Available online at www.sciencedirect.com

ScienceDirect

Engineering Applications of Artificial Intelligence 21 (2008) 183–199

Engineering Applications of ARTIFICIAL INTELLIGENCE

www.elsevier.com/locate/engappai

Enacting agent-based services for automated procurement

A. Giovannucci[a,*], J.A. Rodríguez-Aguilar[a], A. Reyes[b], F.X. Noria[b], Jesús Cerquides[c]

[a]*IIIA-CSIC Campus UAB, Bellaterra, Spain*
[b]*Intelligent Software Components, S.A. 08190 Sant Cugat del Vallès, Barcelona, Spain*
[c]*Department de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Barcelona, Spain*

Received 18 September 2006; accepted 10 April 2007
Available online 2 July 2007

Abstract

Negotiation events in industrial procurement involving multiple, highly customisable goods pose serious challenges to buying agents when trying to determine the best set of providing agents' offers. Typically, a buying agent's decision involves a large variety of constraints that may involve attributes of a very same item as well as attributes of different, multiple items. In this paper we present *iBundler*, an agent-aware service offered to buying agents to help them determine the optimal bundle of received offers based on their business rules. In this way, buying agents are relieved with the burden of solving too hard a problem and concentrate on strategic issues. *iBundler* is intended as a negotiation service for buying agents and as a winner determination service for reverse combinatorial auctions with side constraints. Furthermore, we assess the computational cost added by employing agent technology in the development of *iBundler* to characterise the type of negotiation scenarios that it can acceptably handle.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Negotiation; e-Commerce; Open agent systems; Scalability; Artificial intelligence; Agents; Integer programming; Optimization

1. Introduction

With the advent of the Internet, agent researchers envisioned a promising future to software agents in a large variety of e-commerce settings. Business automation, decision-making and enterprise integration have been widely claimed, among others, as suitable tasks for agents. Time has proven those expectations were too enthusiastic in general, but in our view there are indeed e-commerce scenarios where agent technology can prove valuable (Hohner et al., 2003). In particular, agent technology can contribute to the automation of complex tasks and to the assistance of parties involved in intricate decision-making processes in procurement scenarios. One particular, key procurement activity carried out by most companies concerns the negotiation of both direct and indirect goods and services.

Although negotiation is a key procurement mechanism, most agent-based services deployed have focused on infrastructure issues related to negotiation protocols and ontologies. Thus, the lack of agent-based decision support for trading agents that help improve current trading practices hinders the adoption of agent technology in procurement scenarios.

Furthermore, while a significant number of agent-based applications for electronic commerce have been presented to the agent community during the last years, little attention has been devoted to analysing the practical benefits and shortcomings of agent technology when applied to such domain. Little effort has been devoted to study the applicability of state-of-the-art agent technology to develop actual-world e-commerce applications. Thus, we believe that it is necessary to assess the computational cost added by agent technology in this type of applications so that we can diagnose the improvements required by state-of-the-art agent technology.

For this purpose we report on a case study that intends to shed some light on both matters. In this paper we fully describe *iBundler* (introduced in Giovannucci et al., 2004),

*Corresponding author.
E-mail addresses: andrea@iiia.csic.es (A. Giovannucci), jar@iiia.csic.es (J.A. Rodríguez-Aguilar), toni@isoco.com (A. Reyes), fxn@isoco.com (F.X. Noria), cerquide@maia.ub.es (J. Cerquides).

0952-1976/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2007.04.006

an agent-aware decision support service acting as a combinatorial negotiation solver (solving the winner determination problem) for both multi-item, multi-unit negotiations and auctions. Thus, the service can be employed by both buying agents and auctioneers in combinatorial negotiations and combinatorial reverse auctions (Sandholm et al., 2002), respectively. To the best of our knowledge, *iBundler* represents the first agent-aware service for multi-item, multi-unit negotiations. In fact, *iBundler* was designed to be employed as: (1) an open agent platform within the Agentcities. RDT[1] project that could be discovered, communicate, and offer services to any FIPA compliant agent (FIPA, http://www.fipa.org); (2) an agent façade to *Quotes* (Reyes-Moro et al., 2003), a commercial negotiation tool, to allow for the participation of third-party business agents in actual-world procurement events. In both cases, we study the computational cost of agent awareness for the *iBundler* negotiation service so that its users are aware of the type of negotiation scenarios that *iBundler* can acceptably handle when buying and providing agents are involved. This exercise has also included the determination of those general or domain-dependent measures that can help reduce the cost of the service. At this aim, we have measured the performance in time and memory of *iBundler* through a wide range of artificially generated negotiation scenarios. For each negotiation scenario we sampled at several stages both the time and memory that *iBundler* employed to handle it. We have interestingly observed that the management of ontologies is a rather delicate issue that actually causes a significant overload. Furthermore, we have also observed that the design of highly expressive, compact bidding languages can definitely help cut down the computational cost for any agent-aware negotiation service considering combinatorial scenarios.

The paper is organised as follows. Section 2 introduces the market scenario where buyers and traders are to negotiate, along with the requirements, preferences, and constraints they may need to express. Next, a formal model of the problem faced by the buyer (auctioneer) based on the description in Section 3 is provided. Thereafter, Section 4 details the computational realisation of the agent service as an agency. The description of the service focuses on the agency's architecture, the AUML specification of the interaction protocol offered by the service to trading agents, and ontological issues that needed to be considered in order to offer buying and providing agents the expressiveness required to enact their business rules. Next, Section 5 details the evaluation scenarios arranged to test *iBundler*, and presents and thoroughly discusses the tests' results. Finally, Section 6 summarises our contributions and draws conclusions on our evaluation.

[1]The Agentcities. RDT project's objectives were to create an on-line, distributed test-bed to explore and validate the potential of agent technology for future dynamic service environments (http://www.agentcities.org/EURTD).

2. Market scenario

Although the application of combinatorial auctions (CA) to e-procurement scenarios (particularly reverse auctions) may be thought as straightforward, the fact is that there are multiple new elements that need to be taken into consideration. These are new requirements explained by the nature of the process itself.

While in direct auctions, the items to be sold are physically concrete (they do not allow configuration), in a negotiation involving highly customisable goods, buyers need to express relations and constraints between attributes of different items. On the other hand, multiple sourcing is common practice, either for safety reasons or because offer aggregation is needed to cope with high-volume demands. This introduces the need to express constraints on providers and on the contracts they may be awarded. Not forgetting the provider side, providers may also impose constraints or conditions over their offers.

Consider a buyer intending to buy 200 chairs (any colour/model is fine) for the opening of a new restaurant, and at that aim we employ an e-procurement solution that launches a reverse auction. If we employ a state-of-the-art CA solver, a possible resolution might be to buy 199 chairs from provider A and 1 chair from provider B, simply because it is 0.1% cheaper and it was not possible to specify that in case of buying from more than one provider a minimum of 20 chairs purchase is required. On the other hand, the optimum solution might tell us to buy 50 blue chairs from provider A and 50 pink chairs from provider B. Why? Because although we had no preference over the chairs' colour, we could not specify that regarding the colour chosen all chairs must be of the same colour. Although simple, this example shows that without modelling natural constraints, solutions obtained are seen as mathematically optimal, but unrealistic.

Next, we identify the capabilities required by buyers in the above-outlined negotiation scenario to express their preferences:

(1) *Negotiate over multiple items*. A negotiation event is usually started with the preparation of a request for quotation (RFQ) form, which details the requirements (including attribute values as well as drawings and technical documentation) for the list of requested items (goods or services).

(2) *Offer aggregation*. An RFQ item can be multiply sourced (acquired from several providers), either because not a single provider can satisfy the whole demand or because of buyers' explicit constraints (see below).

(3) *Business sharing constraints*. Buyers might be interested to restrict the number of providers that may have each RFQ item awarded, either for security or strategic reasons. It is also common practice to constraint the contract volume a single provider may gain per item.

(4) *Constraints over single items*. Every RFQ item is described by a list of negotiable attributes. Since: (a) there exists a degree of flexibility in specifying each of these

attributes (e.g. several values are acceptable); and (b) multiple offers referring the very same item can be finally accepted; buyers need to impose constraints over attribute values. For instance, say that the deadline for the recep‘tion of item A is 2 weeks. Although items may arrive any given day within 2 weeks, once the first units arrive, the rest of units might be required to arrive no more than 3 days later.

(5) *Constraints over multiple items*. In daily industrial procurement, accepting certain configuration for one item might affect the configuration of a different item (e.g. to ensure compatibility between products). Hence, buyers need to express constraints and relationship between attributes of different RFQ items.

(6) *Specification of providers' capacities*. Buyers cannot risk to award contracts to providers beyond their capabilities. Towards this aim, they must require to have providers' capacities per item declared.

Analogously, next we detail the expressiveness of the bidding language required by providers:

(7) *Combinatorial offers*. Economy efficiency is enhanced if providers are allowed to offer (bid on) combination of goods. They might lower the price, or improve service assets if they achieve to get more business.

(8) *Multiple bids*. Providers might be interested in offering alternate conditions/configurations for the very same good, i.e., offering alternatives for a same request.

(9) *Multi-unit offering*. Each provider requires to specify his willingness to sell over/below a minimum/maximum number of units.

(10) *Homogeneous offers*. Combinatorial offering may produce inefficiencies when combined with multi-unit offering. Thus a provider may wind up with an award of a small number of units for a certain item, and a large number of units for a different item, being both part of the very same offer (e.g. 10 chairs and 200 tables). It is desirable for providers to be able to specify homogeneity with respect to the number of units for complementary items.

(11) *Packing constraints*. It is often not possible to serve an arbitrary number of units (e.g. a provider cannot sell 27 units because his items come in 25-unit packages). Thus, providers require to specify their packing sizes.

(12) *Complementary and exclusive offers*. Providers usually submit XOR bids, i.e., exclusive offers that cannot be simultaneously accepted. Also, they may wish to indicate that an offer is selected only if another offer is also selected. This type of bidding, hereafter referred to as AND bids, allows to express volume-based discounts (e.g. first 1000 units at 2.5 EUR p.u. and then 2 EUR each).

Although many more constraints and features might be considered, we believe these do address well the nature of the problem. Therefore, the numbered list above must be regarded as the list of requirements we would like the negotiation service proposed in this work to capture.

3. Formal model

In this section we provide a formal model of the problem faced by the buyer (auctioneer) based on the description in Section 2. But before, some definitions are in place.

Items. The buyer (auctioneer) has a vector of items $\Lambda = \langle \lambda_1, \ldots, \lambda_m \rangle$ standing for the list of requested items so that a buyer can negotiate over multiple items according to requirement 1 in Section 2. He specifies how many units of each item he wants $U = \langle u_1, \ldots, u_m \rangle, u_i \in \mathbb{R}^+$. He also specifies the minimum percentage of units of each item $M = \langle m_1, \ldots, m_m \rangle, m_i \in [0, 1]$, and the maximum percentage of units of each item $\bar{M} = \langle \bar{m}_1, \ldots, \bar{m}_m \rangle, \bar{m}_i \in [0, 1]$, $\bar{m}_i \geqslant m_i$, that can be allocated to a single provider. Furthermore, he specifies the minimum number of providers $S = \langle s_1, \ldots, s_m \rangle, s_i \in \mathbb{N}$, and the maximum number of providers $\bar{S} = \langle \bar{s}_1, \ldots, \bar{s}_m \rangle, \bar{s}_i \in \mathbb{N}, \bar{s}_i \geqslant s_i$, that can be simultaneously allocated each item. In this way the buyer can express his business sharing constraints as stated by requirement 3. Finally, a tuple of weights $W = \langle w_1, \ldots, w_m \rangle, 0 \leqslant w_i \leqslant 1$, contains the degree of importance assigned by the buyer to each item.

Item attributes. Given an item $\lambda_i \in \Lambda$, let $\langle a_{i_1}, \ldots, a_{i_k} \rangle$ denote its attributes. Thus, items may have multiple attributes as needed by requirement 1.

Providers' capacities. Let $\Pi = \langle \pi_1, \ldots, \pi_r \rangle$ be a tuple of providers. Given a provider $\pi_i \in \Pi$ the tuple $C^i = \langle c^i_1, \ldots, c^i_m \rangle$ stands for the minimum capacity of the provider, namely the minimum number of units of each item that the provider is capable of serving. Analogously, the tuple $\bar{C}^i = \langle \bar{c}^i_1, \ldots, \bar{c}^i_m \rangle$ stands for the maximum capacity of the provider, i.e. the maximum number of units of each item that the provider is capable of providing. Notice that in this way providers can state their capacities as needed by requirement 6.

Bid. The providers in Π submit a tuple of bids $B = \langle B^1, \ldots, B^n \rangle$. A bid is a tuple $B^j = \langle \Delta^j, P^j, M^j, \bar{M}^j, D^j \rangle$, where $\Delta^j = \langle \Delta^j_1, \ldots, \Delta^j_m \rangle$ are tuples of bid values per item, where $\Delta^j_i = \langle \delta^j_{i_1}, \ldots, \delta^j_{i_k} \rangle \in \mathbb{R}^k, 1 \leqslant i \leqslant m$, assigns values to the attributes of item λ_i; $P^j = \langle p^j_1, \ldots, p^j_m \rangle, p^j_i \in \mathbb{R}^+$, are the unitary prices per item; $M^j = \langle m^j_1, \ldots, m^j_m \rangle, m^j_i \in \mathbb{R}^+$, is the minimum number of units per item offered by the bid; $\bar{M}^j = \langle \bar{m}^j_1, \ldots, \bar{m}^j_m \rangle, \bar{m}^j_i \in \mathbb{R}^+, \bar{m}^j_i \geqslant m^j_i$, is the maximum number of units of each item offered by the bid; and $D^j = \langle d^j_1, \ldots, d^j_m \rangle$ are the *bucket* or *batch* increments in units for each item ranging from the minimum number of units offered up to the maximum number of units. Given a bid $B^j \in B$, we say that B^j does not offer item $\lambda_i \in \Lambda$ iff $m^j_i = \bar{m}^j_i = 0$. Notice that the way we define bid allows to express combinatorial bids over multiple items, multi-unit offering per item, and packing constraints as needed by requirements 9, 10, and 11 in Section 2.

In order to model homogeneity constraints, as needed by requirement 10, we define a function $h : B \to 2^{\Lambda}$. Given a bid $B^j \in B$, $h(B^j) = \{\lambda_{j_1}, \ldots, \lambda_{j_k}\}$ indicates that the bid is homogeneous with respect to the items in $h(B^j)$. In other words, if the buyer (auctioneer) picks up bid B^j the number of units allocated for the items in $h(B^j)$ must be equal.

Furthermore, in order to relate providers to their bids we define function $\rho : \Pi \times B \to \{0, 1\}$ such that $\rho(\pi_i, B^j) = 1$ indicates that provider π_i is the owner of bid B^j. This function satisfies the following properties:

- $\forall B^j \in B\ \exists \pi_i \in \Pi$ such that $\rho(\pi_i, B^j) = 1$, and
- given a bid $B^j \in B$ if $\exists \pi_i, \pi_k \in \Pi$ such that $\rho(\pi_i, B^j) = 1$ and $\rho(\pi_k, B^j) = 1$ then $\pi_i = \pi_k$.

The conditions above impose that each bid belongs to a single provider, but each provider may own multiple bids (as needed by requirement 8 above).

XOR bids. Let $xor : 2^B \to \{0, 1\}$ be a function that defines whether a subset of bids must be considered as an XOR bid. Only bids owned by the very same provider can be part of an XOR bid. More formally $xor(\mathscr{B}) = 1 \Rightarrow \exists! \pi \in \Pi$ such that $\rho(\pi, B^i) = 1\ \forall B^i \in \mathscr{B}$. Thus, f.i. if $\exists B^j, B^k \in B\ xor(\{B^j, B^k\}) = 1$ both bids are mutually exclusive, and thus cannot be simultaneously selected by the buyer.

AND bids. Let $and : 2^B \to \{0, 1\}$ be a function that defines whether an ordered tuple of bids must be considered as an AND bid. Thus, given an ordered tuple of bids $\langle B^{j_1}, \ldots, B^{j_k} \rangle$ such that $and(\langle B^{j_1} \ldots B^{j_k} \rangle) = 1$ then the buyer can only select a bid $B^{j_i}, 1 < i \geqslant k$, whenever $B^{j_1}, \ldots, B^{j_{i-1}}$ are also selected. Furthermore, all bids in an AND bid belong to the very same provider. Put formally, $and(\mathscr{B}) = 1 \Rightarrow \exists!\ \pi \in \Pi$ such that $\rho(\pi, B^i) = 1\ \forall B^i \in \mathscr{B}$. AND bids are intended to provide the means for the buyer to express volume-based discounts. However, they should be regarded as a generalisation of the bidding via price-quantity graphs in (Sandholm, 2002b).

Notice that XOR bids and AND bids capture the expressiveness needed by requirement 12. While XOR bids allow to express exclusive offers, AND bids allow to express complementary offers.

Based on the definitions above we can formally introduce the decision problem to be solved to provide support to the buyer (auctioneer):

(*Multi-attribute, multi-unit combinatorial reverse auction*). The multi-attribute, multi-unit combinatorial reverse auction winner determination problem (MMCRAWDP) accounts for the maximisation for the following expression:

$$\sum_{j=1}^{n} y_j \cdot \sum_{i=1}^{m} w_i \cdot V_i(q_i^j, p_i^j, \Delta_i^j), \tag{1}$$

subject to the following constraints:

1. $q_i^j \in 0 \cup [m_i^j, \bar{m}_i^j]$. This constraint forces that when bid B^j is selected as a winning bid, the allocated number of units of each item q_i^j has to fit between the minimum and maximum number of units offered by the provider.
2. $q_i^j \bmod d_i^j = 0$. The number of allocated units q_i^j to a bid B^j for item λ_i must be a multiple of the batch d_i^j specified by the bid.
3. $\sum_{j=1}^{n} q_i^j = u_i$ (there is no free disposal). The total number of units allocated for each item must equal the number of units requested by the buyer.
4. $\forall \pi_k \in \Pi q_i^j \cdot \rho(\pi_k, B^j) \in \{0\} \cup [c_i^k, \bar{c}_i^k]$. For each item, the number of units allocated to a provider cannot exceed his capacities.
5. $\forall \pi_k \in \Pi q_i^j \cdot \rho(\pi_k, B^j) \in \{0\} \cup [m_i \cdot u_i, \bar{m}_i \cdot u_i]$. The total number of units allocated per provider cannot exceed or be below the maximum and minimum percentages that can be allocated per provider specified by the buyer.
6. $\forall \lambda_{j_t}, t \in h(B^j)\ \ q_i^j = q_t^j$. For homogeneous bids, the number of units allocated to the items declared homogeneous must be the same.
7. $\forall \lambda_i \in \Lambda \sum_{k=1}^{r} x_i^k \in [s_i, \bar{s}_i]$. The number of providers to be awarded each item cannot exceed or be below the maximum and minimum number of total providers specified by the buyer.
8. $and\ (\langle B^{j_1}, \ldots, B^{j_k} \rangle) = 1 \Rightarrow y^{j_1} \geqslant \cdots \geqslant y^{j_k}$. Bids being part of an AND bid can only be selected if the bids preceding them in the AND bid are selected too.
9. $\forall B' \subseteq B$ such that $xor(B') = 1 \sum_{B^j \in B'} y^j \leqslant 1$. XOR bids cannot be jointly selected.
10. $a \cdot v_{i,l} + b \geqslant \delta_{i,l}^j \geqslant a' \cdot v_{i,l} + b'$ where $a, b, a', b' \in \mathbb{R}$. Intra-item constraints are modelled through this expression. It indicates that only those bids whose value for the attribute item related to the decision variable that satisfy the expression can be selected.
11. $c \cdot v_{i,l} + d \geqslant v_{j,k} \geqslant c' \cdot v_{i,l} + d'$ where $c, d, c', d' \in \mathbb{R}$. Inter-item constraints are modelled through this expression. It puts into relation decision variables of attributes belonging to different items.

Here

- $y_j \in \{0, 1\}, 1 \leqslant j \leqslant n$, are decision variables for the bids in B;
- $x_i^k \in \{0, 1\}, 1 \leqslant i \leqslant m, 1 \leqslant k \leqslant r$, are decision variables to decide whether provider π_k is selected for item λ_i;
- $q_i^j \in \mathbb{N} \cup \{0\}, 1 \leqslant j \leqslant n, 1 \leqslant i \leqslant m$, are decision variables on the number of units to select from B^j for item λ_i;
- $V_i : \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^{i_k} \to \mathbb{R}, 1 \leqslant i \leqslant m$, are the bid valuation functions for each item; and
- $v_{i,l}$ stands for a decision variable for the value of attribute a_l of item λ_i.
- $a, b, a', b', c, c', d, d'$ are buyer-defined constant values.

Observe that the constraints coming along Eq. (1) attempt at capturing the requirements in Section 2 as follows: constraint 1 makes possible multi-unit offering (requirement 9); constraint 2 ensures that packing constraints are fulfilled (requirement 11); constraint 3 makes possible that several offers can be aggregated to fulfil the demand (requirement 2); constraint 4 exploits the specification of providers' capacities (requirement 6); constraints 5 and 7 ensure that business sharing constraints (requirements 5 and 7) are not violated; constraint 6 ensures that homogeneous offers are selected according to requirement 10; constraint 8 and 9 ensure that complementary and exclusive offers are selected as expected (requirement 12); constraints 10 and 11 map the constraints over single items and multiple items in requirements 4 and 5.

Finally, there are several aspects that make our model differ from related work. Firstly, traditionally all CA models assume that the buyer (auctioneer) equally prefers all items. Such constraint is not considered in our model, allowing the buyer to express his preferences over items. Secondly, multi-attribute auctions and CAs with side constraints have been separately dealt with. On the one hand, Bichler (Bichler and Kalagnanam, 2005) extensively deals with multi-attribute auctions, including a rich bidding language. On the other hand, Sandholm et al. (2002) focus on multi-item, multi-unit CAs with side constraints where items are not multi-attribute. We have attempted at formulating a model which unites both. Lastly, to the best of our knowledge neither inter-item nor intra-item constraints have been dealt with in the literature at the attribute level (Kalagnanam and Parkes, 2004) though they help us better cope with multiple sourcing scenarios.

4. Implementation

This section details the realisation of the agent service as an agency. Firstly, we present the implementation of the winner determination problem with side constraints as the core of the service. Secondly, we describe the architecture of the *iBundler* agency, along with a description of the protocols and the ontology employed by its agents.

4.1. Winner determination

Consider the problem faced by a buying agent aiming at choosing the optimal set of offers sent over by providing agents taking into account the features of the negotiation scenario described in Section 2. For this purpose he must solve the optimisation problem posed by Eq. (1) in Section 3. The problem is essentially an extension of the CA problem in the sense that it implements a larger number of constraints and supports richer bidding models. The CA problem is known to be NP-complete, and consequently solving methods are of crucial importance.

In general terms, we identify three main approaches that have been followed in the literature to fight the complexity of this problem:

- As reported in Kelly and Steinberg (2000), attempts to make the combinatorial auction design problem tractable through specific restrictions on the bidding mechanism have taken the approach of considering specialised structures that are amenable to analysis. But such restrictions violate the principle of allowing arbitrary bidding, and thus may lead to reductions in the economic outcome.
- A second approach sacrifices optimality by employing approximate algorithms (Hoos and Boutilier, 2000; Zurel and Nissam, 2001). However, and due of the intended actual-world usage of our service, it is difficult to accept the notion of sub-optimality.
- A third approach consists in employing an exact or complete algorithm that guarantees the global optimal solution if this exists. Although theoretically impractical, the fact is that effective complete algorithms for the CA problem have been developed.

Many of the works reviewed in the literature adopt global optimal algorithms as a solution to the CA because of the drawbacks pointed out for incomplete methods. Basically two approaches have been followed: traditional Operations Research (OR) algorithms and new problem-specific algorithms (Sandholm, 2002a; Fujishima et al., 1999). It is always an interesting exercise to study the nature of the problem in order to develop problem-specific algorithms that exploit problem features to achieve effective search reduction. However, as indicated in Holte (2001) and Rothkopt et al. (1995), the CA problem is an instance of the multi-dimensional knapsack problem MDKP, a mixed integer program well studied by the OR literature. It is not surprising, as reported in Andersson et al. (2000), that many of the main features of these problem-specific, new algorithms are re-discoveries of traditional methods in the OR community. In fact, our formulation of the problem can be regarded as similar to the binary multi-unit combinatorial reverse auction winner determination problem in Sandholm et al. (2002) with side constraints Sandholm and Suri (2001). Besides, expressing the problem as a mixed integer programming problem with side constraints enables its resolution by standard algorithms and commercially available, thoroughly debugged and optimised software which have shown to perform satisfactorily for large instances of the CA problem.

With these considerations in mind, the core of our service has been implemented as a mixed integer programming problem. We have implemented two versions: a version using ILOG CPLEX 9.0; and another version using iSOCO's Java MIP modeller that integrates the GLPK library (Makhorin, 2001). In both cases it takes the shape of a software component. Hereafter, we shall refer to this component as the *iBundler* solver. Therefore, this component solves the maximisation problem in Eq. (1)

4.2. Architecture

The *iBundler* service has been implemented as an agency composed of agents and software components that cooperatively interact to offer a decision-support service for highly-constrained negotiation scenarios. *iBundler* can act as a combinatorial negotiation solver for both multi-item, multi-unit negotiations and auctions. Thus, the service can

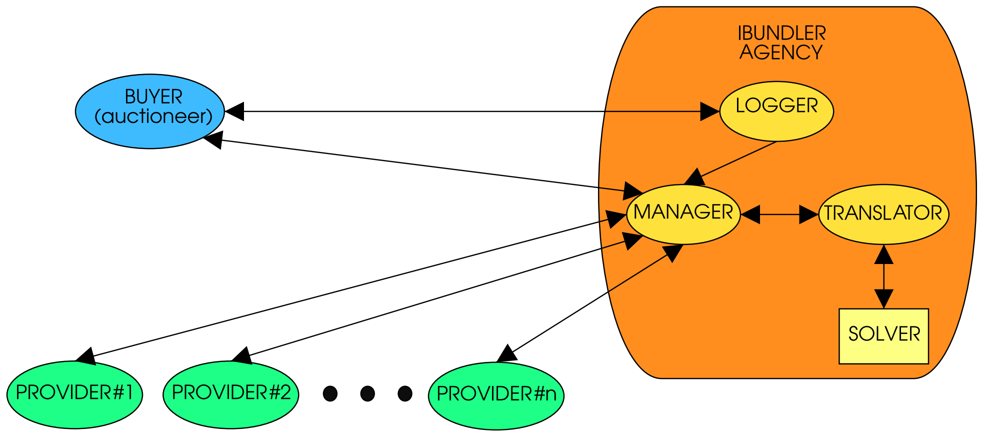


Fig. 1. Architecture of the *iBundler* agency.

be employed by both negotiating agents and auctioneers in CA. Fig. 1 depicts the components of the agency, along with the fundamental connections of buyers and providers with the service. Next, we make explicit the main functionality of its members:

Logger agent. It represents the interface of the *iBundler* agency to the world. The Logger agent is in charge of facilitating registration and deregistration with the *iBundler* service to users (both buyers and providers) as well as their subsequent access to the service via log in and log out.

Manager agent. Agent devoted to providing the solution of the problem of choosing the set of bids that best matches a user's requirements. There exists a single Manager agent per user (buying agent or auctioneer), created by the Logger agent, offering the following services: brokering service to forward buying agents' requirements (RFQs) to selected providing agents capable of fulfilling them; collection of bids; winner determination in a combinatorial negotiation/auction; award of contracts on behalf of buying agents. Furthermore, the manager agent is also responsible for: bundling each RFQ and its bids into a negotiation problem in FIPA-compliant (FIPA,, http://www.fipa.org) format to be conveyed to the Translator agent; and to extract the solution to the negotiation problem handled back by the Translator agent. Observe that Fig. 1 shows the interplay of buying and providing agents with the Manager as the sole access point to the *iBundler* agency.

Translator agent. It creates an XML document representing the negotiation problem in a format understandable by the Solver departing from the FIPA-compliant description received from the Manager. It also translates the solution returned by the Solver into an object of the ontology employed by user agents. It is the bridge between the language spoken by user agents and the language spoken by solver.

Solver component. The *iBundler* component itself extended with the offering of an XML language for expressing offers, constraints, and requirements. The XML specification is parsed into an MIP formulation and solved using available MIP solvers as described in Section 4.1. Therefore, this component solves the optimisation problem posed by Eq. (1) in Section 3.

Our design manages to separate concerns among the three members of the agency. On the one hand, the Manager is strictly devoted to coordination. It represents the façade of the service. Besides, since every negotiation requested by a buyer makes the agency create an instance of the *Manager*, the service can cope with asynchronous and multiple accesses to the service. The Translator agent is in charge of relieving both Managers and Solver from the burden of translating FIPA-compliant specifications into the language required by Solver. Notice that the fact of having only one Translator agent represents a bottle-neck in the overall process when many buyers access the service concurrently. Such limitation could be overcome by creating multiple instances of Translator Agents and Solvers on different machines. However, in this work we focused on the service performances in managing big size negotiation scenarios, not on multiple concurrent accesses to the service. We leave such issue as a possible future development.

To implement the *iBundler* agency we used the following technologies: JADE (Caire, 2002) as the software tool to implement agents, and as the platform where the agency resides (connected to the Agentcities network as a node); Tomcat (http://jakarta.apache.org/tomcat/) as J2EE server to build web interfaces for human traders; and FIPA in building agents, messages, ontology and protocols.

In the following section, we make explicit the interplay of protocols involved in the whole interaction to compose the protocol of the service.

4.3. Interaction protocol

The type of negotiation protocol we consider is an extension of a Contract Net protocol (FIPA, 2003) between a buying agent and some providing agents that involves several interaction protocols involving the agents introduced in Section 4.2. Fig. 2 depicts the interaction

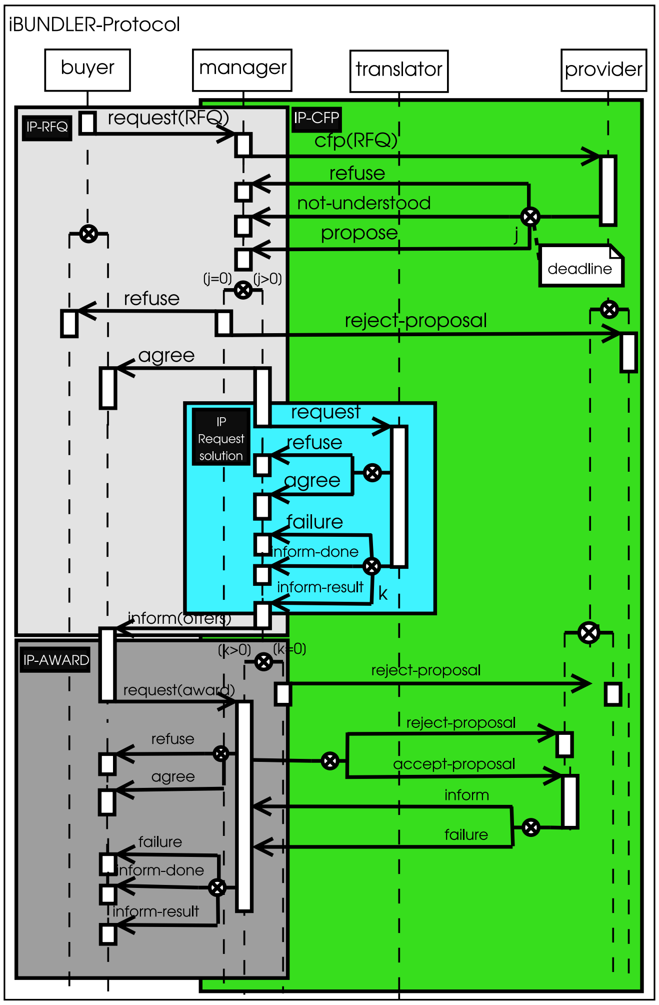


Fig. 2. *iBundler* FIPA interaction protocol.

protocols involved in the interplay of buyers and provides with *iBundler*. Notice that the interaction protocols are expressed in AUML (Agent Unified Modelling Language) (Odell et al., 2000) following the FIPA interaction protocol library specification compiled in (FIPA, 2003).

Observe that the specification in Fig. 2 involves four roles, namely *buyer*, *manager*, *translator*, and *provider*. Whereas multiple agents can act as providers, the remaining roles can be uniquely adopted by a single agent each. Notice too that the *iBundler* interaction protocol is composed of several interleaved interaction protocols, namely:

IP-RFQ. Held between a buyer and the manager agent created by the Logger agent after registration. The buyer delivers an RFQ to his manager agent requesting to obtain the optimal set of offers from the available providers. In case it is not possible to obtain a solution to the problem, the received response is an empty bid set.

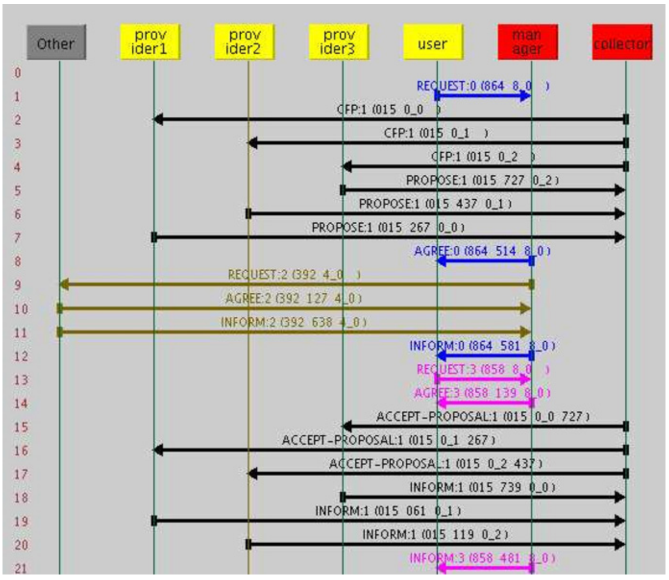
IP-CFP. Prior to delivering the optimal set of offers, the manager interacts with the available providers to request their offers under the rules of this CFP interaction protocol. If no offers are received the manager refuses to deliver the optimal set of offers in the context of the IP-RFQ interaction protocol. Otherwise, the manager agrees on providing the service and proceeds ahead by starting out an instance of the IP-Request-Solution interaction protocol. The protocol winds up with the notification of contract awards to selected providers according to the buyer's decision. In the case in which no optimal solution could be found, the buyer is sent an empty bid set and the IP-CFP protocol is ended communicating a Reject-Proposal to each provider involved. Notice that the manager mediates between buyer and providers.

IP-Request solution. This interaction protocol held between the manager and the translator agent within the *iBundler* agency aims at calculating the optimal set of offers considering the offers submitted by providers, along with the buyer's requirements and constraints. The result delivered by the translator is further conveyed by the manager to the buyer in the context of the interleaved IP-RFQ interaction protocol.

IP-AWARD. At the end of the IP-RFQ interaction protocol the buyer obtains the optimal set of offers. He may request also to receive all offers. Thereafter, if the buyer received a non-empty optimal set of offers ($k>0$ in Fig. 2), the buyer initiates the IP-AWARD interaction protocol in order to request the manager to award contracts to selected providers. Observe that the contract award distribution is autonomously composed by the buyer, and thus the buyer may decide to either ignore or alter the optimal set.

Fig. 3 illustrates an actual interaction of a buying agent with *iBundler* when conducting a negotiation with a limited number (three) of providing agents. Notice that the interactions depicted among buying agents, *iBundler*, and providing agents have been captured with the aid of the Sniffer tool (TILAB, 2005) provided by Jade.

The visualisation depicted by Jade's Sniffer is analogous to a UML sequence diagram, though the exchange of



```
0
1   REQUEST:0 (864  8_0  )
2   CFP:1 (015  0_0  )
3   CFP:1 (015  0_1  )
4   CFP:1 (015  0_2  )
5   PROPOSE:1 (015  727  0_2 )
6   PROPOSE:1 (015  437  0_1 )
7   PROPOSE:1 (015  267  0_0 )
8   AGREE:0 (864  514  8_0 )
9   REQUEST:2 (392  4_0  )
10  AGREE:2 (392  127  4_0 )
11  INFORM:2 (392  638  4_0 )
12  INFORM:0 (864  581  8_0 )
13  REQUEST:3 (858  8_0  )
14  AGREE:3 (858  139  8_0 )
15  ACCEPT-PROPOSAL:1 (015  0_0  727 )
16  ACCEPT-PROPOSAL:1 (015  0_1  267 )
17  ACCEPT-PROPOSAL:1 (015  0_2  437 )
18  INFORM:1 (015  739  0_0 )
19  INFORM:1 (015  061  0_1 )
20  INFORM:1 (015  119  0_2 )
21  INFORM:3 (858  481  8_0 )
```

Fig. 3. Monitoring the *iBundler* interaction protocol.

messages occurs between agents instead of objects. Furthermore, each interaction is assigned a step number corresponding to its position within the ordered sequence of messages it belongs to (as shown along the left-hand side of the central picture).

The servicing of *iBundler* proceeds as follows. At step 1, the buying agent (labelled as user in Fig. 2) sends his RFQ to the manager agent (manager) in the *iBundler* agency. At that point, the manager spawns an auxiliary agent, the so-called collector agent (collector), and delegates to him the collection of bids from providing agents. Once created, the collector requests bids from providing agents provider1, provider2 and provider3 by starting CFP (call for proposal) FIPA protocols (steps 2–4) that include the RFQ. Notice though that such CFP filters out the buying agent's constraints enclosed in the RFQ to hide them away from providers. These subsequently submit their bids to the collector via propose performatives (steps 5–7). Once bids are collected, the manager constructs the combinatorial problem to be solved from the RFQ submitted by the buying agent and all bids collected from providers. Furthermore, he agrees on providing an optimal solution to the buying agent (step 8). Next, the manager asks the translator agent (translator) for a solution to the combinatorial problem (step 9). Upon reception, the translator translates the combinatorial problem into an XML-based problem specification which is shipped to the Solver component. It solves the optimisation problem described in Section 3 and returns the optimal solution to the translator as an XML-based document, so that it can be forwarded by the translator as a FIPA message to the manager (step 11). The optimal solution is finally sent over to the buying agent (step 12) so that he can employ it to decide which providers to award a contract, and for which items and units. The buyer's decision is made available to the manager (step 13), who requests the collector to award the contract to the selected providers, thus terminating the CFP protocol started out at step 2. Finally, the manager acknowledges the buyer that the contract has been indeed awarded to the providers, and on the terms he selected.

Notice that the manager creates the collector to ease implementation—since it is rather complex to extend the FIPA-compliant protocols offered by Jade to take advantage of FIPA-compliant protocols as offered by Jade, and to better control synchronisation over messages exchanged among all agents involved in the interaction with the *iBundler* service.

To summarise, the usage of the *iBundler* service requires the interleaving of several protocols, namely (agents involved in parentheses): (1) request for optimal solution for an RFQ (buyer and manager—messages 1, 8, 12); (2) collection of bids and awards (collector and providers—messages 2–7 and 15–20); (3) request for optimal solution for RFQ and collected bids (manager and translator—messages 9–11); (4) request for contract award (buyer and manager—messages 13, 14, 21).

4.4. Ontology

Although research on automated negotiation in multi-agent systems has concentrated on the design of negotiation protocols and their associated strategies, ontological aspects of negotiation protocols have recently started to attract researchers' attention (see Tamma et al., 2002; Tamma, 2003 and the results of the ADMIT project Agentcities Consortium, 2002). In Tamma et al. (2002) and Tamma (2003)) we find an ontological approach to automated negotiation founded on the following concepts: *negotiation protocol* (rules followed by participants during a negotiation process), *party* (participants, be them either human agents, software agents or even organisations of agents), process (way to reach an agreement on some issue by modifying negotiation attributes), (negotiation) *object*, *offer* (possible combination of values associated to the negotiation attributes which represent an expression of will), *negotiation rule* (set of rules that govern a specific negotiation protocol). Although satisfactory enough for most concepts, particularly as to negotiation protocols regarded as processes and rules, in this work we had to enrich the concepts of *offer* and *object* in order to accommodate the expressiveness required by the actual-world constraints described in Section 2 for bids and RFQs, respectively. To the best of our knowledge, no ontology defined in prior work allows the expressiveness that buying and providing agents require. In other words, there is no adequate ontology for multi-item, multi-unit combinatorial reverse auctions with side constraints. Thus we had to define an *ad hoc* ontology for the *iBundler* service.

The ontology has been defined with the aid of *Protege 2000* (Protege, 2005). Furthermore, the conversion from ontological objects to Java classes is realised via the *beangenerator* Protege 2000 plug-in (Acklin, 2005). The automatically generated Java classes fulfil with the JADE specification in Caire (2002).

Figs. 4–7 provide graphical representations (as shown by the Ontoviz Protégé plug-in) of the core concepts in the *iBundler* ontology, namely, and respectively, the *RFQ*, *ProviderResponse*, *Problem*, and *Solution* concepts. Table 1 lists the sub-protocols in the *iBundler* interaction protocol depicted in Fig. 2 wherein such concepts are employed, along with the concrete messages conveying them.

The *RFQ* concept is employed by buying agents to express their requests for bids. Fig. 4 shows that an RFQ is composed of a sequence of *Request* concepts, one per requested item. In this way, we allow a buying agent to negotiate over multiple items to fulfil requirement 1 in Section 2. A sequence of global constraints (*GlobalConstraint* concept) relating separate, requested items may be part of an RFQ. There are two types of *GlobalConstraint* concepts: constraints that allow to express linear relationships between different attributes of the very same or separate item(s) (*AttributeRelation* concept) and constraints on the values of an item's attribute (AttributeVariation

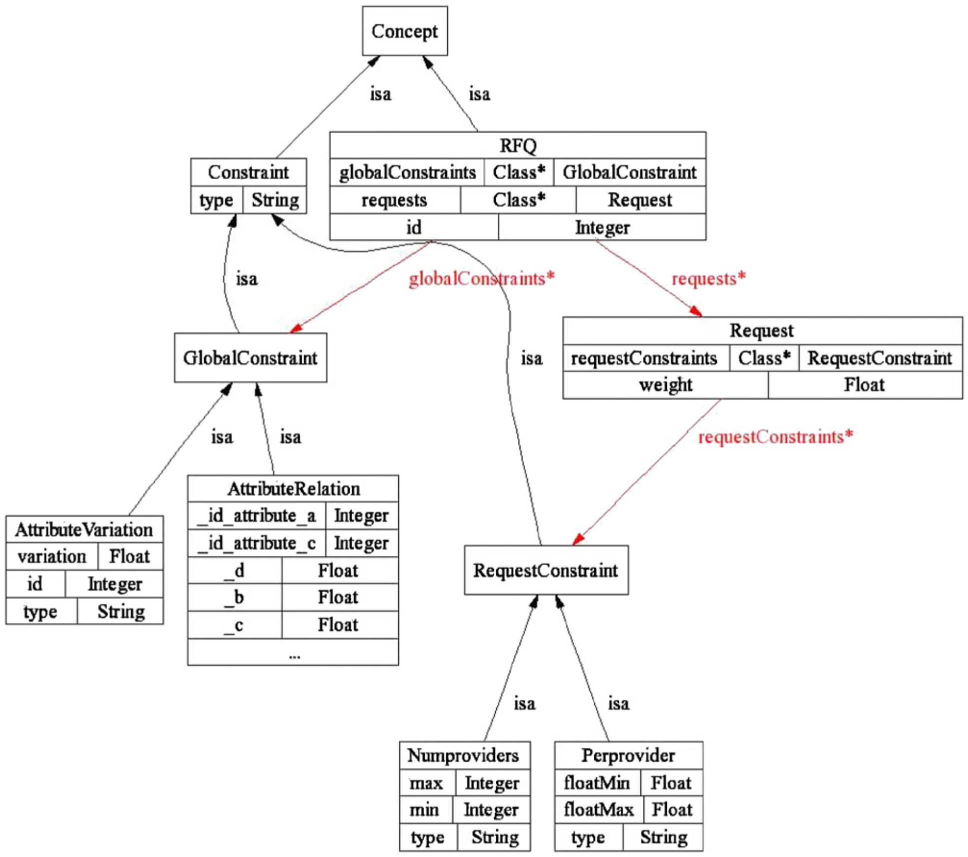


Fig. 4. RFQ concept representation.

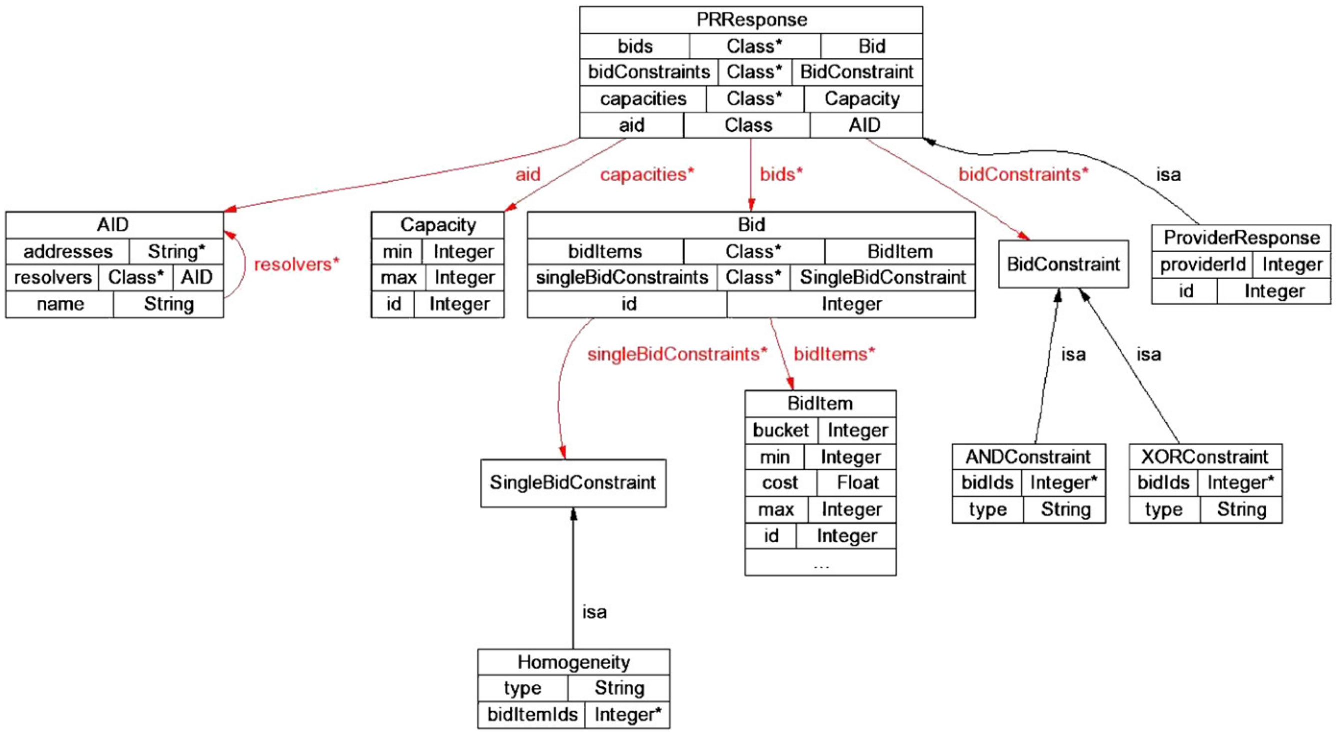


Fig. 5. Bid concept representation.

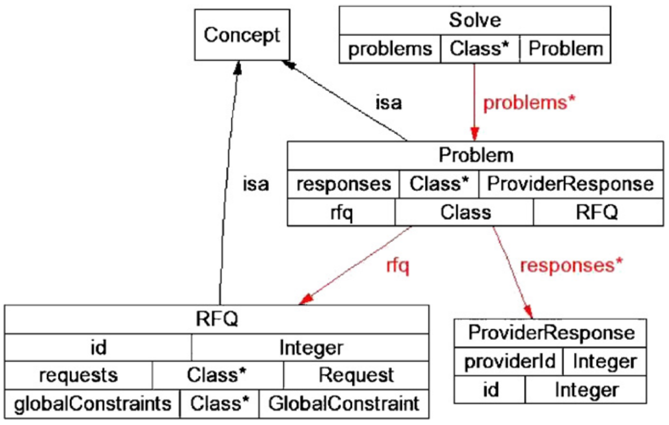

Concept
Solve
problems | Class* | Problem
isa
problems*
Problem
responses | Class* | ProviderResponse
rfq | Class | RFQ
isa
rfq
responses*
RFQ
id | Integer
requests | Class* | Request
globalConstraints | Class* | GlobalConstraint
ProviderResponse
providerId | Integer
id | Integer

Fig. 6. Problem concept representation.

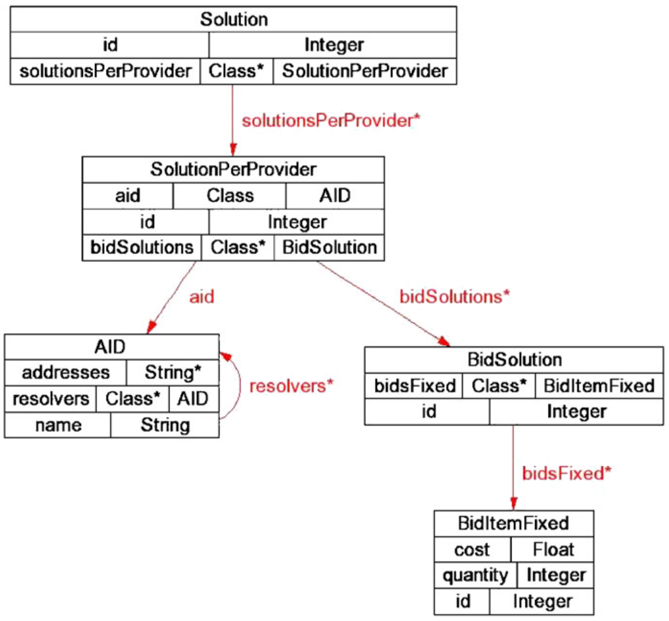


Fig. 7. Solution concept representation.

Table 1
Use of ontological concepts in Fig. 2

Concept	Sub-protocol	Message
RFQ	*IP-RFQ*	*request*(*RFQ*)
ProviderResponse	*IP-CFP*	*propose*(*offer*)
Problem	*IP-Request-Solution*	*request*(*problem*)
Solution	*IP-Request-Solution*	*inform-result*
	IP-RFQ	*inform*(*offers*)

concept). In this way, we obtain the expressiveness needed by requirements 4 and 5. A sequence of constraints on individual items (*RequestConstraint* concept) may be also part of an RFQ. Constraints on individual items can serve to limit the range of providers (*NumProviders* concept) to which the item can be awarded or the range of percentage of units to be awarded to the very same provider (*PerProvider* concept). These concepts provide the expressiveness needed by requirement 3.

Notice that all the constraints specified in an *RFQ* stand for the buyer's business rules.

On the provider side, providing agents express their offers in terms of the *ProviderResponse* concept, which in turn is composed of several elements: a list of *Bid* concepts (each *Bid* allows to express a bid per either a single requested item or a bundle of items) (and so combinatorial bids are possible as needed by requirement 7 in Section 2); constraints on the production/servicing capabilities of the bidding provider (*Capacity* concept) (to capture the needs of requirement 6 in Section 2); constraints on bundles of bids formulated with the *BidConstraint* concept (each *BidConstraint* in turn can be of exclusive –xor– or volume-based discount type –and–, corresponding respectively to the *XOR* and *AND* concepts) that allows to express the relationships of requirement 12 in Section 2; and their capacities through a sequence of *Capacity* concepts as needed by requirement 6. Whereas constraints on bundles of bids put into relation separate bids, constraints on individual bids (expressed as *SingleBidConstraint* concepts) allow to relate the values offered for separate items within the very same bid. As an example, homogeneity constraints can be declared by providers within some bid to make buyers aware that the quantity of items they can select per item must be the same, or else the provider will not concede his bid. Such constraint maps to the *Homogeneity* concept, a particular type of *SingleBidConstraint*. The *Homogeneity* concept allows to express homogeneous offers as needed by requirement 10.

Notice that since a *ProviderResponse* concept is composed of a list of *Bid* concepts, each provider is allowed to submit multiple bids as needed by requirement 8 in Section 2. Furthermore, a *Bid* concept is composed of a sequence of *BidItem* concepts, each one representing a provider's offer per item. Each *BidItem* concept allows a provider to express his multi-unit offering, as needed by requirement 9, and his packing constraints, as needed by requirement 11.

It is important to notice that the concepts defined so far in our ontology ensure, as mentioned above, the expressiveness needed by the requirements in Section 2.

Once the manager collects all offers submitted by providers, he wraps up the *RFQ* concept as received from the buyer along with the offers as *ProviderResponse* concepts to compose the negotiation problem to be solved by the *Solver* component. The resulting concept, *Problem*, is depicted in Fig. 6.

Finally, the solution produced by the Solver component is transformed by the translator agent into a *Solution* concept (see Fig. 7) that is handed over to the manager. The *Solution* concept contains the specification of the optimal set of offers calculated by Solver. Thus *Solution* contains a list of *SolutionPerProvider* concepts, each one containing the bids selected in the optimal bid set per provider, as a list of *BidSolution* concepts, along with the provider's agent identifier, as an *AID* concept. Each

BidSolution in turn is composed of a list of *BidItemFixed* concepts containing the number of units selected per bid along with the bid's total cost.

So far we have concentrated on ontological concepts referring to entities with a complex structure that can be defined in terms of slots. Hereafter we shall draw our attention to agent actions, i.e. the special concepts that indicate actions that can be performed by agents in the *iBundler* agency, as well as buyers and providers.

Thus, the Logger agent offers the services associated to the following actions:

Login. Action requested by trading agents when logging in with the *iBundler* agency.

Logout. Action requested by trading agents when logging out of the *iBundler* agency.

Register. Action requested by trading agents when signing for the *iBundler* agency. They must provide information about themselves. At the end of the registration, the Logger provides them with a user name and a password.

Unregister. Action requested by trading agents when unregistering with the *iBundler* agency. They must provide information about themselves. All the brokering information associated to them is erased by the Logger.

As to the manager, it offers four core services via the following actions:

GetAllBids. The buyer specifies an RFQ along with a list of providers. The manager agent forwards the query to all the providers and delivers back all the responses to the buyer.

Solve. The buyer sends to the manager an *RFQ* along with a list of *ProviderResponse* concepts representing providers' offers. The manager composes a *Problem* out of the *RFQ* and *ProviderResponses* to subsequently request the translator agent for a *Solution*. In this way, the buyer is relieved from the intricate construction of a *Problem* involving the creation of crossed references between *RFQ* and the *Bid* concepts in each *ProviderResponse*. Once the *Solution* is received by the manager it is forwarded to the buyer.

Manage. The buyer sends to the manager an *RFQ* along with a list of providers. The manager sends a filtered version of the *RFQ* (removing the buyer's private constraints) to available providers, collects all their offers, constructs a *Problem* to ask the translator for a *Solution*, which is conveyed to the buyer once calculated.

Buy. The buyer constructs a *Solution* concept and subsequently asks the manager to the bids and providers in the list of *SolutionsPerProvider*. Notice that the buyer may employ the same *Solution* concept recommended by the manager as an optimal solution.

Finally, providers do offer their services through the following actions:

RequestForQuotations. Request for offers received from the manager for a filtered version of the *RFQ* sent by the buyer.

BuySolution. Order to buy selected offers received from the manager.

5. On the cost of agent-awareness

The applicability analysis of agent technology in the literature primarily focuses on scalability issues as robustness, system performance with large populations of agents and ontology engineering. Brazier et al. (2001) address the problem of scalability in naming services and location services. Besides, they analyse the concept of scalability in multi agent systems (MAS) and discuss scalability for many existing multi-agent frameworks. Deters (2001) studies the problems derived from large number of agents running in a MAS, agent resource consumption, the exchange of great number of messages, identifying agent hosting and message routing as bottle-necks. Furthermore, he performs some scalability experiments. An important result in Deters (2001) is that the main deficiencies of JESS (http://herzberg.ca.sandia.gov/jess/) derive from serialisation processes. Kahn investigates how timing of sequential agent registration and lookup varies as the total number of registered agents increases in COABS (Kahn and Della Torre Cicalese, 2003). The works in Klein et al. (2003)) and Fedoruk and Deters (2002) analyse robustness and fault tolerance, whereas Yoo (2002) exemplifies ad hoc, domain-dependent agent technology scaling techniques. On the other hand, the literature on ontology scalability focuses on three major issues: the size of ontology contents, the complexity of ontology construction and knowledge reusability (Jarrar and Meersman, 2002; Wache et al., 2004). In particular, Jarrar states that experience shows that "unscalable solutions emerging from academic research often fails at the industrial level" (Jarrar and Meersman, 2002).

Thus, we believe that it is an urging necessity to report on practical deployments of actual-world agent-based applications in order to: (1) progressively derive best methodological practices; and (2) assess the improvements required by state-of-the-art agent technologies to be adopted at industry level. Particularly since much of the research effort on agent technology does not consider the application of widely employed agent frameworks and programming tools to real-world problems.

We consider *iBundler* as representative of the main trends on the state-of-the-art agent programming tools and platforms. Firstly, because it is based on the FIPA specification standard, that is surely the most widely adopted by the agent community.[2] Secondly, the considerations emerging from the experiments derived in this paper are related to the FIPA nature of the agent platform, not to a particular JADE implementation. Thus, the results in Section 5 are not limited to the JADE framework, being valid for all the FIPA-compliant agent frameworks.

[2]OGM (www.ogm.org) is another standardisation effort based on CORBA IDL interface. This solution is efficient for agent migration and client-server applications, but less suitable than FIPA-compliant platforms for peer-to-peer applications. For an interesting comparison refer to OMG and FIPA standardisation for agent technology: competition or convergence? (1999).

In this section we detail the way we conducted our evaluation. Firstly, in Section 5.1 we describe how to generate artificial negotiation scenarios for testing purposes. Next, in Section 5.2 we detail the different stages considered through our evaluation process. In Section 5.3, we analyse time performance, and finally, in Section 5.4, the memory use through all the evaluation stages.

In order to run our tests we employed the following technology: a PC with a Pentium IV processor, 3.1 Ghz, 1 GB RAM running a Linux Debian (kernel v.2.6) operating system (http://www.debian.org); Java SDK 1.4.2.04 (http://java.sun.com); JADE v.2.6; and ILOG CPLEX 9.0 (http://www.ilog.com).

5.1. *Artificial negotiation scenarios*

In order to evaluate the agent service performance, the time needed by *iBundler* to receive an RFQ from a *Buyer* agent and to collect the different bids from providers is considered of no interest. Because they depend on some uncontrolled variables (e.g. the time needed by providers to send their bids and the network delay). Thus, our evaluation starts from the moment at which all the required data (RFQ and bids) are available to the *Manager* agent. We tried to simulate such an ideal situation generating multiple data sets in separate files, each one standing for a different input negotiation problem composed of FIPA messages, each one containing both an RFQ and the bids received as a response to this. In this way we can use the file stream as if it was the incoming message stream, and perform all the subsequent message manipulation as if the message had been received from a socket.

Another important consideration has to do with the way we sampled time and memory. We established checkpoints through the process carried out by *iBundler* when solving a negotiation problem. Such checkpoints partition the process into several stages. We observed time and memory at the beginning and at the end of these stages.

In order to automate the testing it was necessary to develop a generator of artificial negotiation scenarios involving multiple units of multiple items. The generator is fed with mean and variance values for the following parameters: *number of providers* participating in the negotiation; *number of bids per provider* (number of bids each provider sends to the *Manager* agent); *number of RFQ items* (number of items to be negotiated by the *Buyer* agent); *number of items per bid* (number of items within each bid sent by a provider); *number of units per item per bid*; and *bid cost per item*. In this first experimental scenario we did not generate neither inter-item nor intra-item constraints.

The generator starts by randomly creating a set of winning combinatorial offers. After that, it generates the rest of bids for the negotiation scenario employing normal distributions based on the values set for the parameters above. Thus, in some sense, the negotiation scenario can be regarded as a set of winning combinatorial bids surrounded by noisy bids (far less competitive bids). Notice that the generator directly outputs the RFQ and bids composing an artificial negotiation scenario in FIPA format. In this manner, both RFQ and bids can be directly fed into *iBundler* as buyers' and providers' agent messages.

We have analysed the performance of *iBundler* through a large variety of negotiation scenarios artificially generated by differently setting the parameters above. The data representing each negotiation scenario are saved onto a file, named by a string of type *A.B.C.D*, where *A* stands for the number of providers, *B* stands for the number of bids per provider, *C* stands for the number of RFQ items, and *D* stands for the number of items per bid. For instance, 250.20.100.20 represents the name of a data set generated for 250 providers, 20 bids per provider, 100 RFQ items, and 20 items per bid.

The artificial negotiation scenarios we have generated and tested result from all the possible combinations of the following values:

- Number of providers: 25, 50, 75, 100;
- Number of bids per provider: 5, 10, 15, 20;
- Number of RFQ items: 5, 10, 15, 20;
- Number of items per bid: 5, 10, 25, 50.

5.2. *Evaluation stages*

In order to introduce the evaluation stages that we considered, it is necessary to firstly understand how JADE manipulates messages and ontological objects. In particular we summarise the process of sending and receiving messages (for a complete description refer to the JADE documentation). Fig. 8 graphically summarises the activities involved in sending and receiving messages. In the figure, the squared boxes represent data, whereas the rounded boxes represent processes.

JADE agents receive messages as serialised objects in string format. JADE decodes the string into a Java class, the *ACLMessage* JADE class (which represents a FIPA ACL Message). One of these class fields is the *content* field, which usually contains either the action to be performed or

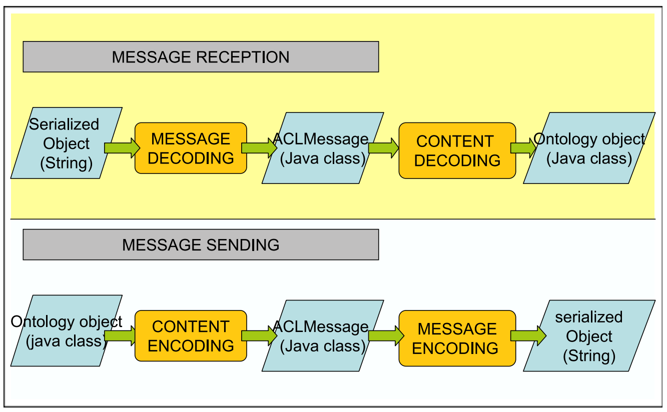


Fig. 8. Message life cycle in JADE.

the result of a performed action. Next, JADE extracts the content of the message. The content is once more a string, on which JADE needs to perform an ontology check to decode it. As a result, a Java object representing the ontological object is built upon the *content* field, guaranteeing that the ontological structure is not violated.

As to the dual case, i.e. when a JADE agent sends a message, the process works the other way around. JADE encodes the ontological object representing the communication content into a string that sets the *content* field of the *ACLMessage* class. During this process JADE verifies that the message content matches perfectly with an ontology object. Once the *content* field is set, the agent sends the message: the ACLMessage class is decoded into a string that is sent through a socket.

Considering the process above, we sampled both the time and memory use through the following stages of the *iBundler*'s solving process:

- $\Delta t1$: JADE decodes all the FIPA messages contained in the data set file containing the input negotiation problem, converting them into instances of the *ACLMessage* Java class.
- $\Delta t2$: the *Manager* agent composes the problem by creating an instance of the *Problem* Java ontology class and setting its fields after merging the RFQ and the collected bids.
- $\Delta t3$: the ACLMessage to be sent to the *Translator Agent* is filled with the Java class representing the *Problem* ontology class. At this stage an ontology check occurs.
- $\Delta t4$: the above-mentioned ACLMessage is now encoded by the *Manager* agent, and subsequently sent to the *Translator* agent through a socket. Once received, the *Translator* agent decodes it into an *ACLMessage* class.
- $\Delta t5$: the *Translator* agent extracts from the received message the *Problem* ontology class containing the *RFQ* and all the collected *Bids*. Another ontology check is involved in this phase.
- $\Delta t6$: this stage is devoted to the transformation of the *Problem* ontology class into a matrix-based format to be processed by the *Solver* component.
- $\Delta t7$: at this stage the *Solver* component solves the MIP problem using ILOG CPLEX.
- $\Delta t8$: the output generated by *Solver* in matrix-based format is decoded by the *Translator* agent into the *Solution* ontology class.
- $\Delta t9$: the *Translator* agent fills the response message with the *Solution* ontology class, encodes the corresponding *ACLMessage* class, and sends it. After that, the *Manager* agent decodes the message upon reception.
- $\Delta t10$: the *Manager* agent extracts the *Solution* concept from the received *ACLMessage*. The last ontology check occurs.
- $\Delta t11$: the solution is decomposed into different parts, one per provider owning an awarded bid.
- $\Delta t12$: the solution containing the set of winning offers is sent from the *Manager* agent to the *Buyer* agent. Note that this object is small with respect to the original problem since it only contains the winning bids.

5.3. Time performance

Next we show the variation in time performance per stage by varying the different degrees of freedom available to create an artificial negotiation scenario. In particular, we consider the following types of negotiation scenarios:

- 100.20.100.X: the number of items contained in a single bid varies (where X takes on the $5, 10, 25$, and 50 values).
- 100.X.100.50: the number of bids each provider sends varies (where X takes on the $5, 10, 15$, and 20 values).
- X.20.100.50: the number of providers varies (where X takes on the $25, 50, 75$, and 100 values).

Fig. 9 depicts the time spent in each of the described stages, considering different number of bids per provider. We experimented similar trends varying the number of items and the number of providers.[3] These results suggest that the variables' sensitivity is similar in all cases, i.e. varying the *number of items per bid*, the *number of providers* or the *number of bids per provider* leads to similar trends. Therefore, the stages that are more time-consuming are quite the same in every possible configuration: for instance, stage $\Delta t10$ is always the most time-consuming, no matter the parameter being varied. Moreover, we can observe similar trends for the rest of stages (from $\Delta t1$ to $\Delta t10$). Hence, it seems that the time distribution along the different stages can be regarded as independent from the parameter setting.

Fig. 10 illustrates the average percentage, over all the performed trials, of the total time that each stage consumes. We observe that: (1) The $\Delta t1$, $\Delta t3$, $\Delta t4$, $\Delta t5$, $\Delta t9$, $\Delta t10$ stages are the most time-consuming (92% of the total time). Since these stages involve ontology checking and message encoding and decoding, we can conclude that these activities are a bottle-neck. (2) The solver time ($\Delta t7$) is almost a negligible part of the total time. (3) Manipulating classes (stages $\Delta t2$, $\Delta t6$, $\Delta t8$ and $\Delta t11$) and solving the combinatorial problem ($\Delta t7$) are not as time-consuming as encoding and decoding messages and ontology objects.

Fig. 11 depict the accumulated time spent on all stages for a collection of negotiation scenarios, which we refer to as the *total time*. More precisely, Fig. 11 depicts configurations whose total time lies between 30 and 50 s. It is conceivable to regard them as the edge values, although it

[3]The way in which the times varies increasing these parameters is not linear. Anyway we did not study deeply this aspect, since the main question for us is the difference of these times with respect to the solver component time by itself.

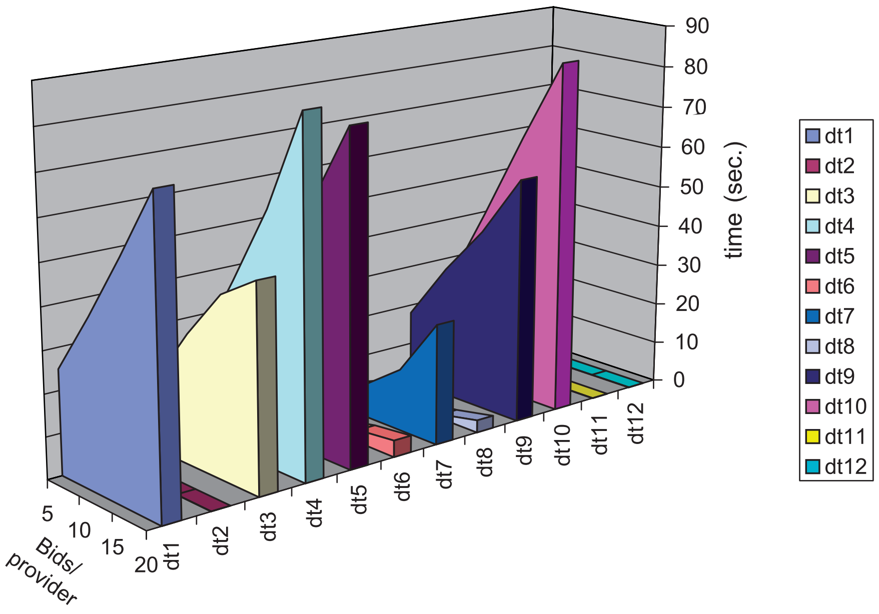


Fig. 9. Time performance.

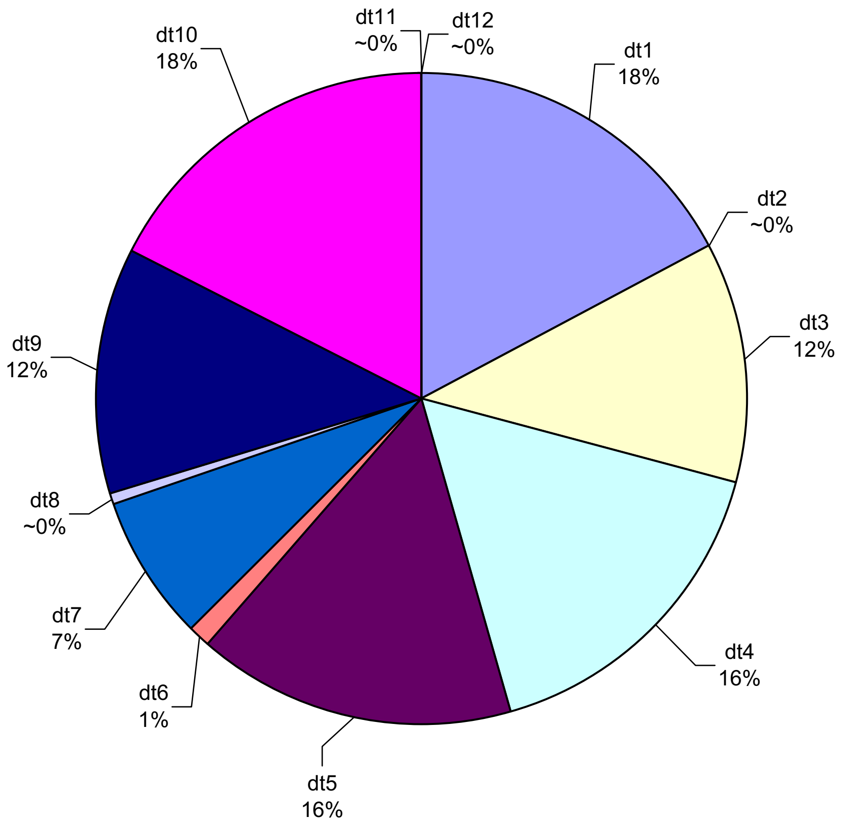


Fig. 10. Average time per evaluation stage.

is a very arbitrary matter. Some observations follow from analysing the figures above:

- The agent-awareness of *iBundler* is costly. We observe that the percentage of total time employed to solve the winner determination problem is small with respect to agent related tasks.
- Using the solver component we can easily solve problems of more than 2000 bids in less than 1 min, whereas the agent service can handle in reasonable time less than 750 bids.

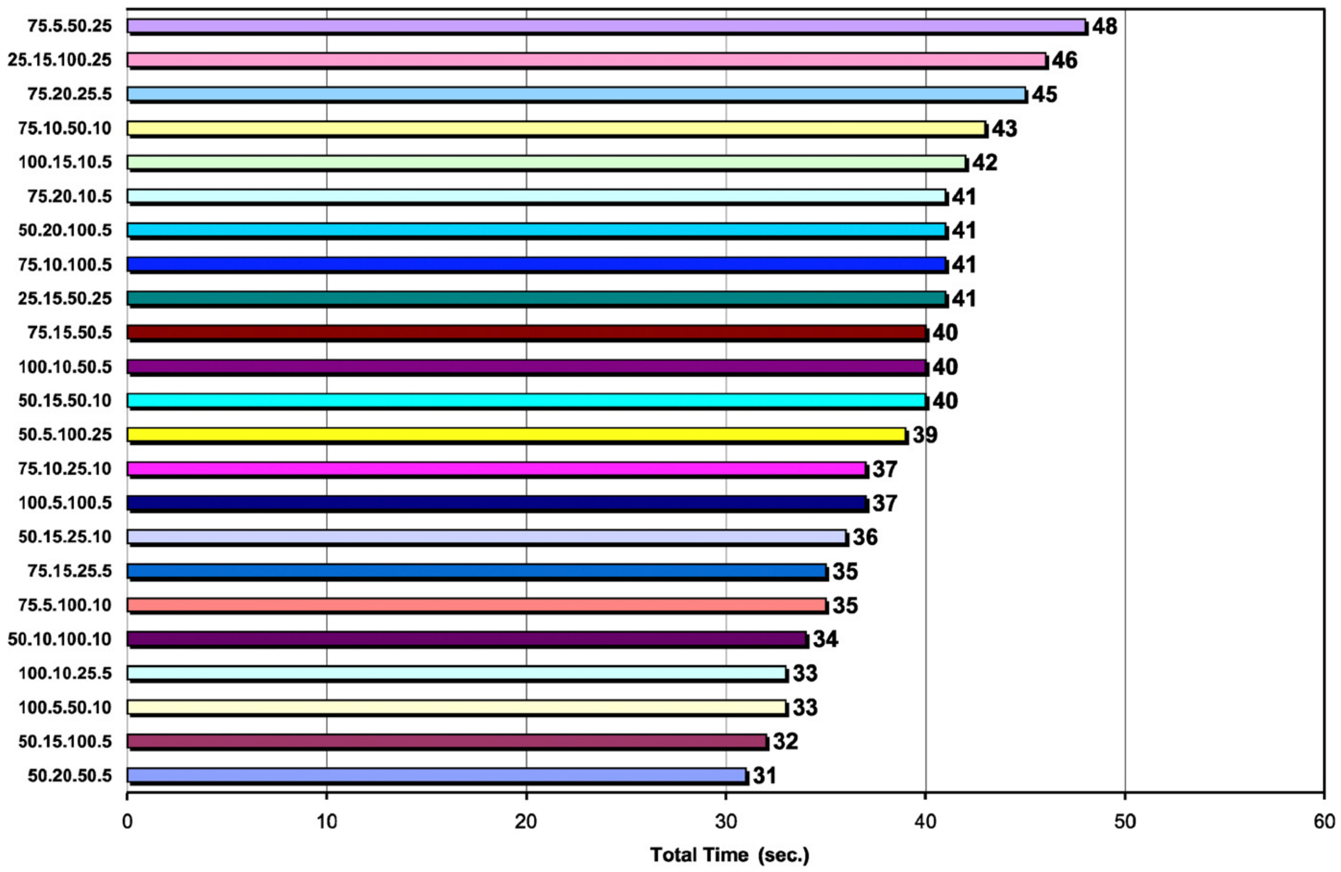


Fig. 11. Time performance for negotiation scenarios on the edge of acceptability.

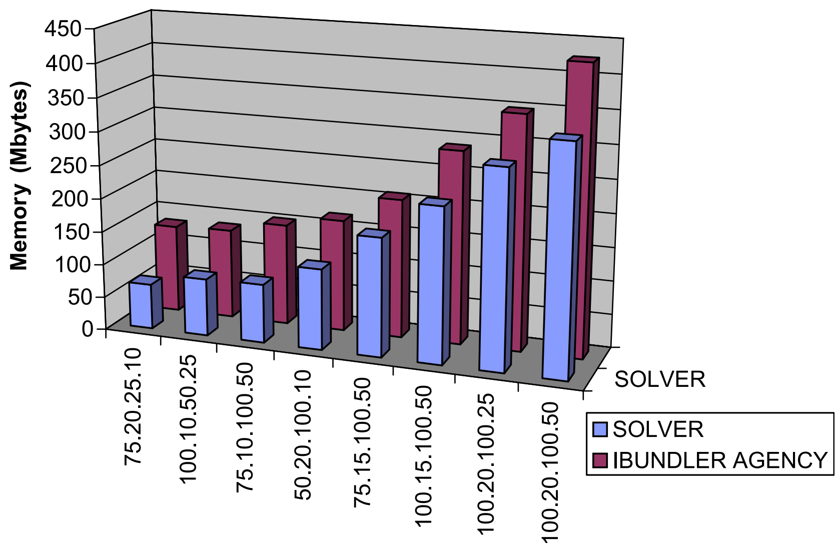


Fig. 12. Memory use.

- Therefore, small and medium-size negotiation scenarios can be soundly tackled with *iBundler*. Nonetheless, time performance significantly impoverishes when handling large-size negotiation scenarios.

5.4. Memory use

In this case we found similar results when comparing the *Solver* component with *iBundler*. The amount of memory required in the worst case is quite the same for both cases. The memory consumption in both cases is highly dependent on the ontology structure. It is not surprising that the memory peak is similar in both cases, as the information quantity to represent is actually the same. The biggest amount of information is used to represent all the bids. Both *Solver* and JADE have to load in memory the information representing a problem, namely an RFQ and the received bids (the former as a Java object and the latter as a file containing matrices). Fig. 12 compares the memory use for the *iBundler* agency and *Solver*.

6. Conclusions

This paper describes the implementation of the *iBundler* service, an innovative agent-aware decision support service for negotiation scenarios that operates as a winner determination solver for both multi-item, multi-unit negotiations and auctions. Although the current implementation is largely inspired on the interaction with professional buyers through the development of the sourcing (Aberdeen Group, 2002) solution described in Reyes-Moro et al. (2003), it is our belief that *iBundler* is general enough to effectively empower agents to conduct from simple to largely sophisticated negotiations in open agent environments. Notice that the implementation of *iBundler* contributes along two main directions. On the one hand, we have incorporated actual-world side constraints to the winner determination problem for CA. On the other hand, we have realised a new ontology that accommodates both operational constraints and attribute-value constraints for buying and providing agents, offering a highly expressive bidding language.

The tests that we ran show that offering *iBundler* as an agent service implies a significant time overload, while the memory use is only slightly affected. The main cause of such an overload is related to the encoding and the decoding of ontological objects and messages. The message serialisations and deserialisations, along with ontology checkings heavily overload the system as the dimensions of the negotiation scenario grow. We propose several actions to alleviate this effect. Firstly, we have observed that the main amount of information is gathered in representing bids. Their presence in objects and messages is the foremost cause of *iBundler*'s time overload. Thus, a suitable workaround would be to use, at ontology design time, a more synthetic bidding language, in which bids can be expressed more concisely. For instance, introducing a preprocessing phase in which equal (and even similar) bids are grouped, in order to obtain a more compact representation. The resulting ontology would generate more tractable objects. Secondly, it would be also helpful to improve the performances of the JADE modules devoted to the ontology checking and serialisation processes. All in all *iBundler* can satisfactorily handle small and medium-size negotiation scenarios. Thus, although the automation of the negotiation process with agents helps in saving time in managing negotiations, the scalability in terms of time response of *iBundler* is limited.

Acknowledgements

This work has been supported by project Web-i(2) (TIC-2003-08763-C02-01). Andrea Giovannucci enjoys the BEC.09.01.04/05-164 CSIC scholarship.

References

Aberdeen Group, 2002. Making e-sourcing strategic: from tactical technology to core business strategy. Technical report, Aberdeen Group.

Acklin, B.V., 2005. Java protege beangenerator. ⟨http://acklin.nl/page.php?id=34⟩.

Agentcities Consortium, 2002. Demonstration documentation for checkpoint 1. Technical Report Deliverable D5.4, IST-2000-28385.

Andersson, A., Tenhunen, M., Ygge, F., 2000. Integer programming for combinatorial auction winner determination. In: Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS), Boston, MA, pp. 39–46.

Bichler, M., Kalagnanam, J., 2005. Configurable offers and winner determination in multi-attribute auctions. European Journal of Operational Research 160 (2).

Brazier, F., van Steen, M., Wijngaards, N., 2001. On MAS scalability. In: Proceedings of Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada, pp. 121–126.

Caire, G., 2002. Jade tutorial. Application-defined content languages and ontologies. Technical report, TILAB S.p.A.

Deters, R., 2001. Scalability & multi-agent systems. In: Proceedings of Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada.

Fedoruk, A., Deters, R., 2002. Improving fault-tolerance by replicating agents. In: AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, New York, pp. 737–744.

FIPA, Fipa. ⟨http://www.fipa.org⟩.

FIPA, 2003. FIPA interaction protocol library specification. Technical Report DC00025F, Foundation for Intelligent Physical Agents.

Fujishima, Y., Leyton-Brown, K., Shoham, Y., 1999. Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. In: Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99), pp. 548–553.

Giovannucci, A., Rodríguez-Aguilar, J.A., Reyes-Moro, A., Noria, F.X., Cerquides, J., 2004. Towards automated procurement via agent-aware negotiation support. In: Third International Joint Conference on Autonomous Agents and Multiagent Systems, New York.

Hohner, G., Rich, J., Ng, E., Reid, G., Davenport, A.J., Kalagnanam, J.R., Lee, H.S., An, C., 2003. Combinatorial and quantity-discount procurement auctions benefit mars, incorporated and its suppliers. Interface 33 (1), 23–35.

Holte, R.C., 2001. Combinatorial auctions, knapsack problems, and hill-climbing search. In: Lecture Notes in Computer Science, vol. 2056. Springer, Heidelberg.

Hoos, H.H., Boutilier, C., 2000. Solving combinatorial auctions using stochastic local search. In: Proceedings of AAAI-2000.

Jarrar, M., Meersman, R., 2002. Scalability and knowledge reusability in ontology modeling. In: Milutinovic, V. (Ed.), Proceedings of the International conference on Infrastructure for e-Business, e-Education, e-Science, and e-Medicine, vol. SSGRR2002s. SSGRR education center, Rome, Italy.

Kahn, M.L., Della Torre Cicalese, C., 2003. COABS grid scalability experiments. Autonomous Agents and Multi-Agent Systems 7 (1–2), 171–178.

Kalagnanam, J., Parkes, D.C., 2004. Auctions, bidding, and exchange design. In: Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era. Dordrecht, D. Simchi-Levi, Wu, Shen (Eds.), Kluwer Academic Publishers.

Kelly, F., Steinberg, R., 2000. A combinatorial auction with multiple winners for universal service. Management Science 46, 586–596.

Klein, M., Rodriguez-Aguilar, J., Dellarocas, C., 2003. Using domain-independent exception handling services to enable robust open multi-agent systems: the case of agent death. Autonomous Agents and Multi-Agent Systems 7 (1–2), 179–189.

Makhorin, A., 2001. Glpk—gnu linear programming toolkit. ⟨http://www.gnu.org/directory/GNU/glpk.html⟩.

Odell, J., van Dyke Parunak, H., Bauer, B., 2000. Extending UML for agents. In: Proceedings of the Agent-Oriented Information Systems Workshop, Austin, TX. Seventeenth National Conference on Artificial Intelligence, pp. 3–17.

OMG and FIPA standardisation for agent technology: competition or convergence? 1999. ⟨http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/ch2/ch2.htm⟩.

Protege, 2005. Protege 2000. ⟨http://protege.stanford.edu⟩.

Reyes-Moro, A., Rodríguez-Aguilar, J.A., López-Sánchez, M., Cerquides, J., Gutiérrez-Magallanes, D., 2003. Embedding decision support in e-sourcing tools: quotes, a case study. Group Decision and Negotiation 12, 347–355.

Rothkopt, M.H., Pekec, A., Harstad, R.M., 1995. Computationally manageable combinatorial auctions. Management Science 8 (44), 1131–11147.

Sandholm, T.W., 2002a. Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence 135, 1–54.

Sandholm, T.W., 2002b. emediator: a next generation electronic commerce server. Computational Intelligence 18 (4), 656–676 (special issue on agent technology for electronic commerce).

Sandholm, T.W., Suri, S., 2001. Side constraints and non-price attributes in markets. In: International Joint Conference on Artificial Intelligence (IJCAI), Seattle, WA. Workshop on Distributed Constraint Reasoning.

Sandholm, T.W., Suri, S., Gilpin, A., Levine, D., 2002. Winner determination in combinatorial auction generalizations. In: First Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy, pp. 69–76.

Tamma, V., 2003. An experience in using ontologies for automated negotiation. In: Presentation at the Agentcities Information Day 4.

Tamma, V., Wooldridge, M., Dickinson, I., 2002. An ontology for automated negotiation. In: First Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy.

TILAB, 2005. Jade, the java agent development framework. ⟨http://jade.tilab.com⟩.

Wache, H., Serafini, L., García-Castro, R., 2004. D2.1.1 survey of scalability techniques for reasoning with ontologies. Technical report, Knowledge Web.

Yoo, M.-J., 2002. An industrial application of agents for dynamic planning and scheduling. In: AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, New York, pp. 264–271.

Zurel, E., Nissam, N., 2001. An efficient approximate allocation algorithm for combinatorial auctions. In: Proceedings of the ACM Conference on Electronic Commerce, Tampa, Florida.