varied the number of manager and contractor agents. In the following graphs, the results of the Contract Net Protocol are shown using continuous curves and the results of the new protocol in discontinuous curves. It should be noted that the results of the Contract Net Protocol indicated in these graphs correspond to the experiments which were carried out successfully, i.e. without blockages. In several cases, the multi-agent system was blocked using the CNP in both small and big agent populations. The explanation of this blockage is certainly due to the reasons mentioned at the beginning of the presentation (cf. Section 2.1).

To be able to measure the performance of the Contract Net Protocol in these blockage cases, we were sometimes obliged to modify the data of the case, i.e. the tasks to be carried out by the agents and their appearance dates, in order to avoid the problem of data interference which does not conform to reality.

To do these experiments, we carried out several tests. We took the average of the results in order to obtain the final values indicated in the tables and graphs below which compare the performance of the protocols. In the rest of this section, we will discuss the results obtained in these experiments, organized in to four groups.

*First experimental group*

In this first experimental group, we observed the behavior of the agents during the use of these protocols by small agent populations. In the first series of this group, we tested these behaviors with an equal number of manager and contractor agents (cf. Figure 16). In the second series, we increased the number of manager agents and decreased the number of contractor agents but the total number of agents in each of the two series was the same (cf. Figure 17).

**Series 1.1.** Table 2 shows the results that we obtained after several experiments on a total population of 4 agents, i.e. 2 managers and 2 contractors. When moving from 4 to 50 tasks, we observed that the ratio of the evolution of the negotiation time using the suggested protocol is less than its evolution using the CNP. In addition negotiation time using the extension of the CNP was lower than the negotiation time using the CNP.

*Table 2.* Results of the protocols with n = 4 agents (2 managers and 2 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 1490 | 2910 | 4020 | 5940 | 11266 | 18322 |
| Average time using our extension of the CNP (Ms) | 1320 | 2530 | 3460 | 5050 | 9140 | 14258 |

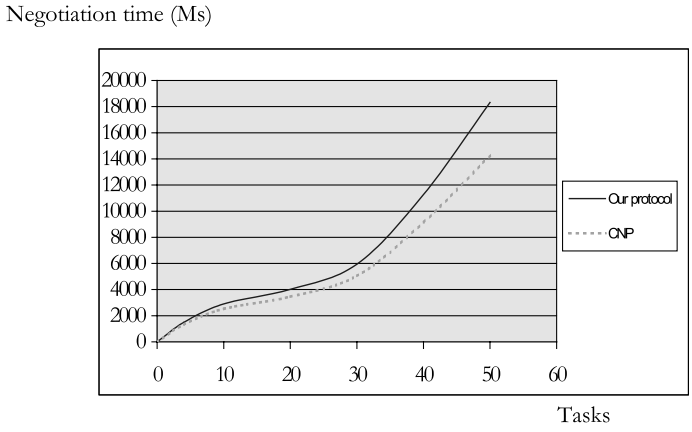



*Figure 16.* Performance of the protocols with n = 4 agents (2 managers and 2 contractors).

**Series 1.2.** In this experimental series, we observed that the agent negotiation time is shorter than in the previous series (cf. Table 3). This reduction in time is due to the fact that there is only one possible contractor agent which is asked to carry out the tasks, so there is no longer any time wasted. In addition, we have not observed blockages during the use of the CNP, due to the same reasons, i.e. the existence of only one possible contractor agent. However, negotiation time using the CNP remained higher than that using the extension of the CNP. We have observed that the difference in negotiation times using the two protocols becomes increasingly significant as the number of tasks negotiated by the agents increases (cf. Figure 17). We will see that this property characterizes our protocol and that it remains satisfied in all the following experiments. This confirms the usefulness of our protocol compared to the CNP.

*Table 3.* Results of the protocols with n = 4 agents (3 managers and 1 contractor).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 1021 | 1950 | 3732 | 4800 | 8650 | 15415 |
| Average time using our extension of the CNP (Ms) | 880 | 1588 | 2260 | 3320 | 5920 | 10060 |

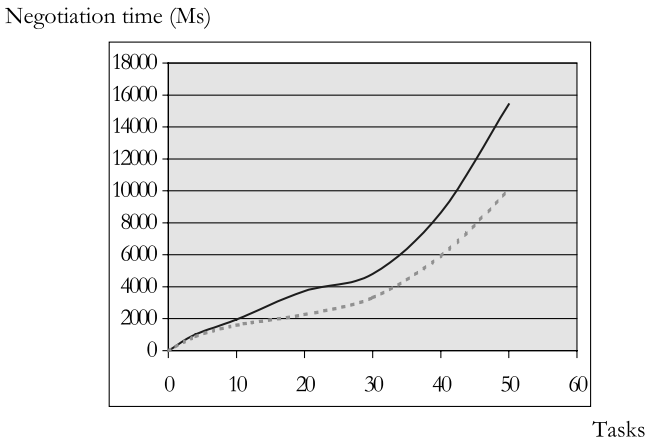



*Figure 17.* Performance of the protocols with n = 4 agents (3 managers and 1 contractor).

*Second experimental group*

It is essential to check the behavior of the agents in medium size populations before observing their behaviors on larger populations. This group of experiments gathers the results of our tests on a population of 10 agents. In the first series, we considered two manager agents and 8 contractor agents (cf. Table 4); in the second series, we used 6 manager agents and 4 contractor agents to observe the opposite situation, i.e. more managers than contractors (cf. Table 5). In this group and in each of the following tests, negotiation time needed by the agents are higher than times needed in the previous experimental group with the same sets of achieved tasks. The explanation lies in the

number of agents which has been increased in the multi-agent system. Consequently, the announcements are more widely distributed and there are more bids since there are more agents participating in this group than in the previous experimental group.

**Series 2.3.** The remarks made about the previous experimental group on the evolution of negotiation times according to the number of tasks, when using each of the two protocols, are also valid in this group (cf. Figures 18 and 19).

*Table 4.* Results of the protocols with n = 10 agents (2 managers and 8 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 1612 | 3230 | 4414 | 6814 | 14989 | 21262 |
| Average time using our extension of the CNP (Ms) | 1370 | 2580 | 3810 | 5908 | 10015 | 15298 |

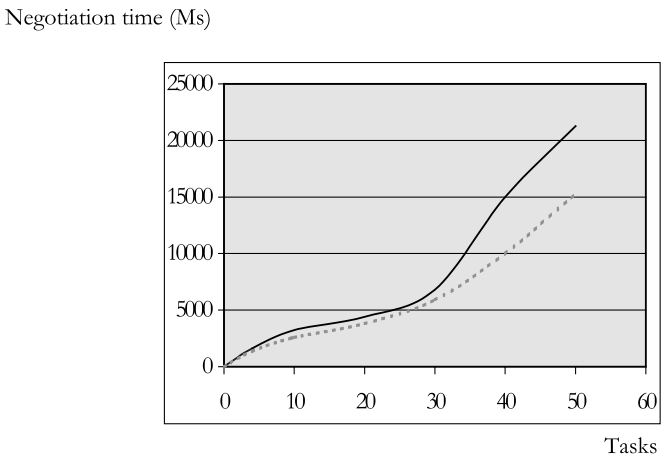



*Figure 18.* Performance of the protocols with n = 10 agents (2 manager and 8 contractors).

**Series 2.4.** In this experiment series, we considered 6 managers and 4 contractors. Compared to the previous experimental series with the same number of agents (10 agents), we have observed in this series that the negotiation time is appreciably longer. We can explain this by the fact that there are fewer contractor agents in the system compared to the previous case. The agents spend more time to analyze the tasks announced before accepting them.

*Table 5.* Results of the protocols with n = 10 agents (6 managers and 4 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 1994 | 4005 | 5904 | 7112 | 15229 | 23786 |
| Average time using our extension of the CNP (Ms) | 1540 | 3228 | 4042 | 6110 | 11789 | 16247 |

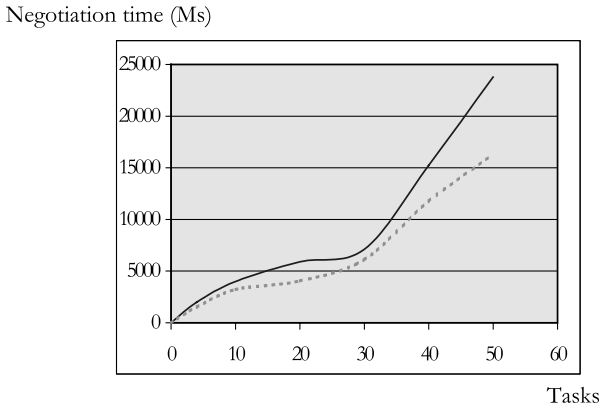



*Figure 19.* Performance of the protocols with n = 10 agents (6 managers and 4 contractors).

*Third experimental group*

From the results in Tables 6 and 7, we see that the difference between the negotiation times based on the new protocol and the CNP has become larger than that obtained in the previous experiments. These results are obtained on a population of 20 agents. We have varied the number of tasks from 4 to 50 tasks in each of this series (cf. Figures 20 and 21).

**Series 3.5**

*Table 6.* Results of the protocols with n = 20 agents (4 managers and 16 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 2398 | 5989 | 7024 | 12587 | 19365 | 28001 |
| Average time using our extension of the CNP (Ms) | 1794 | 3824 | 4225 | 6335 | 12569 | 18549 |

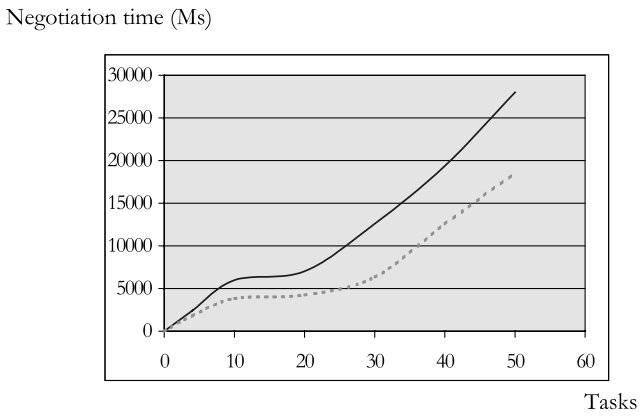



*Figure 20.* Performance of the protocols with n = 20 agents (4 managers and 16 contractors).

**Series 3.6.** Considering that the number of agents is higher in this group of experiments, it is obvious that the negotiation time will be longer than that obtained in the last experimental group. When comparing the first series with the second series, the difference between the results is also appreciably greater, for the same reason as for the last group.

*Table 7.* Results of the protocols with n = 20 agents (14 managers and 6 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 2998 | 6581 | 9004 | 13101 | 20879 | 28925 |
| Average time using our extension of the CNP (Ms) | 2242 | 4255 | 4651 | 6700 | 10589 | 19072 |

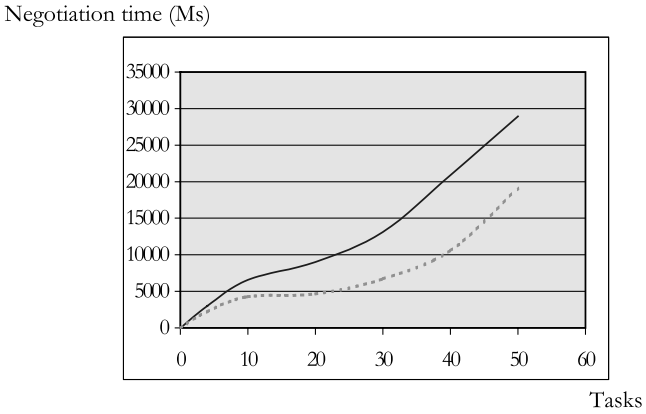



*Figure 21.* Performance of the protocols with n = 20 agents (14 managers and 6 contractors).

*Fourth experimental group*

In this last group of tests, we increased the number of agents in the system to 40 (cf. Figures 22 and 23). We will see that sometimes the difference between the results of the two protocols regarding negotiation time is small, which is shown at the beginning and the end of Figures 22 and 23. This is due to the fact that the agents take more time, in these cases, to find an agreement using our protocol. Nevertheless, our protocol remains better than the Contract Net Protocol regarding the other advantages it offers and that the Contract Net Protocol does not have.

**Series 4.7.** In this series, the total number of agents is 40, with 14 managers and 26 contractors (cf. Table 8).

*Table 8.* Results of the protocols with n = 40 agents (14 managers and 26 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 4988 | 8225 | 12244 | 19698 | 24334 | 29950 |
| Average time using our extension of the CNP (Ms) | 3509 | 7901 | 10821 | 16941 | 18788 | 29872 |

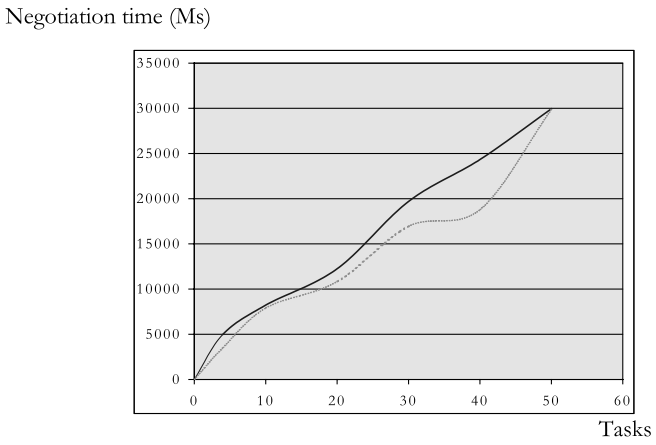



*Figure 22.* Performance of the protocols with n = 40 agents (14 managers and 26 contractors).

**Series 4.8.** The results in the following table were obtained with 40 agents, 24 managers and 16 contractors (cf. Table 9).

*Table 9.* Results of the protocols with n = 40 agents (24 managers and 16 contractors).

| Number of tasks | 4 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Average time using the CNP (Ms) | 5209 | 9245 | 13202 | 20578 | 25789 | 30589 |
| Average time using our extension of the CNP (Ms) | 3621 | 8189 | 10934 | 16991 | 18933 | 29978 |

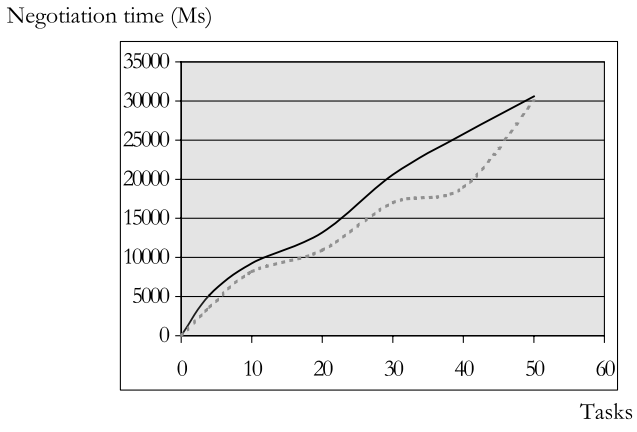



*Figure 23.* Performance of the protocols with n = 40 agents (24 managers and 16 contractors).

In the previous section, we have presented and formally analyzed the properties satisfied by our protocol. Knowing that a formal analysis of these properties is not sufficient to assert the applicability of the protocol, we performed several series of tests in order to confirm the results that we announced. All the experiments show that the time performance

of the protocol proposed is superior to that of the Contract Net Protocol. Moreover, the protocol proposed satisfies other properties which improve its performance compared to those of the Contract Net Protocol which does not satisfy them.

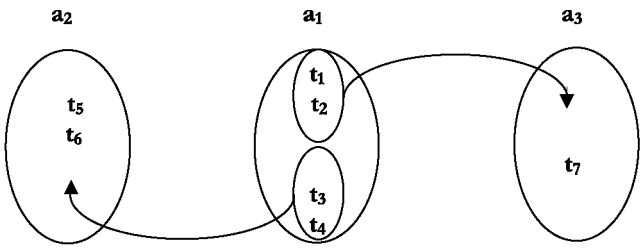
## 5. Related work

Several models have been proposed to solve the problem of coordination in multi-agent systems. Jennings presented a system that handles task allocation by having a central analyzer that uses information it has about the abilities of all the other agents to design tasks [17]. This solution requires a central agent. El Fallah and Haddad worked on distributed planning in multi-agent systems using recursive Petri nets as a formal representation model of agent actions [13]. This model involves the sharing of plans among the agents. Distributed Artificial Intelligence has previously addressed tasks with order of precedence and with overlapping problem solvers as in PGPG [12]. In this work, coordination is based on an organizational view of node activity where each node acts subject to its local control, solving a sub-goal of the global goal. In another approach a group of agents has to achieve a common task [24], but this is beyond the scope of our work.

In this article we focus on the Contract Net Protocol and its extensions. We only describe this work in order to show the difference between the aim of our protocol and that of the existing protocols and also in order to analyze the advantages and limitations of each protocol [1].

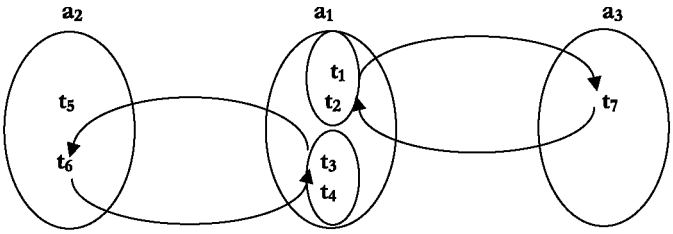
In the first approach of Sandholm [20] (Marginal cost-based contracting), the decisions of the agents are based on the calculation of marginal costs. Sandholm introduced the concept of rationality of a contract for an agent. He considers that a contract is rational for an agent if the agent is in a better state if it accepts the contract than if it does not. An agent accepts a contract if it increases its payoff by taking into account the costs for executing the contracted tasks. In this new protocol, the agents propose contracts to other agents and accept or reject contracts based on their marginal costs.

In the second approach of task allocation [21], Sandholm and Lesser were interested in the problem of allocating several tasks at the same time to the same agent (cf. Figure 24). This figure indicates that a manager agent can gather several tasks under the same contract




*This figure describes a situation in which agent $a_1$ proposes to agent $a_2$ the set of tasks formed by $t_3$ and $t_4$. $a_2$ agrees to carry them out. $a_1$ accepts that agent $a_3$ executes the set of tasks $t_1$ and $t_2$.*

*Figure 24.* Grouped allocation of tasks.




*This figure describes a situation in which agent $a_1$ proposes to agent $a_2$ the set of the tasks formed of $t_3$ and $t_4$. $a_2$ agrees to carry out these two tasks offered by $a_1$ if this agent deals with task $t_6$.*

*Figure 25.* Exchanging tasks.

with one agent. According to Sandholm and Lesser, this approach reduces the execution time of the tasks when they are inter-dependent because the necessary means for the execution of a task can be used to carry out the other tasks. Nevertheless, this extension has some limitations because it does not answer one of the problems raised by this recombination of tasks: should a manager favor the agents which bid for a set of tasks or those which bid only for some tasks and for what kind of application should these agents be preferred to others?

In all the cases, if the agents are self-interested the bids sent by the contractors are likely to be smaller than those which can be sent using the Contract Net Protocol. The reason is that the contractor agents will tend to reduce their bids since they propose to perform several tasks at the same time. In this sense, we think that this extension remains strongly dependent on the application domain.

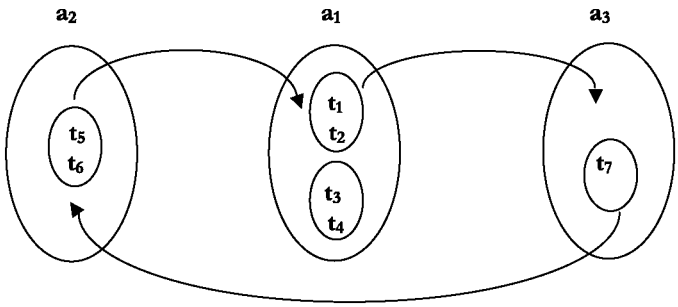
In order to improve the performance of the agents, Sandholm and Lesser consider that the agents which carry out their tasks separately can sometimes exchange certain contracts (Swap-contracts). Figure 25 described this situation. The protocols defined in this approach for self-interested and cooperative agents are different. A cooperative agent can be committed to carry out the exchanged task without compensation by the manager of the contract. However, an individualistic or a self-interested agent requires compensation for each additional processing carried out compared to the processing of its original task.

We think that this kind of protocol is not sufficient. In certain applications, in order for this protocol to be used it must be combined with other protocols because the agents may not want to exchange their tasks, for instance, when considering confidentiality constraints.

The fourth approach presented by Andersson and Sandholm [7, 8] shows that it is possible to make contracts circulate between agents by exchanging messages. A contract can be considered as valid if one of the interested agents signs it. Figure 26 described this situation.

This protocol remains applicable only for cooperative multi-agent systems in which the contractor agents can contribute to improvng the performance of the managers. However, self-interested agents will not disseminate the information announced, in order to increase their chances to obtain the contracts.

In a cooperative multi-agent system, the advantage of this approach remains in the fact that the announcements will be widely disseminated between the agents, i.e. each agent disseminates them to the other agents it knows. This may facilitate the application of




*This figure indicates that agent $a_1$ has received a call for proposal for tasks $t_5$ and $t_6$ from agent $a_2$. $a_1$ is not interested in these tasks, it sends the information to agent $a_3$. $a_3$ can also propose one of its tasks to agent $a_2$.*

*Figure 26.* Task allocation by circulation of announcements.

several agents for an announcement. Nevertheless, there are still serous limitations: (1) the multi-agent system is likely to be easily saturated by messages duplicated by the agents which concurrently send the same bids. With $n$ agents in the system and $k$ tasks to be executed, $(kn)^2$ bids may circulate at the same time between the agents; (2) consequently, the processing of each agent will thus be unnecessarily increased by these messages; (3) negotiation time thus increases.

The Leveled Commitment Protocol presented in [6, 7, 21, 26] considers that an agent may be decommitted from a contract if it pays a certain penalty required by the manager of the contract. However, using penalties is not suitable for cooperative multi-agent systems in which the agents voluntarily take part in the execution of the tasks of the other agents. The limits of this approach is that penalties in these systems will prevent the agents from voluntarily offering to execute the tasks. Penalties are not always easy to calculate, and a consensus between the agents is necessary to choose these values. Sandholm et al started to address this work in [23].

Sandholm has also highlighted the limits of all the approaches quoted above, in that they are context-dependent and so we have to decide which protocol to implement, depending on the context.

In more recent and promising work, [6, 7] have addressed the problem of reallocating tasks in multi-agent systems. They propose a way to enable the agents to use several protocols during their negotiation in the same application. In their work, they have compared sequences of four of their contract types that we have discussed earlier. They have tried to provide a guideline for the designers of the multi-agent systems regarding the contract types to use and how to sequence them. However, these experiments are based on certain assumptions.

(1) The experiments have been carried out with a population of 8 agents and 8 tasks. We understand that it is difficult to increase the number of agents and tasks, because of the computational complexity problem. But we also believe that it is very difficult to retain general and relevant conclusions from experiments on populations with a low number of agents. In reality, there are many multi-agents systems with large numbers of agents than 8. This approach needs to be tested on a large number of tasks like in [20] where the tests have been done on 771 tasks.

(2) The application and the domain tasks were initially known by the authors (designers of the multi-agent system). Negotiation time has been divided into several intervals and agents knew which protocol to apply a priori in such or such time interval. Thus there is certain knowledge a priori (a certain determinism), which makes it difficult to generalize the results obtained. In a real application, the choice of a protocol is determined by the task to be performed. Consequently, considering that agents do not know a priori either the tasks, which can be completely different from each other, or their order or their occurrence dates, they do not know at what time to apply such or such protocol. What would be interesting would be to perform tests on various applications to confirm the results obtained on the relevance of the best sequences of protocols that they found.

(3) Another aspect which is not developed by the authors in their work is how agents can switch individually and dynamically from one protocol to another without needing the specifications of the designer. Of course, this problem does not arise in their application since the duration of the validity of a protocol has been fixed in advance. The cost of the solution should therefore also include the cost that is necessary for agents to make agreements on the protocol to be used. In addition, they should check that the consensus on the protocol to be adopted is easily achievable by the agents when it has not been fixed by the designer. It would be interesting to perform tests and to evaluate the quality of the results obtained by the agents while taking into account all these constraints.

Having said this, trying to categorize negotiation protocols according to the applications and the tasks that they contain would undoubtedly be a very promising step to improve negotiation between agents and would facilitate the work of the designers of these systems.

In order to apply the CNP to a problem of transport (Autonomous Cooperative Shipping Companies) [14], Fischer, Müller, and Pischel have decentralized the process like in [20]. This enables the contractors to decompose the tasks into sub-tasks, enabling them to bid for the set of the task announced by the manager or for subtasks when the contractors are not able to perform all of the task. However, this extension is domain-dependent, i.e. the extension can be applied only to those whose tasks are complex and can be decomposed.

In addition, it does not solve the problem that may be created by this decomposition. As we have seen for the extension suggested by Sandholm and Lesser, should a manager favor agents which bid for a set of tasks or those which bid only for some tasks, and for what kind of application should these agents be preferred? In all the cases, we also think that if agents are self-interested, the bids sent by the contractors are likely to be smaller than those which can be sent under the Contract Net Protocol. The reason is that the contractor agents will tend to reduce their bids since they propose to perform several tasks at the same time. In this sense, we think that this extension remains strongly dependent on the application domain.

This approach is similar to the one presented in the work of Bouron [9]. Bouron considers that when an agent carries out a task contracted with a manager, the agent may break this task into sub-tasks, if it is not able to carry out the whole of the task. It must behave as a manager for these new sub-tasks for which it must find contractors. This process can be reproduced by each new sub-contractor. A contractor agent confronted to such a situation may behave in two different ways.

- It may ask for assistance once it has been granted the contract. This solution, known as fast commitment of the agents, can block the manager and the contractor if the latter cannot find the agents necessary for the achievement of its task. The contractor must then break its commitment;
- It can also seek the agents likely to help it in the achievement of its task before being accepted by the manager. This solution, known as late commitment of the agents, involves more problems than the previous one. The contractor will try to accept other agents for the execution of tasks it is not even sure will be allocated. It may easily create a fatal deadlock in the system.

Lee considers that a negotiation process can be subdivided into several negotiation sub-processes [18]. Initially, it is assumed that each agent is informed of a set of tasks. The agents must negotiate between themselves to reallocate the tasks which they have contracted.

At a given instant, each agent announces a set of tasks that it tries to allocate. The agents select from among these announcements those that maximize their utility. In the event of infinite blocking on task reallocation, the agents can stop the negotiation process and can decide to carry out the tasks which were initially affected to them.

This negotiation model contains several limitations: (1) it leads to fast commitments of the contractors, which involves signing contracts which will perhaps not be honored; (2) it is difficult to find contractors which will perform the tasks accepted by the first contractor agents. Since the first agents reallocate these tasks, they must necessarily ask for higher bids from the second contractors compared to those they have themselves submitted to the original managers of these same tasks. This enables them to compensate for the processing done; (3) the execution time of the accepted tasks may be longer if the contractors do not find the agents necessary to achieve the tasks.

## 6. Summary

In the first part of this article we presented a new negotiation protocol which extends the Contract Net Protocol defined by Smith and Davis. Our protocol presents the following advantages: (1) it enables *M-N* negotiations, i.e. a contractor agent can manage concurrently several negotiation processes with *m* manager agents, and a manager agent can manage concurrently several negotiation processes with *n* contractor agents; (2) it is more efficient in time and it is fault tolerant. We then proposed the algorithms describing the behaviors of each contractor and manager agent and gave concrete examples to illustrate the use of this protocol. In the second part, we analyzed the properties of our protocol. First, we showed the convergence property of our protocol, then we analyzed the properties for releasing a blocked negotiation process with failed agents. In order to evaluate our protocol, we presented a goods delivery application, where each truck is represented by an agent which negotiates to deliver goods while taking into account its capacity and the distance from the tasks to be performed. We described the experiments and the simulation results. Finally, we discussed related work on multi-agent coordination, mainly on task allocation. We have deliberately presented different work extending Smith and Davis's protocol in order to show the difference between our protocol and each existing protocol.

In this article, we have shown the use of our protocol or solving agent coordination problems by task allocation. We plan to extend this research to electronic commerce. First, we intend to test the protocol and show its efficiency in a national project, electronic commerce with intelligent negotiator agents [16]. Then, we plan to tackled the problem of more complex negotiation processes, i.e. *M-N-P* negotiations with *m* sellers, *n* buyers and *p* dependent products [5] using our proposed protocol.

## Appendix A. Annexes

**Proof of lemma 2:** To prove this lemma, we reason by induction on the number j of tasks announced to an agent $a_i$.

- For $j = 1$. When only one task $T_k$ is announced to agent $a_i$, it is obvious that after a rejection no rearrangement is possible for $T_k$. With a preference order $p = 1$, $Neg_{i,k}$ cannot be rearranged.
- For $j = 2$. Let $T_k$ and $T_{k'}$ be two tasks for which agent $a_i$ negotiates the execution. Let $Neg_{i,k} = (a_i, M, T_k, P, s, \upsilon)$ and $Neg_{i,k'} = (a_i, M', T_{k'}, P, s', \upsilon')$ be two negotiation processes respectively associated with $T_k$ and $T_{k'}$.

(1) Assuming that $T - ord_i(T_k) < T - ord_i(T_{k'})$, $p = 1$:

$PreBid(T_k) \geq PreBid(T_{k'})$, there is not a better arrangement for $T_k$.

(2) If $T - ord_i(T_k) > T - ord_i(T_{k'})$, $p = 2$:

$PreBid(T_k) \leq PreBid(T_{k'})$; two cases are then possible :

**Case$_1$.** $s' \in \{$*Pre*R*eject*, *Definitive*R*eject*$\}$

As $a_i$ is authorized to change the schedule of the tasks announced when it is rejected to a contract (Section 2), there is a better arrangement for $T_k$ such that:

$T - Ord_i(T) = (T_k, T_{k'})$. $Neg_{i,k}$ can be $(p - 1 = 1)$ time rearranged.

**Case$_2$.** $s' \notin \{$*Pre*R*eject*, *Definitive*R*eject*$\}$

There is no better arrangement for $T_k$ which respects the choice of the agent such that: $T - Ord_i(T) = (T_{k'}, T_k)$.$Neg_{i,k}$ cannot be rearranged.
$Neg_{i,k}$ can thus be at most $(p - 1)$ time positively rearranged by $a_i$.

- Assume that at the instant $\delta$, m tasks have been announced to $a_i$, with $T\text{-}ord_i(T_k) = p$, $Neg_{i,k}$ is at most $(p - 1)$ time positively rearranged by $a_i$.
- We must show that at the next instant $\delta'$, if a new task is announced to $a_i$ and if $T\text{-}ord_i(T_k) = p$, $Neg_{i,k}$ can be at most $(p - 1)$ time positively rearranged by $a_i$, and with $T\text{-}ord_i(T_k) = p + 1$, $Neg_{i,k}$ can be at most (p) time positively rearranged by $a_i$

Let $T^m$ be a set of tasks announced to $a_i$ at the instant $\delta$ and $T^{m+1}$ be the set of tasks announced at the instant $\delta'$, let $T_x$ be a new task, $T^{m+1} = T^m \cup T_x$.

Let $Neg_{i,x} = (a_i, M', T_x, P, s_x, \upsilon_x)$ be the negotiation process associated with the execution of $T_x$. Several cases are possible according to the schedule of $T_x$ compared to $T_k$ in $T - Ord_i\ (T^{m+1})$.

**Case$_1$.** $\text{T-ord}_i\ (T_x) > \text{T-ord}_i\ (T_k)$, $\text{T-ord}_i\ (T_k) = p$

Considering that the rearrangement of $T_k$ depends only on the tasks $T_l \in T^{m+1}$ such that $\text{T-ord}_i\ (T_l) < \text{T-ord}_i\ (T_k)$, $Neg_{i,k}$ is then independent of $Neg_{i,x}$.
The set of negotiation processes on which $Neg_{i,k}$ depends, at instant $\delta'$ is the same that those at instant $\delta$. Consequently, $Neg_{i,k}$ is at most $(p - 1)$ positively rearranged by $a_i$.

**Case$_2$.** $\text{T-ord}_i\ (T_x) < \text{T-ord}_i\ (T_k)$, $\text{T-ord}_i\ (T_k) = p + 1$ and $Neg_{i,k}$ depends on $Neg_{i,x}$.
.If $s_x \in \{PreReject, DefinitiveReject\}$

Considering the algorithm describing the behaviors of contractor agent $a_i$, task $T_x$ can necessarily be reordered, so $\text{T-ord}_i\ (T_k)$ changes by 1 degree (cf. Figure 10). Since *PreBid* of $T_k$ grows according to $\text{T-ord}_i\ (T_k)$, reordering $T_x$ gives a positive rearrangement of $Neg_{i,k}$.
From the previous assumption, $T_k$ can be at most $(p - 1)$ positively reordered by $a_i$ at instant $\delta$; thus we have $Neg_{i,k}$ being at most p positively rearranged by $a_i$ at instant $\delta'$.

.If $s_x \notin \{PreReject, DefinitiveReject\}$
There is not a better arrangement for $T_k$ which respects the choice of the agent:

$$T - Ord_i\ (T^{m+1}) = (.., T_x, .., T_k, ..).$$

$Neg_{i,k}$ is thus at most $(p - 1)$ time positively rearranged by $a_i$.

In all the cases, we have proved that if $\text{T-ord}_i\ (T_k) = p$, $Neg_{i,k}$ is at most $(p - 1)$ time positively rearranged by $a_i$.
Now that we have proved this lemma, we can give the proof of the theorem on the termination of the algorithm.

**Proof of theorem 2:** Observation of the automaton in Figure 2 describing the possible transitions of a negotiation between a manager and its contractors reveals that loops can occur during the application of the protocol. There are two cases of loops:

(a) A first loop on states (2) and (4), i.e. the contractors block on sending messages of the type *PreBid* and the manager blocks on rejecting these contractors;
(b) A second loop on states (3), (5) and (4).

To show that the protocol does not block the manager and its contractors, we must show that there is no infinite sequence of loops on (a) and (b). Let us see the two loops separately.

I *Loops on (a)*

Let S be a set of agents in the multi-agent system and $a_m$ a manager agent of a negotiation process for the execution of a task $T_p$. Each time that a contractor agent receives an

announcement from $a_m$ for the execution of a task $T_p$, it returns a message comprising its bid, i.e. *PreBid* ($a_r$, $T_p$) for $a_m$. When the manager receives this message, it sorts $a_r$ (definition 3) in its list of contractor agents (cf. Figure 11). According to the rank of $a_r$, the manager $a_m$ can adopt one of the two following behaviors:

1. $a_m$ returns for $a_r$ a message of the type *PreAccept* ($a_r$, $T_p$) if $\forall\, a_{i/i\neq r}$, A-ord$_p$ ($a_r$) < A-ord$_p$ ($a_i$), $a_r$ is at the head of the list of the contractors of $a_m$ for the announced task.
2. $a_m$ returns for $a_r$ a message of the type *Pre*R*eject* ($a_r$, $T_p$) if there is at least a potential contractor $a_{i/i\neq r}$, such that *PreBid* ($a_i$, $T_p$) $\geq$ *PreBid* ($a_r$, $T_p$) and $a_m$ will send a message of the type *PreAccept* ($a_i$, $T_p$) for $a_i$.

In the first case, the negotiation between $a_m$ and $a_r$ goes to state (3) in Figure 2, i.e. comes out of the loop (A). In the second case, the negotiation with contractor $a_r$ goes to state (4), i.e. looping on (a).

Because of the time constraint answers to the messages, which we have imposed on the manager and the contractor, and owing to the fact that the pre-rejected contractors can modify their *PreBid*s if their situation evolves, the states of the negotiation between the manager and all its contractors depend on the behaviors adopted by the potential contractor $a_i$. These behaviors are the following:

1. The potential contractor $a_i$ returns a *DefinitiveBid* ($a_i$, $T_p$) $\geq \underset{1\leq j\leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)].

   The evaluation of the *DefinitiveBid* of $a_i$ by $a_m$ involves the signing of the contract between $a_m$ and $a_i$, and the final rejection of contractor $a_r$ from the negotiation process with $a_r$ which goes to state (7), i.e. comes out of the loop (a).
2. The potential contractor agent $a_i$ returns a *DefinitiveBid* ($a_i$, $T_p$) < $\underset{1\leq j\leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)] and

$$\forall\, a_i \in S, j \neq i,\ \textit{DefinitiveBid}\ (a_i, T_p) \geq \textit{PreBid}\ (a_j, T_p)$$

   The negotiation between the manager and its contractors finishes in the same way as for case (1), i.e. comes out of the loop (a).
3. The potential contractor agent $a_i$ returns a *DefinitiveBid* ($a_i$, $T_p$) < $\underset{1\leq j\leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)] and

$$\exists\, a_i \in S, j \neq i \text{ such that } \textit{DefinitiveBid}\ (a_i, T_p) < \textit{PreBid}\ (a_j, T_p)$$

   As $a_r \in S$ and $a_r$ is in the list of pre-rejected contractors, several cases are then possible.

- **Case$_{3.1}$.** $a_j = a_r$. $a_r$ is then temporarily accepted by $a_m$. The negotiation between $a_r$ and $a_m$ goes to state (3) in Figure 2, i.e. comes out of the loop (a).
- **Case$_{3.2}$.** $a_j \neq a_r$. The negotiation between $a_j$ and $a_m$ goes to state (3). The manager sorts again the pre-rejected contractors according to their *PreBid*s and integrates the potential contractor ($a_i$) in the list of the pre-rejected contractors according to its *DefinitiveBid*.

The next state of the negotiation between $a_m$ and $a_r$ depends on the rank of the new pre-rejected contractor ($a_i$) and on the behavior of the new potential contractor ($a_j$).

- If *DefinitiveBid* ($a_i$, $T_p$) < *PreBid* ($a_r$, $T_p$), the schedule of the agents of S by $a_m$ for the execution of task $T_p$ is such that:

$$A - Ord_p(A) = (a_j, .., a_r, .., a_i)$$

Based on lemma 2, the negotiation between $a_m$ and $a_r$ for the execution of task $T_p$ converges to a state of *PreAccept* ($\geq$), i.e. comes out of the loop (a).

- If *DefinitiveBid* ($a_i$, $T_p$) $\geq$ *PreBid* ($a_r$, $T_p$)
The schedule of the agents of S by $a_m$ for the execution of task $T_p$ is such that:

$$A - Ord_p(A) = (a_j, .., a_i, .., a_r)$$

The negotiation between $a_m$ and $a_r$ for the execution of task $T_p$ converges then to a final rejection state (7), i.e. comes out of the loop (a).

II *Loops on (b)*

In the same way as above, we show that there is no infinite loop on states (3), (5) and (4) of Figure 2.

In order to enable the negotiation between contractor $a_r$ and manager $a_m$ to go through state (3) (cf. Figure 2), contractor $a_r$ must send a

$$PreBid\ (a_r, T_p) \geq \underset{1 \leq j \leq n}{\text{Max}}[PreBid\ (a_j, T_p)]$$

and to go to state (5), $a_r$ must return a *DefinitiveBid* ($a_r$, $T_p$). According to the final bid of $a_r$, several cases are possible:

1. *DefinitiveBid* ($a_r$, $T_p$) $\geq \underset{1 \leq j \leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)]
   The negotiation between manager $a_m$ and all its contractors ends with the signing of the contract with $a_r$ and the final rejection of all the contractors $a_j \in S$ with $j \neq r$, i.e. comes out of the loop (b).
   Knowing that the contractors, even if pre-rejected, can send *PreBid*s when their situations evolve, other cases are also possible.
2. *DefinitiveBid* ($a_r$, $T_p$) < $\underset{1 \leq j \leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)] and

$$\forall\ a_j \in S, j \neq r,\ DefinitiveBid\ (a_r, T_p) > PreBid\ (a_j, T_p)$$

   The manager signs the contract with $a_r$, which involves the transition of the negotiation with $a_r$ towards state (6), i.e. coming out of the loop (b).
3. *DefinitiveBid* ($a_r$, $T_p$) < $\underset{1 \leq j \leq n}{\text{Max}}$ [*PreBid* ($a_j$, $T_p$)] and

$$\exists\ a_j \in S, j \neq r, \text{ such that } DefinitiveBid\ (a_r, T_p) < PreBid\ (a_j, T_p)$$

The state of the negotiation between contractor $a_r$ and manager $a_m$ is in state (5) in graph of negotiation transitions (cf. Figure 2). Several cases arise:

- **Case$_{3.1}$.** Manager $a_m$ receives a confirmation from $a_j$ for its *PreBid* before receiving a *DefinitiveBid* from $a_r$. The negotiation between $a_m$ and $a_r$ is immediately interrupted with a transition to state (7). The contract is signed with $a_j$, i.e. coming out of the loop (b).
- **Case$_{3.2}$.** The new potential contractor $a_j$ has not yet confirmed its *PreBid* to manager $a_m$. Contractor $a_r$ is thus pre-rejected with a transition of the negotiation state with $a_m$ towards (4) (cf. Figure 2). It will be reordered in the list of pre-rejected contractors. In the same way as for case I, we can show that the negotiation between $a_r$ and $a_m$ converges to a state of *DefinitiveAccept* or *DefinitiveReject* according to the *DefinitiveBid* sent by $a_j$.

In all the cases, we have shown that the negotiation process does not loop; the examples presented in section 3 illustrate some of the cases described in the proof of this theorem.

## References


1. S. Aknine, S. Pinson, and M. F. Shakun, "New multi-agent models for multiple negotiations," *French Artificial Intelligence Review*, vol. 15, no. 1, 2001.
2. S. Aknine and S. Pinson, "Reliable algorithms for multi-agent task allocation," *8th International Conference on Intelligence Systems*, Colorado, USA, June 24–26, 1999.
3. S. Aknine, "Issues in cooperative systems: Extending the contract net protocol," *IEEE Joint Conference on the Science and Technology of Intelligent Systems*, Maryland USA, September 14–17, 1998.
4. S. Aknine and S. Pinson, "Un nouveau protocole de négociation flexible pour la coopération multi-agent," in M. P. Gleizes and P. Marcenac, (eds.), *Ingénierie des Systèmes Multi-agents*, Hermès, pp. 165–174, 1999.
5. S. Aknine, "Strategies and behaviors of agents in multi-phased negotiations," *Third International Conference on Electronic Commerce and Web Technologies, EC-WEB with DEXA*, LNCS, Springer Verlag, September 2–6, Aix-en-Provence, France, 2002.
6. M. Andersson and T. Sandholm, "Time-Quality Tradeoffs in reallocative Negotiation with Combinatorial Contract Types," AAAI, 1999.
7. M. Andersson and T. Sandholm, "Contract type sequencing for reallocative negotiation," *International Conference on Distributed Computing Systems*, ICDCS, 2000.
8. M. Andersson and T. Sandholm, "Leveled commitment contracting among individually rational agents," *International Conference on Multi-agent Systems*, Paris, 1998.
9. T. Bouron, "Structures de Communication et d'Organisation pour la Coopération dans un Univers Multi-agent," Ph.D. Thesis, Université Paris 6, 1992.
10. T. D. Chandra and S. Toueg, "Unreliable failures detectors for reliable distributed systems," *Journal of the ACM*, vol. 43, no. 2, 1996.
11. G. B. Dantzig, "All shortest routes in a graph," *On Theory of Graphs*, Rome, Gordon, 1966.
12. E. H. Durfee and V. R. Lesser, "Partial global planning: A coordination framework for distributed hypothesis formation," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 21, no. 5, 1987.
13. El-Fallah Seghrouchni, A. et Haddad, S. "A coordination algorithm for multi-agent systems", *Agents Breaking Away, European Workshop on Modelling Autonomous Agents in a Multi-agent World, Maamaw'96*, The Netherlands, 1996.
14. K. Fischer, J. P. Muller, and M. Pischel, "Scheduling an application domain for DAI," *Applied Artificial Intelligence, An International Journal*, vol. 10, pp. 1–33, 1996.
15. R. Guerraoui and A. Schiper, "Consensus service: A modular approach for building agreement protocols in distributed systems," *26th IEEE Symposium on FTCS*, Sendai, Japan, June 1996.

16. K. Hamdouni, *M-N negotiation models for electronic commerce*, Master Thesis (French Language), Paris-Dauphine University, 2001.
17. N. R. Jennings, "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions," *Artificial Intelligence*, vol. 75, no. 2, 1995.
18. L. C. Lee, "Progressive multi-agent negotiation," *Second International Conference on Multi-Agent Systems*, 1996.
19. M. Raynal, "Revisiting the non-blocking atomic commitment problem in distributed data management systems," *Ingénierie des Systèmes d'Information*, vol. 5, no. 6, 1997.
20. T. Sandholm, "An Implementation of the contract net protocol based on marginal cost calculations," *11th National Conf. On AI, AAAI*, 1993.
21. T. Sandholm and V. Lesser, "Issues in automated negotiation and electronic commerce: Extending the contract net framework," *ICMAS-95, First International Conference on Multi-agent Systems*, MIT Press, 1995.
22. T. Sandholm and V. Lesser, "Advantages of a leveled commitment contracting protocol," *AAAI'96, National Conference on Artificial Intelligence*, Portland, 1996.
23. T. Sandholm, S. Sikka, and S. Norden, "Algorithms for optimizing leveled commitment contracts," *International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, pp. 535–540, 1999.
24. O. Shehory and S. Kraus, "Methods for task allocation via coalition formation," *Artificial Intelligence, Elsevier Science*, 1998.
25. R. G. Smith, "The Contract net protocol: High-level communication and control in a distributed problem-solver," *IEEE Transactions on Computers*, vol. 12, 1980.
26. R. G. Smith and R. Davis, "Frameworks for co-operation in distributed problem solving," *IEEE Transaction on System, Man and Cybernetics*, vo. 11, no. 1, 1981.
27. G. Weiss, (ed.), *Multiagent Systems: A modern approach to distributed artificial intelligence*, MIT Press, 1999.