Enabling Assisted Strategic Negotiations in Actual-world Procurement Scenarios

Jesús Cerquides , Maite López-Sánchez

WAI Volume Visualization and AI Research Group

Departament de Matemàtica Aplicada i Anàlisi, MAiA

Facultat de Matemàtiques, Universitat de Barcelona

Gran Via de les Corts Catalanes, 585

08007 Barcelona, Spain

cerquide@maia.ub.es, maite@maia.ub.es

Antonio Reyes-Moro

ATAMA Solutions

Ildefons Cerdà 14, 2º

08172 Sant Cugat del Vallès, Barcelona, Spain

toni.reyes@atamasolutions.com

Juan A. Rodríguez-Aguilar

IIIA-CSIC, Artificial Intelligence Research Institute

Campus de la Universitat Autònoma de Barcelona

08193 Bellaterra, Barcelona, Spain

jar@iiia.csic.es

Abstract

In the everyday business world, the sourcing process of multiple goods and services usually involves complex negotiations (via telephone, fax, etc) that include discussion of product and service features. Nowadays, this is a high-cost process due to the scarce use of tools that streamline this negotiation process and assist purchasing managers' decision-making. With the advent of Internet-based technologies, it has become feasible the idea of tools enabling low-cost, assisted, fluid, on-line dialogs between buyer enterprises and their providers wherever they are located. Consequently, several commercial systems to support on-line negotiations (e-sourcing tools) have been released. It is our view that there is still a need for these systems to incorporate effective decision support. This article presents the foundations of *Quotes,* a commercial sourcing application developed by iSOCO that, in addition to cover the whole sequence of sourcing tasks, incorporates decision support facilities based on Artificial Intelligence (AI) techniques that successfully address previous limitations within a single and coherent framework. The paper focuses on the computational realisation of sourcing tasks along with the decision support facilities they require. While supported negotiation events are Request for Quotations/Proposals (RFQs/RFPs) and reverse auctions, decision support facilities include offer generation, offer comparison, and optimal bid set computation (winner determination) in combinatorial negotiations. Additionally, the paper presents a compound of experiences and lessons learned when using *Quotes* for real sourcing processes.

Keywords: negotiation, e-procurement, sourcing, auctions, artificial intelligence.

Submitted: 30-05-2003

Accepted: 07-07-2006

1 Introduction

The sourcing process of multiple goods or services usually involves complex negotiations that include discussion of products' features as well as quality, service and availability issues. Consequently, several commercial systems to support on-line negotiations (e-sourcing tools) have been released. In fact, e-sourcing is becoming an established part of the business landscape [28].

By industry, the highest penetration [28] is in chemicals and pharmaceuticals (24 percent) and automobile manufacturing (20 percent), whereas retail, wholesale and distribution, metals and metal products, finance, banking, and accounting are at 4 percent each. Across all industries, e-sourcing adopters are using the technology primarily for maintenance, repair, and operations (MRO) goods (71 percent), followed by standard parts (67 percent) and raw materials (57 percent).

However, there is still an enormous challenge confronting users who want to get the maximum value out of e-sourcing. Many companies have realized this value on a transactional level but failed to see it through to the bottom line. The early value of e-sourcing tools has been tactical rather than strategic [1], as most adopters got into e-sourcing primarily to negotiate price reductions. Therefore there is still a need for these systems to incorporate effective decision support to enact the focus on strategy.

Traditionally, the core of the sourcing process comprises the following tasks:

- Request for Quotation/Proposal (RFQ/RFP) elaboration;
- Provider selection for RFQ/RFP delivery;
- Offer generation;
- Negotiation through offer-counteroffer interaction or reverse auction; and
- Selection of best offers.

This paper aims at describing how the above-mentioned sourcing tasks are performed by *Quotes*.

Although several commercial systems to support on-line negotiations have been released, to the best of our knowledge, not a single system can claim to address the full complexity of on-line negotiations. The first generation of sourcing tools merely incorporates single-item, price-quantity reverse auctions mechanisms (see [3] and [7]). Others only offer basic negotiation capabilities that are usually reduced to a demand-offer matching tool (for example, on-line vertical marketplaces such as MetalSite [19]). In general terms, there is a lack of decision support functionalities (decision making in sourcing can involve a few hundred offerings each of which is described by several dozen attributes). Finally, there is a lack of technology support for computationally complex negotiation paradigms, which inhibit the application of interesting models such as combinatorial reverse auctions ([15], [25], [8]) and multi-stage negotiations.

This article presents *Quotes* ([20], [21]) as a solution for strategic sourcing that we believe satisfactorily addresses previous limitations within a single and coherent framework. *Quotes* is a commercial tool that supports all sourcing tasks enumerated above. These tasks are solved by means of different Artificial Intelligence (AI) techniques that might be not theoretically new but that together form an innovative tool that pose an opportunity to proof them useful in real sourcing events.

From the point of view of decision support, we have identified three processes where to apply AI techniques that help buyers and providers in their decision making processes. These three processes have been studied and implemented in *Quotes* with satisfactory results. Although they are thoroughly described along different sections in this paper, next we briefly summarize them:

1. *Automated offer generation*. Providers can translate their business knowledge into *bidding* rules that allow instantaneous and automatic construction of recommended offers. A random neighbourhood search algorithm controlled by a rule based system

reasons with these rules in order to construct an initial offer that maximizes both buyer and provider preferences, thus rapidly conducting negotiations to *win-win* situations.

2. *Multi-attribute, multi-item scoring algorithm*. Based on the importance assigned by the buyer to each item attribute in an RFQ (price, quality, delivery time, etc.) and his flexibility to accept offers beyond his preferences, a *fuzzy* offer-matching algorithm scores each offer and ranks it accordingly. In this manner, the buyer can easily discriminate competitive from non-competitive offers. Analogously, providers can benefit from the very same algorithm in order to discriminate incoming RFQs.
3. *Computation of the optimal bid set (winner determination) for multi-item (combinatorial) negotiations*. Given a set of offers for either a multi-item RFQ or an auction, a mixed integer programming algorithm obtains the optimal subsets of offers according to various criteria such as minimization of price, start/finish date, maximization of product quality, etc. Multi-criteria optimisation can be supported by using multi-attribute scoring. Although the used algorithm is not new, the way the problem is formulated can be considered as an important contribution to research in combinatorial negotiations.

The rest of the paper is organised starts with an overview of the system architecture in section 2 and continues with ten different sections that can be grouped into two main blocks. Sections 3 to 8 correspond to the sourcing tasks enumerated above, whereas the last three sections are dedicated to more general aspects. The first block starts with section 3 describing RFQs elaboration. Section 4 introduces the supplier side by introducing the definition of production profiles and provider selection for RFQ delivery. Still on the provider side, Section 5 describes manual and automated (via business rules) offer generation. Next, offer-counteroffer interaction and reverse auctions are presented separately, in sections 6 and 7 respectively. Finally, section 8 details negotiation closing through the selection of preferred offers. The second block starts with section 9 presenting how different negotiation stages (RFQ/RFP and reverse auctions) can be interleaved in a single sourcing event. Section 10 compiles iSOCO's experiences when organizing real sourcing events with the *Quotes* tool. Finally, some conclusions and related work are provided in Section 11.

2 System Architecture Overview

This section provides an overview of *Quotes*' functional architecture to identify the modules that realise the above described decision making processes.

Figure 2-1 depicts *Quotes*' functional architecture. White boxes depict core functional modules connected by arrows that stand for their dependencies, whereas grey boxes represent technical units that bridge *Quotes* to external services.

As to functional modules, we distinguish the following:

- *Sourcing Event Manager*: It supports the creation and management of sourcing events through their different stages.
- *Negotiation Engine*: It is responsible for running a sourcing event stage complying with its negotiation or auction protocol rules.
- *Provider Selection*: It filters in and out the providers to be involved in a sourcing event.
- *Offer Generation*: It allows for either manual or automatic offer/bid construction.
- *Winner Determination*: It determines the set of winning bids for auctions.
- *Ranking assessment*: It assesses the ranking of offers and RFXs (RFI, RFP, RFQ).
- *Scoring*: It scores the degree of matching of both a buyers' requirements with the offers he/she receives, and a provider's preferences with RFXs he/she receives.

Additionally, a collection of technical units bridge *Quotes* to external services: a persistence service provides connection to a database; an integration gateway acts as an interface to third party services (e.g. ERPs); a user interface module manages the web façade; a notification module is in charge of e-mail communication; and further services (such as localisation, logging, and authentication) extend the capabilities of the overall application.

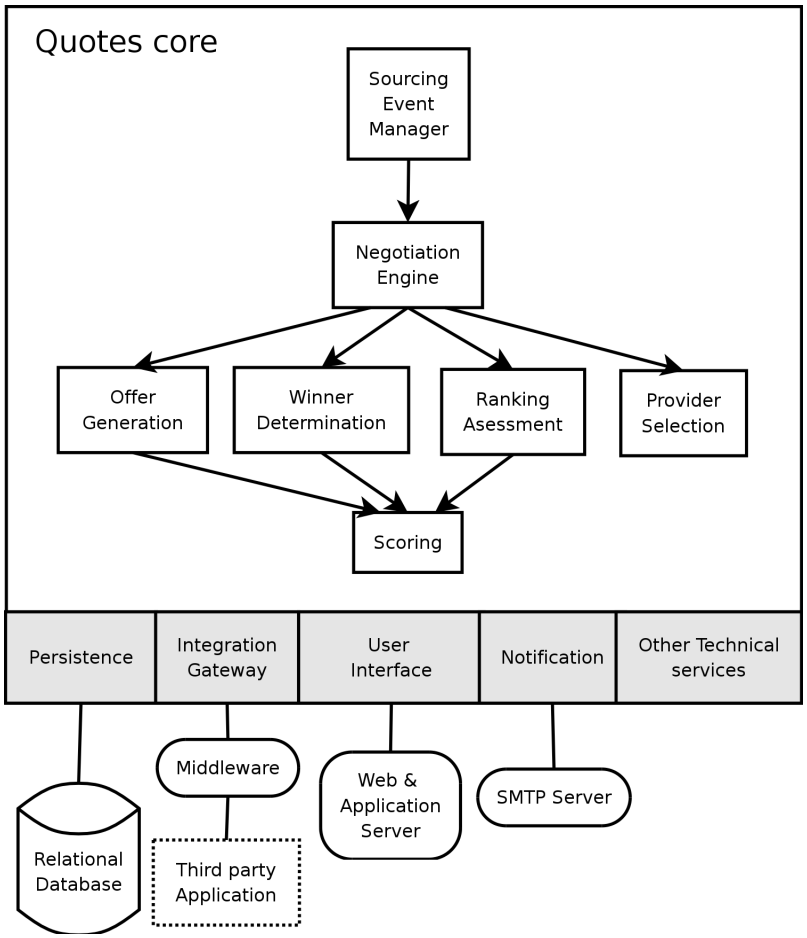

Figure 2-1 : System Architecture.

3 Request for quotation (RFQ) elaboration

Quotes supports multi-attribute, multi-item RFQs, enabling the creation of multiple types of RFQs (commodity, catalogue, BOM --Bill of Material--, or group-by). Furthermore it provides the expressiveness needed to cope with multi-criteria negotiation procedures.

Typically a buyer creates an RFQ by sequentially adding items. Each item specifies a product, be it either a good or service. A paradigmatic example of multi-item RFQ occurs at industrial settings. The production plan outlined by some company's ERP (Enterprise Resource Planning) or SCM (Supply Chain Management) application comes in the shape of a list of items to be produced along with the parts required for each product, the so-called bill of material. It is the basis for the buyer to start out multiple sourcing events, each one devoted to the procurement of the parts of each one of the items whose production has been planned out.

The process of including an item is composed of two steps: template selection and RFQ value setting. It starts when some buyer creates an RFQ item from a list of product templates. A product template consists of a list of attributes describing the product. Each attribute is defined in terms of a name, its unit, and its domain. There are four main types of attribute domains:

- *Any Number*. For attributes that can take any numerical value. This type is recommended for attributes whose values cannot be constrained. (e.g. 'Delivery time').
- *Range of Numbers*. This type is intended for attributes whose possible values belong to a numerical range and it is always associated to a maximum and a minimum value definition. For example, 'Quantity' taking on values between 100 and 1000.

- *Set of Labels*. Non-numerical values can be of type 'set of labels' when there is an associated list of predefined textual values and there is no order among them. For instance, 'Conventional', 'Semi-synthetic', and 'Full-synthetic'.
- *Ordered Set of Labels*. Used for quantitative measures defining an order for associated values. As an example, 'Temperature resistance' degrees such as 'Low', 'Medium', and 'High'.

Once a product template is selected, the buyer selects a value type per attribute (which in turn depends on its attribute type) together with its requested values and its relative importance with respect to the rest of attributes (see next Figure 3-1). For instance, for an *Any Number* attribute a buyer can specify its desired value as either a single number or as a range of numbers satisfying his needs. In the last case, three values are required: a minimum, a maximum, and a preference slope indicating which values are most preferred within the range. The slope can take on three different values:

- *Flat*. All values within the range are equally preferred.
- *More Is Better (MIB)* indicates that higher values are more preferred than lower ones.
- *Less Is Better (LIB)* specifies that lower values are more preferred than higher ones.

The same kind of single / range value specification can be used for the *Range of Numbers* and *Ordered Set of Labels* types. The only difference is that attribute values must belong to the domain defined by the corresponding attribute template. As to the *Set of Labels* type, the buyer is enabled to choose one value (label) or a subset of values. Notice though that since no order is associated, there is no need for defining any range or slope.

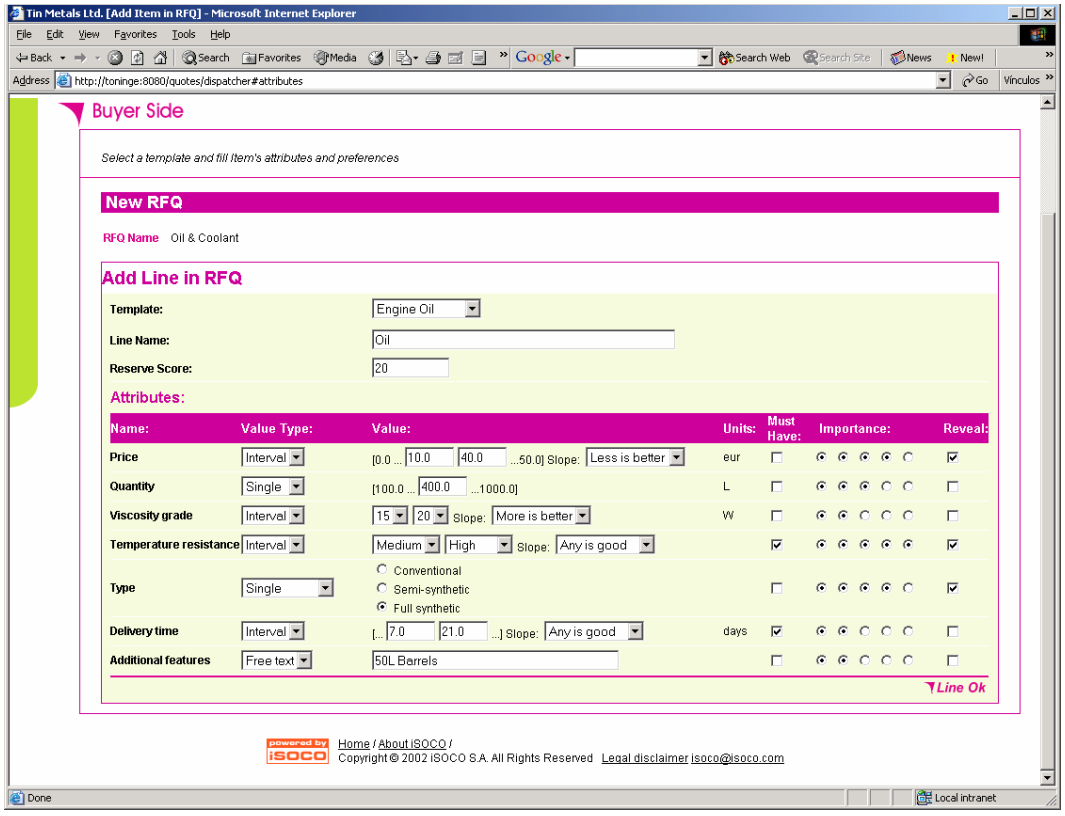

Figure 3-1. RFQ value specification.

4 Automated provider selection

4.1 Providers' preferences and capabilities: production profiles

While buyers need to specify their product requirements in terms of negotiable attributes, providers can analogously do the same regarding their product capabilities and their preferences over incoming RFQs. When specifying their production profiles, providers are requested to specify both their production capabilities and their selling preferences:

- Production capabilities describe the products that can be offered. They are specified through the range of attribute values that can be provided. *Quotes* uses this information for RFQ delivery.
- Selling preferences allow a provider to state which requests he may favour. With this information, *Quotes* is able to aid the provider when deciding which RFQs to prioritise. This is a way of reducing the high cost involved in analysing large collections of RFQs.

4.2 Provider selection

Provider selection is the automatic process that guarantees that only those providers supplying some of the required goods do receive the RFQ. In addition, providers will see customised views of these RFQs, so that the *RFQ view* for a given provider exclusively contains those items that he or she is capable of providing.

Provider selection consists of two filtering steps. The first step solely requires identifying those providers offering products specified with the same template than the product the buyer is requesting. For every identified provider, the second step focuses on checking the compatibility between providers' capabilities and buyers' requested attribute values.

5 Offer generation

Once an RFQ has been delivered to selected providers, those providers can then generate their first offers. *Quotes* allows offers to be elaborated either manually by providers or automatically by the system. Nevertheless, since providers in real sourcing events prefer to have complete control over their offering, the automated offer generation functionality limits to composing indicative offers (as opposed to firm offers).

5.1 Manual offer elaboration

Figure 5-1 shows an RFQ item description as seen by a provider.

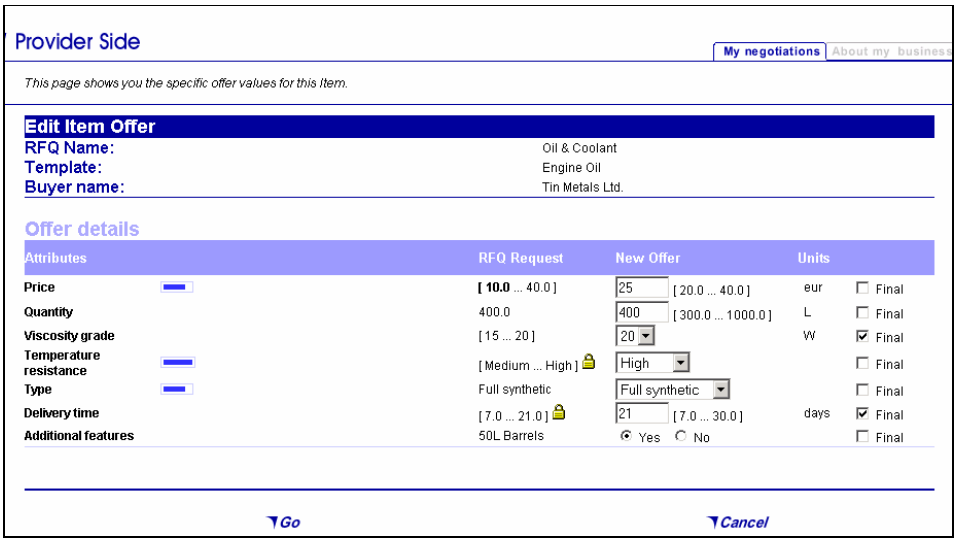

Figure 5-1. Offer edition for an RFQ item

Notice that, whilst an RFQ item represents a subspace of preferred attribute values within the space defined by the template, offers are restricted to represent points in the very same space. The reason is mainly semantic, since assigning more than one value to each attribute could be ambiguous in representing interdependencies and could in fact be interpreted as different alternative offers. Attribute value alternatives (disjunction) must be specified as separate offers.

5.2 Automated provider response

The system requires providers to specify their business rules so that they can be subsequently applied to automatically generate offers.

5.2.1 Business rules as bidding rules

A bidding rule is an *if-then* rule that checks and changes the value of one or several attributes. Examples of rules include discount per volume, additional charges for express delivery, no delivery charge when a minimum price is offered, etc. Figure 5-2 shows the general syntax to define bidding rules.

RULE: ***if*** CONDITION ***then*** ACTION;

CONDITION: attribute_name1 [=, !=] value1 |

attribute_name1 [#,!#] (min_value, max_value) |

(CONDITION) |

CONDITION [and, or] CONDITION;

ACTION: attribute_name2 [fix, +%, -%] value2

Figure 5-2: Bidding rule syntax

Operations for checks and changes can be selected among a list of available operations that depend on the chosen attribute type:

- *Condition operators* are: =, !=, #, !#. Equality (=) and inequality (!=) operators can be chosen for all types of attributes (both numerical and sets of labels) and require a value to be compared with. On the contrary, membership (#) and non-membership (!#) operators only apply to types that allow to specify intervals (that is, *any number*, *range of numbers*, and *ordered sets of labels*).
- *Action operators* are functions to be applied to the attribute that has been chosen to be affected. These are: assignment (fix), which can be applied to all attribute types, and percentage increment (+%) and decrement (-%), which are only applied to numerical types.

5.2.2 Automated offer generation

In this section we describe the module of *Quotes* that is in charge of generating indicative offers on behalf of providers exploiting the above-mentioned bidding rules. Such process takes place immediately after provider selection. Its objective is to build a complete offer (where each attribute is assigned some value). Moreover, the algorithm pursues to build the *best* complete offer in terms of either the buyer's preferences or the provider's preferences or both. For the sake of clarity, this section assumes an offer to be composed of a single item (that is, it is only offering one product), so that 'offer' is used instead of 'offer item'. For the general case of offering more than one product, the process described below will be repeated for each item.

The implemented algorithm is a variation of the well-known hill-climbing random neighbourhood search procedure. We start with an incomplete offer *O* as the candidate solution and then enter into an iteration phase. Each step we generate a neighbouring offer *O'* of *O* to which we apply the provider's business rules (which have been already explained in the pervious subsection). If the new solution *O'* is better than the candidate offer *O* we accept *O'* as the current solution. The process continues until the termination criterion (no improvement for the last *k* iterations) is reached.

In order to explain completely the optimisation procedures, the following definitions are in place.

Definition 1: Space of solutions.

We define an offer as a tuple *o* of the form $<o_1, o_2,\ldots, o_n>$ where $o_i \in (A_i^p \cap A_i^b) \cup \varnothing$, *i*=1,..,n, being:

- *n* the number of negotiable attributes of the offered item;
- A_i^p the set of values produced by provider *p* for the i-th attribute (a_i);
- A_i^b the set of values ask by buyer *b* for the i-th attribute (a_i);
- the symbol $\varnothing$ denotes unassigned value.

In other words, *an offer o* sets a value for each requested attribute (i.e., o_i is the value offered for attribute a_i), provided that such value is within both provider's capabilities and buyer's acceptable values.

We say that an offer *o* is *complete* if $o_i \neq \varnothing \ \ \forall\ i$=1,..,n and *partially complete* if $\exists i$ such that $o_i = \varnothing$. Additionally, we say that an offer *o* is *more partially complete* than *o'* (denoted as $o > o'$) if *o* has less or equal *unassigned* values than *o'.*

The evaluation function defines the objective of the optimisation process in terms of determining if *o*' is better than *o*. In our case the objective is twofold: obtain a complete offer that optimises a target function.

Definition 2: Objective function.

The objective function *c*: $O \rightarrow \Re$ is defined over the set of offers *O* as

$$c(o) = w^p \cdot \sqrt{s^p(o)} + w^b \cdot \sqrt{s^b(o)}$$

where

- s^p: $O \rightarrow$ [0..100] is the scoring function based on the preferences of provider *p* (as detailed in subsection 8.1).
- s^b: $O \rightarrow$ [0..100] is the scoring function based on the preferences of buyer *b*.
- w^p and w^b are weighting factors so that $w^p + w^b = 1$ and $w^p \geq 0$, $w^b \geq 0$

Such evaluation function tries to favour high scorings while penalizing big differences between the buyer's and the provider's revenues. In other words, it prefers a 50 - 50 rather than an 80 - 20 thus seeking win-win situations (and this is desirable because if a provider generates an offer that best suites his/her preferences without taking into account the buyer's ones, it will be likely rejected by the buyer). Finally, w^p and w^b can be used to tune the objective function (for example, to focus more on the buyer's preferences).

Optimisation algorithm

Informally, we say that an offer *o'* is *better* than an offer *o* (denoted as $o \mathbf{f} o'$) if *o'* is more *partially complete* than *o* and *c*(*o'*) > *c*(*o*). Consequently, we only favour solutions that are closer to be complete and that improve the existing candidate solution. Figure 5-3 shows the pseudo-code for the optimisation algorithm.

```
Function optimise (O,B)
o = <∅ , ∅,…,∅>
While termination_criterion not reached
    randomly select an attribute a_i in the condition of a bidding rule.in B
    randomly obtain a new value o_i' for the chosen attribute.
    obtain o' as the result of executing the set of bidding rules over <o_1,…,o_i', …,o_n>
    if o f o' then o = o'
end_while
```

If *O* is *complete* **return** with success

else return with failure

Figure 5-3. Optimisation offer algorithm

If the process ends with success, a complete *offer* is obtained and *Quotes* automatically submits it as an indicative offer. Assuming $w^p = w^b = 1$ the offer is likely to be close to a win-win agreement and thus reducing the number of buyer-provider interactions. However the negotiation process may progress further since the submitted offer is set as indicative.

5.2.3 Example

Consider a buyer request consisting of 10 units (non-mandatory) of memory SIMMS of exactly 128 Mb at 133 Mhz. Furthermore, the buyer is not willing to pay more than 10000. Additionally, we assume there is a provider of SIMM Modules whose bidding rules are shown in Figure 5-4.

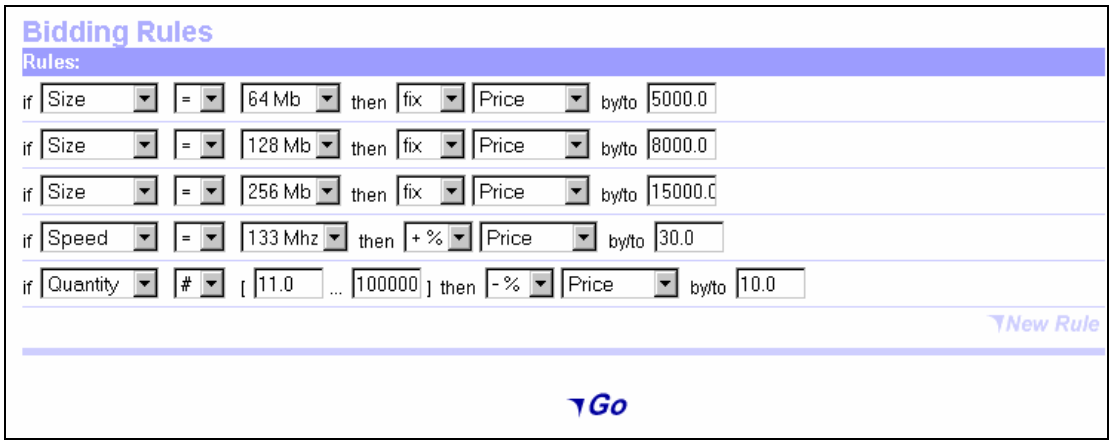

Figure 5-4: Bidding rules example

Table 5-1shows the evolution of an offer through the optimisation algorithm execution. Since we assume the provider has not specified any preferences (and thus it is not possible to compute his scoring), we only show the evolution of the buyer's score.

Iteration	Price	Quantity	Size	Speed	Package	$s^b(o)$
0	∅	∅	∅	∅	∅	--
1	∅	∅	∅	133	∅	--
2	∅	∅	∅	133	Branded	--
3	∅	8	∅	133	Branded	--
5	10400	8	128	133	Branded	35%
12	9960	19	128	133	Branded	66%
36	9960	11	128	133	Branded	80%

Table 5-1: Evolution of an automatically generated offer

The algorithm quickly assigns values for speed, package and price. Speed and price only admit one value each, and the value selected for Package has no effect on the buyer's score. At fifth iteration, the algorithm has managed to obtain a complete offer. At 12th iteration, a random move of quantity enables the application of a volume-based discount rule (last bidding rule in Figure 5-4). This improves the buyer's score since it fixes the price below 10000. From this point, the only allowed movement is to decrease the number of units to match as much as

possible the quantity required by the buyer. The algorithm successfully terminates with the offer <9960,11,128,133,Branded>, which guarantees the price to be below 10000 offering only an additional unit.

6 Negotiation

So far potential providers have been requested for offers and even some of them have already submitted either manual offers or automatically-generated indicative offers. The process now enters into a negotiation phase. Negotiation is conducted through multiple, simultaneous, structured dialogs. Each dialog is established between the buyer and a single provider and it is ruled by a negotiation protocol. Buyer actions can be: offer acceptance, offer rejection, counter-offer submission, and request for firm (offer). Provider actions are limited to the submission of either firm or indicative offers. Nevertheless, a provider can negotiate for an RFQ both by submitting offers sequentially (following a negotiation dialog) or in parallel (offering different alternatives to the buyer).

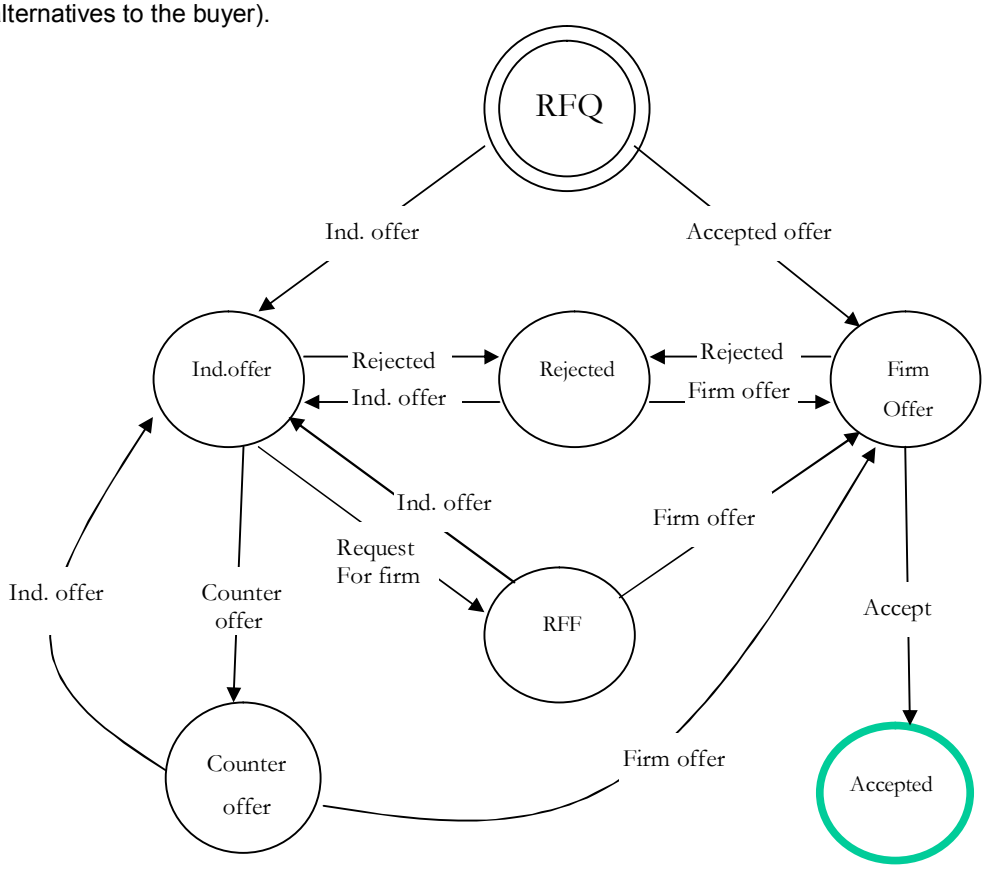

Figure 6-1: Negotiation finite state automaton.

The following sequence illustrates a typical negotiation.

1. The buyer submits an RFQ asking for *service1* and *service2*.
2. *Quotes* identifies a potential provider and automatically constructs two indicative offers on his behalf based on his bidding rules: *offer1* for *service1* and *offer2* for *service2*.
3. The buyer evaluates *offer1* and submits a counter-offer asking for lowering the price.
4. The provider responds with an extension of offer1 so that it also includes an offer for service2. In other words, he is accepting a price reduction provided that the buyer acquires both *service1* and *service2*.
5. The buyer rejects *offer2*.

6. The buyer evaluates the modified *offer1',* agrees with it and requests a firm offer.
7. The provider concedes the firm offer.
8. The buyer accepts the firm offer and therefore the negotiation successfully finalises.

Formally, negotiation can be described by means of a finite state automaton where messages between a buyer and a provider define the transition between negotiation states. Figure 6-1 depicts it.

Notice that the buyer holds a one-to-many negotiation with providers. *Quotes* allows that multiple dialogs take place in parallel between the buyer and the providers competing for some RFQ. And at the same time, each provider is allowed to simultaneously hold several dialogs with the buyer whenever each dialog corresponds to a different offer.

As we have previously stated, negotiation assumes an RFQ has been sent, so negotiation formally starts with a provider sending an offer. This initial state switches either to an *Indicative Offer* state (reached when an "Ind. Offer" message is sent) or to a *Firm Offer* state. Intermediate states are Counter-offer, RFF (Request for Firm), and Rejected.

Negotiation ends when the buyer accepts a firm offer. This corresponds to an *Accepted* state (in black). Another final state is *Failure*, which can be reached from any other state (although it has been omitted for the sake of clarity). In addition to this, neither the buyer nor any provider is directly responsible for generating the transition to this state: *Quotes* causes it as a consequence of an internal action (such as a system exception, a timeout, etc.).

7 Auctions

In order to start an auction, an RFQ is required together with a selection of qualified providers. Only selected providers will be invited to participate in a buyer-customized auction event. Auctions in *Quotes* include several parameters [24]:

- When to clear the auction. Available options are: to be cleared by buyer, when a specific time is reached (thus, an ending date must be provided), or when no bids have been received for a specified time.
- Begin date: when to launch the auction. Prior to the auction start, its terms are accessible to invited providers.
- First bid unconstrained. It allows a provider to bid for the first time with an offer that does not necessarily overbid the best bid. As a consequence the winner remains the same but auction competitiveness is increased.
- What information is revealed to bidders. Three different levels of information visibility concerning providers' identities can be established: none, nickname or full identity.
- What bidding information is revealed to bidders while the auction stage (highest bid, all bids, or none).
- Ranking revelation. If enabled, each bidder can visualise the relative position of his bids.
- Voyeur providers. Whether providers that have not submitted any bid can see information about the auction evolution or not.
- Maximum number of auction extensions and time per extension.
- Extension detection time. Extensions are launched when bids are submitted within this period of time.

Additionally, bidding constraints can be imposed on providers, lots, items, or attributes:

- Constraints on providers affect bidding capabilities for certain bidders, as for example, different starting values for attributes, fixed bids for specific items, etc.
- Constraints on lots and items fix the maximum/minimum bid increment/decrement.

- Constraints on attributes specify starting and target values as well as increment/decrement constraints.

In the most general case, *Quotes* supports combinatorial multi-attribute reverse auctions [25]. This allows providers to directly bid for bundles of items. They are convenient for providers that have non-additive values for bundles of items (providers' offers for bundles of items may be better than for their separate items). Furthermore, they allow providers to express complementarities (interdependencies) over the requested items to avoid the risk of being awarded incomplete bundles. Notice also that providers are allowed to place multiple bids for bundles of items. Figure 7-1 and Figure 7-2 depict how buyers and providers visualise the very same combinatorial auction.

Figure 7-1. Buyer façade of an ongoing auction

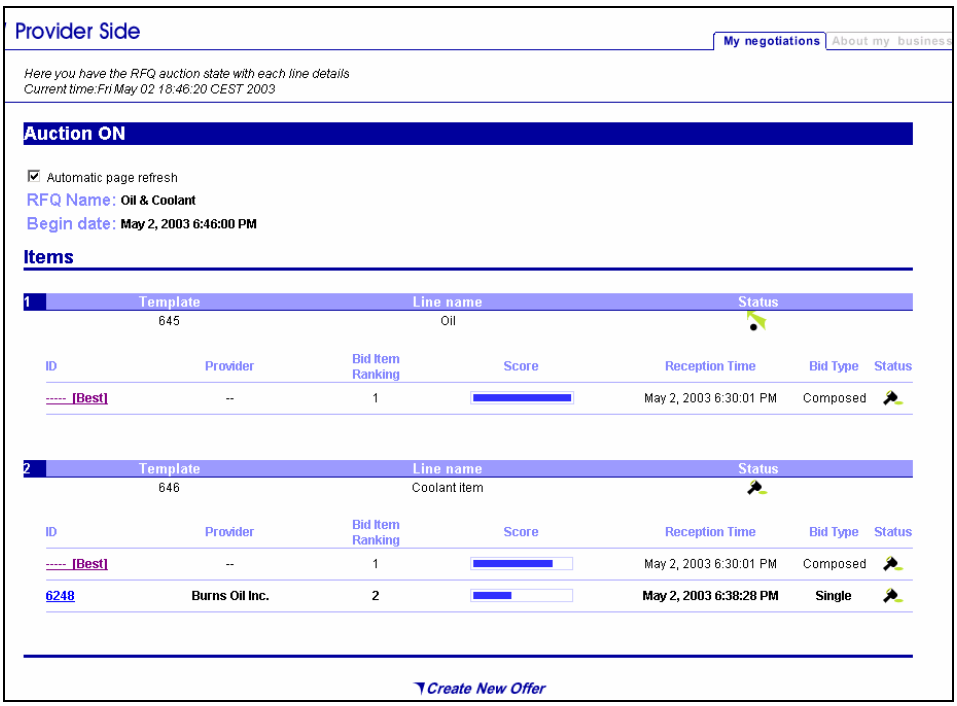

Figure 7-2. Provider façade of an ongoing auction

8 Selection of best offers

Ideally, both negotiations and reverse auctions should finish with the award of items to the best offers. Nevertheless, when dealing with complex (multi-item, multi-attribute) negotiation events, this is not a straight forward task. In order to assist buyers, *Quotes* provides a decision support module (the so called RFQ-offer matching module) for offer and bid assignment endowed with ranking and comparing facilities. They are based on scoring criteria fully detailed in subsection 8.1.

Furthermore, *Quotes* incorporates an additional decision support module that determines the best offers in combinatorial negotiations whose implementation is presented in subsection 8.2.

An interesting issue that is beyond the scope of this paper is the problem of *preference elicitation*, i.e. discover and/or quantify the buyer's preferences over a product configuration ([5], [29]).

8.1 Scoring criteria: Fuzzy matching

Quotes provides both buyers and providers with a fuzzy matching module that allows them to score the negotiation messages (RFQs and offers) they receive based on their own preferences. In this manner, a buyer can order incoming offers from different providers in the same way that a provider can order incoming RFQs from different buyers. This is particularly useful when dealing with a large number of negotiation messages because the more interesting a message the earlier it should be identified and answered. And the sense of interest is extracted from buyers' and providers' preferences.

Most commercial offer selection tools are based on simple implementations of Multi attribute utility theory (MAUT, [16]). We extend these techniques by incorporating fuzzy functions in the RFQ-offer matching module (see [22]). At this aim, we firstly represent both requested and offered attribute values as fuzzy functions. Secondly, this pair of fuzzy functions are combined and defuzzyfied (by computing the supremum of their intersection) in order to obtain a scoring (a degree of matching) at attribute level. These crisp values are then weighted with the importance of each attribute so that the scoring for an item is obtained. Finally, the scorings for all items in a message are aggregated to yield a total scoring value.

After the work by Baas and Kwakernaak [4] fuzzy functions have been mostly applied as triangular fuzzy numbers representing preferences. On the other hand, preferences over continuous attributes can be modelled by linear functions in the [0, 1] interval. We go further in three aspects: we parametrise fuzzy functions' support (positive values), we model interval preferences with trapezoidal fuzzy functions, and we allow values in the central part of the preferred intervals to increase or decrease linearly. The three extensions are determined based on users' preferences.

8.1.1 Fuzzy functions

As we have previously seen, both buyers and providers define their preferences. On the one hand, buyers specify their preferences when assigning values to RFQ item attributes. On the other hand, providers specify their preferences when defining preferred values in their product profiles (see subsection 4.1). Internally, these preferences are represented as fuzzy functions.

A remarkable feature of fuzzy logic [11] is its ability to handle the concept of relative truth of one proposition "x is P" through the specification of a membership function that represents predicate P. In our case, we can see a preference as a predicate, and the degree of truth of the proposition as the degree of the preference satisfaction of an offered value x. For example, consider a domain of four quality values ('low', 'medium', 'high', and 'luxurious'), a buyer requesting 'luxurious' quality, and a provider offering 'high' quality. In this case we cannot state that the provider satisfies completely the buyer's preference, but, since 'high' is close to 'luxurious', satisfaction should neither be zero.

In *Quotes*, this degree of satisfaction is computed by means of a fuzzy function (also known as membership function of the fuzzy set defined by predicate P), which is defined for each preference over each item attribute. This section shows how these functions are defined for different types of preferences.

These fuzzy functions have been designed taking into account the considerations listed below. Most considerations constitute design guidelines whose application should result in fuzzy functions modelling intuitive human satisfaction (i.e., buyers' and providers' intuitive evaluations). Additionally, three different examples of the resulting fuzzy functions are presented.

Design guidelines for attribute satisfaction fuzzy functions

1) Value preferences for each item attribute define a fuzzy function, whose universe is defined to be the attribute value domain specified in the corresponding item template (offered and requested values do belong to the same domain).

2) Satisfaction values belong to the [0,1] interval. When an offered attribute value o_i coincides with a preference, its satisfaction degree is 1. Otherwise, it will take decreasing values (down to 0) as o_i goes further away from the preference inside the domain.

   This implies that satisfaction must behave asymptotically when the domain is not limited (that is, for *Any Number* template attribute type).

3) If no slope information is associated to a preference (i.e. neither 'LIB -Less is better-' nor 'MIB -More is better-' slopes have been defined), satisfaction must behave symmetrically (and, by following the previous consideration, must assign satisfaction values that decrease proportionally with the distance to the preferred values). For example, if the preferred value is 4, both 3 and 5 offered values should take the same satisfaction degree.

4) Multiple values within a preference mean that they are different preferred options (i.e., the preference would be satisfied with any of the values). Thus, all values in a Flat preference interval (or preference set) take the maximum satisfaction:

   $$\forall\, o_i \in \text{Preference interval or Preference Set},\ \text{Satisfaction}(o_i) = 1$$

5) When preferences are specified through intervals with non-Flat slopes (i.e., MIB and LIB), satisfaction values range from $\propto$ to 1, being $1 \geq \alpha \geq 0$ the minimum satisfaction degree that preferred values can take inside the preferred interval.

   $$\forall\, o_i \in \text{Preference MIB or LIB interval},\ \text{Satisfaction}(o_i) \geq \alpha$$

   In our implementation α has been set to 0.5. The value has been fixed because of usability reasons (to relieve users from specifiying intrincate non-intuitive customisation values).

6) Outside non-Flat intervals, those values 'close enough' to the preferred side (i.e., maximum value in an MIB interval or minimum value in an LIB interval) would also take satisfaction values inside $[\alpha, 1]$. By 'close enough' we consider values within a neighbourhood of the interval (computed as a percentage β of the interval length).

   Formally, if the preference interval is defined as $[\min_p, \max_p]$ over an attribute whose domain is defined by $[\min_d, \max_d]$ (where $\min_d$, $\max_d$ are infinite for *Any value* domains), we distinguish two cases:

   if preference interval has an MIB slope then:

   $$\forall\, o_i \in (\max_p, \min(\max_d, \max_p + (\max_p - \min_p) \cdot \beta/100)]\ \ \text{Satisfaction}(o_i) \geq \alpha$$

   Otherwise, if preference interval has an LIB slope then

   $$\forall\, o_i \in [\max(\min_d, \min_p - (\max_p - \min_p) \cdot \beta/100), \min_p)\ \ \text{Satisfaction}(o_i) \geq \alpha$$

   The rest of values always take satisfaction values under α.

   MIB: $\forall\, o_i \notin (\min_p, \min(\max_d, \max_p + (\max_p - \min_p) \cdot \beta/100)]\ \ \text{Satisfaction}(o_i) \leq \alpha$

   LIB: $\forall\, o_i \notin [\max(\min_d, \min_p - (\max_p - \min_p) \cdot \beta/100), \max_p)\ \ \text{Satisfaction}(o_i) \leq \alpha$

   Considering neightbourhood intervals is aimed to soften abrupt decreasing (discontinuities) in satisfaction degrees. The value of β been fixed to 10% for usability reasons.

7) Symmetry does not apply for non-Flat preference intervals. In this manner, satisfaction degrees for offered values falling on the right side outside an MIB interval should decrease with a smoother slope than left-sided values.

   The same applies for LIB intervals: left sided values outside the interval should decrease with a smoother slope than right sided values.

8) In case an offer contains multiple offered values (o_{i_1},.., o_{i_m}) for an attribute, the satisfaction of the offer for this attribute must correspond to the best offered value satisfaction:

$$\text{Satisfaction}\ (o_{i_1},..,\ o_{i_m}) = \max\ (\text{Satisfaction}(o_{i_1}),...,\text{Satisfaction}(o_{i_m}))$$

   As previously seen, offers are restricted to take on a single value per attribute. Nevertheless, this consideration has been included so that RFQs can be also scored against provider's preferences using the same scoring computation method. Thus, for example, if a provider's preference for an attribute is 4 and an RFQ requests the attribute values to be 3 or 4, the computed satisfaction must be 1.

Keeping these considerations in mind, we present tree different examples of fuzzy functions defined for different types of attribute domains and preferences (the rest of cases result in analogous fuzzy functions, and thus can be naturally inferred from these ones):

Example 1: Fuzzy function definition for an attribute whose domain is 'range of numbers' and its preference type is 'single'

Figure 8-1 shows the fuzzy function defined for a single preference value of 10 whose template defines its domain as the [-20, 20] range of numbers. The function is generated so that satisfaction is 1 at 10 and it decreases down to 0 at the further extreme of the domain interval (which in this case is -20). Since symmetry must be respected (no slope can be defined for single preferences), the same decreasing slope is applied to values at the right side of the preferred value. In this manner, the maximum value for the domain –which is 20-- will take a satisfaction degree of 0.66.

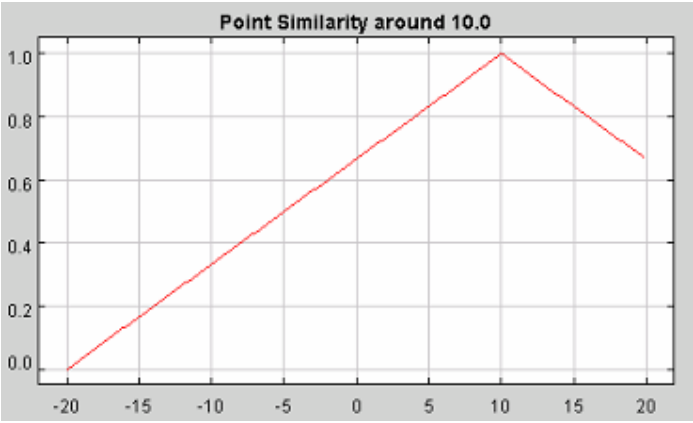

Figure 8-1: Fuzzy function for a single preference inside a range of numbers domain.

Example 2: Fuzzy function having domain attribute type = ordered set of labels and preference type = flat interval.

This second example corresponds to a fuzzy function whose domain is an ordered set of labels {QA, QB,.., QI} and whose preference is the flat interval [QC, QE] (that is, all QC, QD, QE values are equally preferred) (see Figure 8-2). In this case, labels inside the interval have satisfaction equal to 1, and as in the previous example, satisfaction values decrease for labels down to 0 at QI (the further label). Again, labels on the left of the preferred interval decrease with the same slope than the right side, so QA is assigned a satisfaction degree of 0.5.

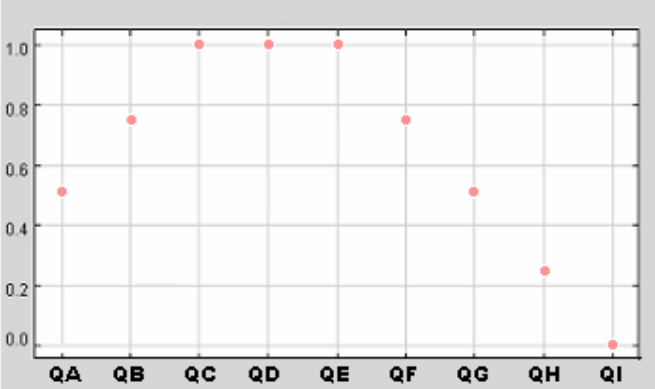

Figure 8-2: Fuzzy function for a flat interval preference and ordered set of labels domain

Example 3: Domain attribute type: any number, preference type: MIB interval.

Last example corresponds to an MIB (more is better) interval preference ([-20,20]) considering a non-restricted numerical domain (that is, any number is allowed). Satisfaction values for this domain are computed using a 4-piece formula considering α=0.5 and β=10 (see Figure 8-3):

(1) $\forall\ o_i \in$[-20, 20], Satisfaction(o_i)= $0.5 + \frac{o_i + 20}{80}$

(2) $\forall\ o_i \in$[20, 24], Satisfaction(o_i)= $1 - \frac{o_i - 20}{8}$

(3) $\forall\ o_i >$ 24, Satisfaction(o_i)= $e^{-\frac{(o_i-\mu)^2}{2\cdot\sigma}}$ where:

σ = spread $\cdot$ (Length[‡1] ([-20,20] $\cup$ [20, 24]))^2 $\cdot$ ρ

μ = 24 , spread= 2, ρ = 2

(4) $\forall\ o_i <$ -20, Satisfaction(o_i)= $e^{-\frac{(o_i-\mu)^2}{2\cdot\sigma}}$ where:

σ = spread $\cdot$ (Length ([-20,20] $\cup$ [20, 24]))^2 $\cdot$ ρ

μ = -20, spread= 2, ρ = 1

First piece (1) corresponds to the satisfaction inside the preference interval, and increases linearly from 0.5 at the minimum up to 1 at the maximum.

Second piece (2) corresponds to the 10% additional interval, which in this example is 4 units long, and decreases from 1 at the maximum of the preference interval (i.e., 20) down to 0.5 at the maximum of the additional 10% interval (i.e., 24).

Third and forth pieces define satisfaction degrees for the domain values outside the previous intervals. Since the domain is not restricted, we use probabilistic functions that decrease asymptotically along the x axis. Mean (μ) values are set to be the preference interval point that is closer to the described side, (assuming the interval to be the union of both defined preference and additional neighbourhood intervals). In this manner, for the right side (3) mean is the maximum of the additional neighbourhood interval (i.e., 24) and for the left side (4) it is the minimum of the defined preference interval (i.e., -20).

Regarding variance (σ) values, they are assigned as the product of a spread value, a ρ value, and the square value of the length of the interval. Spread is fixed to 2 for both sides, but since symmetry is not desired for an MIB interval, ρ takes different values for each side. More

[‡1] Length([min,max])= max-min being [min,max] an interval

concretely, ρ is set to 2 on the right side (3) and to 1 on the left side (4), so that variance in the side closer to the more preferred values is twice the variance of the other side.

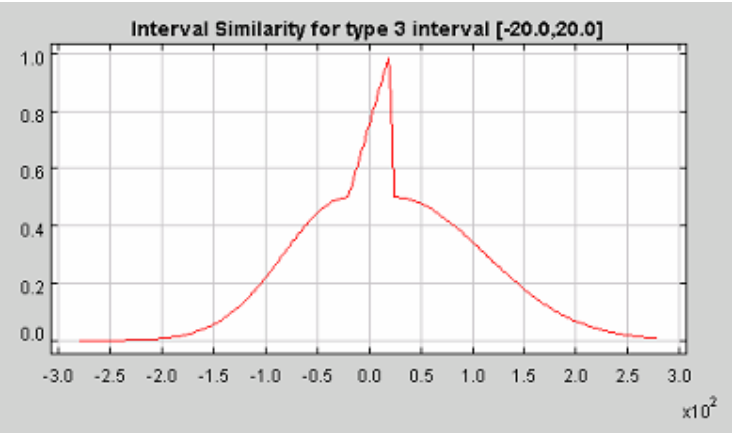

Figure 8-3: fuzzy function for an MIB interval preference and any number domain.

8.1.2 Offer evaluation

An RFQ specifies buyer's preferences over a set of items that describe products (or services) in terms of attributes. *Quotes* scores providers' offers for an RFQ so that the buyer can have an ordered list of offers, ranking first those offers that best satisfy his preferences. Therefore, in order to score an offer in relation to an RFQ, it is necessary to compute satisfaction values at the level of attributes and propagate them to the item level and up to the offer level (since an offer may encompass several items).

In a more formal manner, given an offer *o* composed of m items o^j *j*=1..m, we describe an offer item *j* as the n-tuple of the form $<o^j_{j_1}, o^j_{j_2},\ldots, o^j_{j_n}>$ where $o^j_{j_i}$ represents the i-th attribute value of item *j*.

In order to compute attribute scoring, *Quotes* creates the above-described fuzzy functions. Offered attribute values, which are always single, are then used to compute satisfaction degrees:

(5) $$\text{scoring}(o^j_{j_i}) = \text{Satisfaction}\ (o^j_{j_i})$$

Attribute satisfaction degrees are then weighted to obtain offer item scoring. Since preferences are RFQ attribute values, weights $w^j_{j_i}$ are taken to be the importance the buyer assigned to an attribute *i* in the *j*-th RFQ item (see subsection **¡Error! No se encuentra el origen de la referencia.**).

$$\text{scoring}(o^j) = \sum_{i=1}^{j_n} \frac{\text{scoring}(o^j_i) \cdot w^j_i}{j_n}$$

Finally, we compute offer scoring as a weighted combination of item scorings.

$$s^b(o) = scoring(o) = \sum_{j=1}^{m} \frac{scoring(o^j) \cdot w^j}{m}$$

where w^j stands for the weight of the *j*-th item.

In the current implementation, all item weights are set to 1 and scoring values are translated from the [0,1] to the [0,100] range.

8.1.3 RFQ evaluation

Similarly to offer scoring functionality, *Quotes* facilitates providers an RFQ scoring computation that prioritises buyers' requests. Again, fuzzy functions are generated for each attribute, using its domain as defined by templates along with preferences as providers' preferred values (see subsection 4.1).

The main difference appears in the computation of the formula number (5). The reason is that attribute values in RFQ items are not restricted to be single, so that value intervals or sets must be evaluated against these fuzzy functions. In this case, *Quotes* assumes attribute scoring to be the best satisfaction degree of all values *x* defined in the attribute value range rfq_i^j.

$$scoring(rfq_i^j) = \underset{\forall x \in rfq_i^j}{MAX} (Satisfaction(x))$$

8.2 Winner determination in combinatorial scenarios

Allowing providers to bid on combinations of goods has the interesting feature of enhancing economic/service efficiency. Thus suppliers may offer more competitive bids with the perspective of gaining more business [23]. However, the determination of an optimal winner combination is a complex problem which, excluding very small instances, cannot be manually solved with common data analysis tools. Thus, consider the decision problem faced by a buyer when negotiating with providers. While in direct auctions, the items that are going to be sold are physically concrete (they do not allow configuration), in a negotiation event involving multiple, highly customisable goods, buyers need to express relations and constraints between attributes of different items. On the other hand, it is common practice to buy different quantities of the very same product from different providers, either for safety reasons or because offer aggregation is needed to cope with high-volume demands. This introduces the need to express business constraints on the number of providers and the amount of business assigned to each of them. Not forgetting the provider side, providers may also impose constraints or conditions over their offers. Offers may be only valid if certain configurable attributes (e.g. quantity bought, delivery days) fall within some minimum/maximum values, and assembly or packing constraints need to be considered. Once the buyer collects offers, he is faced with the burden of determining the winning offers. The problem is essentially an extension of the combinatorial auction (CA) problem, which can be proven to be NP (Rothkopt 1995). Hence that it would be desirable to relieve buyers from solving such a problem. However, although the application of combinatorial auctions (CA) to e-procurement scenarios (particularly reverse auctions) may be thought as straightforward, the fact is that we identify multiple new elements that need to be taken into consideration.

Current CA reviewed do not model these features with the exception of [6] and [26], where coordination and procurement constraints can be modelled. The rest of work focuses more on computational issues (CA is an NP-hard problem [23]) than in practical applications to e-procurement. Suppose that we are willing 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 supplier 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 chair's 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 means of modelling these natural constraints, solutions obtained are seen as mathematically optimal, but unrealistic and with a lack of common sense, thus obscuring the power of decision support tools, and preventing the adoption of these technologies in actual-world settings.

In this section we present iBundler, a decision support component acting as a combinatorial negotiation solver (solving the winner determination problem) for both multi-item, multi-unit negotiations and auctions. Thus, the component can be employed by both buyers and auctioneers in combinatorial negotiations and combinatorial reverse auctions [25] respectively.

Furthermore, it extends current combinatorial auction models by accommodating both operational constraints and attribute-value constraints.

8.2.1 Desiderata

Next we detail the capabilities required by buyers in the kind of negotiation scenario outlined above. The requirements below are intended to capture buyers' constraints and preferences and outline a powerful bidding language for providers:

1) *Negotiate over multiple items*. A negotiation event is usually started with the preparation of a request for proposal (RFQ) form. The RFQ form describes in detail the requirements (including attribute-values such as volume, quality specifications, dates as well as drawings and technical documentation) for the list of items (goods or services) defined by the negotiation event.

2) *Offer aggregation*. A specific item of the RFQ can be acquired from several providers simultaneously, either because not a single provider can provide with the requested quantity at requested conditions or because buyer's explicit constraints (see below).

3) *Business sharing constraints*. Buyers might be interested to restrict the number of providers that will finally trade for a specific item of the RFQ, either for security or strategic reasons. It is also of usual practice to define the minimum amount of business that a provider may gain per item.

4) *Constraints over single items*. Every single item within an RFQ is described by a list of negotiable attributes. Since: a) there exists a degree of flexibility in specifying each of these attributes (i.e. 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. An example of this can be the following: suppose that the deadline for the reception of certain item A is two weeks time. However, although items may arrive any day within two weeks, once the first units arrive, the rest of units might be required to arrive in no more than 2 days after.

5) *Constraints over multiple items*. In daily industrial procurement, it is common that accepting certain configuration for one item affects the configuration of a different item, for example, when dealing with product compatibilities. Also, buyers need to express constraints and relationship between attributes of different items of the RFQ.

6) *Specification of providers' capacities*. Buyers cannot risk to award contracts to providers whose production/servicing capabilities prevent them to deliver overcommitted offers. At 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. The features of the language below are intended to capture providing agents' constraints and preferences.

7) *Multiple bids over each item*. Providers might be interested in offering alternate conditions/configurations for a same good, i.e., offering alternatives for a same request. A common situation is to offer volume-based discounts. This means that a provider submits several offers and each offer only applies for a minimum (maximum) number of units.

8) *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.

9) *Multi-unit offering*. Each provider needs to specify that they will only participate in trading if a minimum (maximum) amount of business is assigned to him.

10) *Homogeneous combinatorial 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*. Packing units are also a constraint, in the sense that it is not possible to serve an arbitrary number of units (e.g. a provider cannot sell 27 units to a buyer because his items come in 25-unit packages). Thus providers are required to be capable of specifying the size of packing units.

12) *Complementary and exclusive offers*. Providers usually submit XOR bids, i.e., exclusive offers that cannot be simultaneously accepted. Also, there may be needed that an offer is only selected if another offer is also selected. We refer to this situation as an AND bid. This type of bids allows specifying volume-based discounts. For example, when some pricing is expressed as a combination of a base price and a volume-based price (e.g. the first 1000 units at €2.5 per unit, and €2 per unit for the rest).

Obviously, many more constraints regarding pricing and quantity can be considered here. But we believe these faithfully address the nature of the problem. Actually, iBundler has been applied to scenarios where some of these constraints do not apply while additional constraints needed to be considered. This was the case of a virtual shopping assistant, an agent that was able to aggregate several on-line supermarkets and optimise the shopping basket. To do so, it was necessary to model the fact that delivery cost depends on the amount of money spent at each supermarket.

8.2.2 Implementing winner determination

Consider the problem faced by a buyer aiming at choosing the optimal set of offers sent over by providers taking into account the features of the negotiation scenario described above. The problem is essentially an extension of the combinatorial auction (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. 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 ([10], [13] and [25]). 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, the fact is that the CA problem is an instance of the multi-dimensional knapsack problem MDKP (as indicated in [12]), a mixed integer program well studied by the operation research literature. It is not surprising that, as reported in [2], many of the main features of these problem specific new algorithms are rediscoveries of traditional methods in the operations research 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 [26] with side constraints [27]. 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 the iBundler decision support service has been modelled and implemented as a mixed integer programming (MIP) problem. We have implemented two versions: one using ILOG CPLEX 7.1 in combination with SOLVER 5.2; and another one using iSOCO's Java MIP modeller that integrates the GLPK library [17]. In both cases it takes the shape of a software component.

8.2.3 Winner determination performance

Since combinatorial auction solvers are computationally intensive, a major issue is whether our service is to behave satisfactorily in highly-demanding trading scenarios. At this aim, we have conducted some empirical measures on the performance of iBundler. Figure 8-4 shows how *it* behaves when solving negotiation problems as the number of bids, the number of items, and the items per bid increase. Notice that in order to run our tests we devised a customisable generator of data sets which artificially created negotiation problems by wrapping the solution with noisy bids. In this way, not only were we able to measure the performance of iBundler, but also to automatically verify the sound behaviour of the service in a large variety of negotiation scenarios demanding the many capabilities of the service.

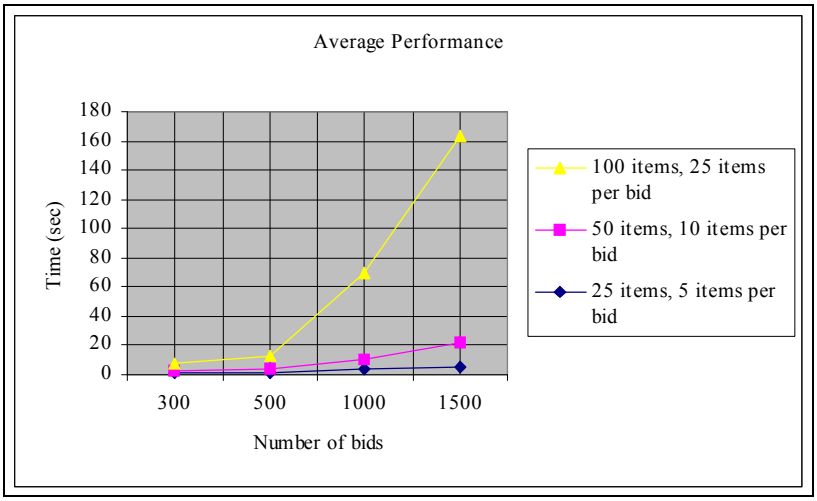

Figure 8-4. Average performance of the iBundler decision support component

9 Multi-stage sourcing events

Once the different tasks that are usually involved in sourcing events have been detailed in previous sections, this section aims to describe *Quotes* support for multi-stage negotiations [9].

A typical situation in industry is the specification of sourcing events that involve several stages. Depending on whether the product features are not fully known by the buyer, the product's complexity and/or the number of potential providers, sourcing professionals design the sourcing process as a sequence of steps (stages) with various objectives.

Consider the negotiation of a new frame contract for raw material:

- Typically the sourcing event will start with new providers being invited to an RFI (request for information) with the goal to qualify them. An RFI negotiation is performed analogously to RFQ negotiations (it requires an RFI elaboration task similar to RFQ elaboration as well as providers' responses). The result of this stage is the determination of winning (qualified) providers. Notice that already qualified providers will not be invited to this stage.
- At the end of the RFI stage, qualified providers are then invited to an RFQ with the aim of negotiating required product features and transaction terms. On the providers' side, offers are either manually or automatically built as responses to the received RFQ by assigning values to the product features and transaction terms. Thereafter, the buyer can conduct simultaneous one-to-one negotiations as part of the one-to-many negotiation process.
- This negotiation phase may end up with the buyer accepting some offer(s) or may be continued as a reverse auction stage with a subset of the most competitive providers. At this point, the buyer may opt for launching a reverse auction with the objective of increasing comptetitiveness along those product features and transaction terms whose negotiated values remain unsatisfactory.

As we have observed in current market solutions there is little support for creating customized sourcing events as compositions of negotiation processes. *Quotes* allows buyers to undertake multi-stage sourcing events by means of stage switching as described above. Moreover RFI, RFQ, and auction stages can be interleaved (and repeated) by the buyer at will. Conditions and product requirements can be changed from stage to stage, for instance, attribute importance adjustments or new product requirements. Offers can be migrated from stage to stage as long as they still satisfy these new requirements. Additionally, this will help the buyer to determine providers that should be excluded out of the next stage (although in some cases providers may still be invited but only to compete for a subset of a multi-item RFQ). Obviously, new providers can always be invited to a next stage.

Quotes controls the status of the whole process as well as the state of each provider, thus communicating to them their inclusion or exclusion in the next stage.

Figure 9-1 shows how *Quotes* implements multi-stage sourcing events.

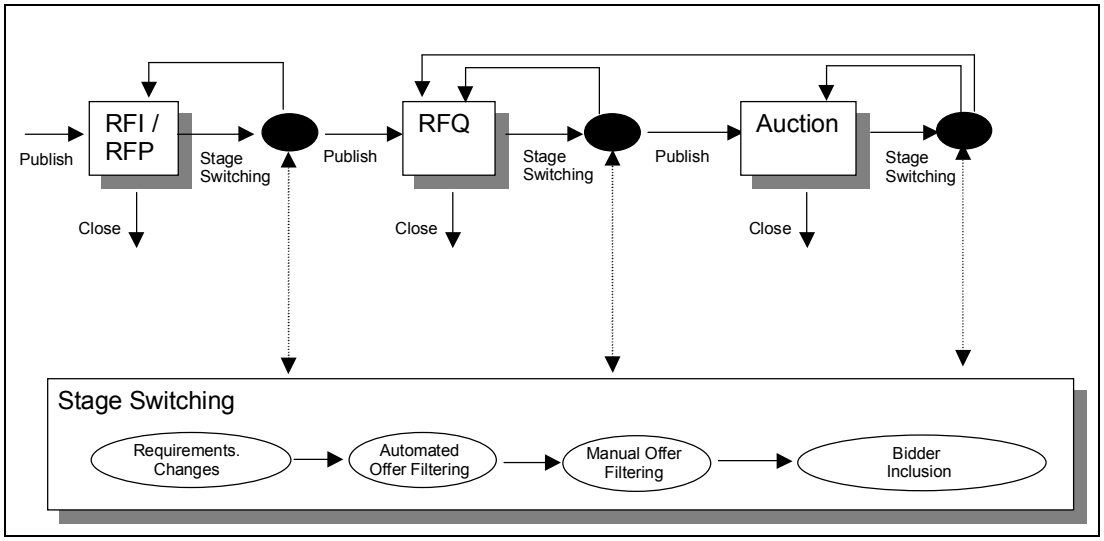

Figure 9-1. Multi-stage sourcing at Quotes

A sourcing event starts with the definition of a new RFx (RFI, RFP, RFQ and reverse auction mechanism). The RFx is published and executed. When the RFx finishes the buyer can determine winners and close the stage, thus finalizing the sourcing event. Alternatively the buyer can undergo a *switch-to-stage* procedure. By doing so, *Quotes* creates a new RFX stage based on the current one. The buyer can then change the requirements (add or remove items, add or remove attributes) and determine the negotiation mechanism (RFI, RFP, RFQ or auction). *Quotes* automatically filters out (rejects) offers that do not satisfy newly requirements and allows the buyer to manually cancel others. Finally, the buyer decides which providers continue with the next event and has the choice of inviting new ones.

10 Results

Quotes has been successfully applied to several real-life negotiation scenarios of varying complexity (from single line to multi-line, multi-attribute) and economic value (few thousand € to frame contracts of several million €). Next, we analyse the outcomes of the application of *Quotes* to real-world procurement situations along the lines of [14].

Table 10-1 provides some figures about four different negotiation event types conducted with Quotes by companies in two different sectors: Oil & gas and Food producers & processors. On average, the savings obtained by *Quotes* users ranged from 9,85% in RFQs for Food producers to 15,33% in reverse auctions set up by Oil & gas companies.

Event Type	Quantity	Total amount	Average amount	Average savings	Average num. of providers
Reverse auction	34	10.197.380 €	299.923 €	15,33%	8,35
RFQ	4	6.342.400 €	1.585.600 €	9,85%	4,67
RFP	1	0 €	0 €	0,00%	19,00
RFI	28				
TOTAL	67	16.539.780 €	1.885.523 €	8,39%	10,67

Table 10-1: Negotiation events conducted with Quotes in Oil & gas and Food producer & processor sectors.

Quotes has been also used by Steel companies to conduct a total number of 1593 sealed-bid auctions. Unfortunately, these auctions where entirely managed by end users and we lack of the corresponding results. In total, up to twenty two different companies have been able to negotiate electronically by using *Quotes.* These companies belong to a wide range of sectors (in addition to the ones we have mentioned, namely, of oil and gas, food producers and processors, and steel): tobacco, forestry and paper, banks, beverages, support services, household goods, electronics and electrical equipment, logistics, and support services.

Until now, *Quotes* has been used to negotiate for both direct and indirect goods as well as services. Nevertheless, negotiations for indirect goods prevailed. Although these negotiations considered price as a main factor, further attributes (e.g. delivery time, payment method, etc.) were too included. Thus, in fact, negotiations for indirect goods were run as multi-attribute negotiations. On the other hand, a significant number of negotiations for direct goods and services have also been conducted. Their particularity is that they did also require the use of *Quotes*' multi-attribute negotiations. While negotiations for direct goods have been mostly run by companies in the food sector, negotiations for services have been run by companies in the utilities' sector.

All negotiations and auctions did involve previously-qualified providers. Companies that run negotiations for direct goods or services were particularly reluctant to inviting new providers to their negotiations since their qualification procedures are rather strict and constitute a must for any provider to participate in any sourcing event.

Actual buyers encountered *Quotes* scoring mechanism as a simple, intuitive, and powerful way to quickly differentiate good from bad offers. However, when making final decisions or comparing similar offers they often obviated the scoring values provided by *Quotes* and fine-analysed offers by means of other evaluation functions. We are currently trying to incorporate alternative scoring functions that cope with the full needs of sourcing professionals along with preference elicitation mechanisms (although this issue remains the Achilles tendon of sourcing applications).

A common agreement was the convenience of modelling off-line negotiation processes in a natural way, without introducing inefficiencies and frictions derived from changing the "rules of the game" (that is, for example, substituting the off-line negotiation processes by on-line auctions or using negotiation artefacts that do not model previous processes). Furthermore, providers appreciated the transparency introduced by the tool (since all participants' actions can be audited).

Buyers' belief is that combinatorial offering introduces high complexity for providers to bid and to understand auction dynamics. Consequently, *Quotes*' combinatorial capabilities have been solely applied to a small set of actual-world scenarios where the buyer pre-defines valid item combinations (lots) on which providers can bid. Alternatively to combinatorial negotiations, some buyers required to enforce providers to bid for all items when auctioning lots. Observe that both sourcing strategies did intend to reduce the bidding complexity on the provider side.

The possibility of automatic offer submission is seen with interest for repetitive sourcing events in private e-sourcing platforms where providers and business rules are well known or belong to a provider qualification procedure or a frame contract. Nonetheless, the full application of such automatisms faces cultural barriers such as providers being not so keen on revealing capabilities/preferences to third parties; perception of e-sourcing tools as a hazard for sourcing professionals, etc.

Leading users suggested additional auction rules that best suited their necessities. For example, a buyer forced *Quotes* to incorporate a bidding rule that resulted in an increase of the number of participants in an auction event. This rule allowed inactive bidders to send their first bid without overbidding the best bid.

Finally, the results obtained in terms of economic outcome were no different than the promises made by e-sourcing analysts. Negotiation time was reduced from weeks to days, mostly due to the elimination of communication synchronism (telephone, fax) and administrative tasks. Price and condition benefits were also obtained. Obviously, price savings were more noticeable in auctions, but on-line negotiation also achieved price/service reduction below target, a result that increased buyer's satisfaction with the tool.

11 Conclusions

E-sourcing is becoming an established part of the business landscape [28]. We are witnessing the continuous, tightly competitive progress of e-sourcing applications in the market. Nonetheless we can still identify two major, unsatisfactorily solved issues that prevent them from supporting effective strategic sourcing, namely:

- *Capability to support sourcing processes for varying industries and businesses*. Since sourcing processes are highly dependent on each business case (because of each industry's particularities and individual businesses' practices) it is extremely complex to capture in a single product all processes and negotiation requirements of a general-purpose solution. Along this direction, perhaps the major drawback of most market solutions is the lack of support for creating customised sourcing events as compositions of negotiation processes (multi-stage negotiation processes). For instance, a sourcing event might be composed of a pre-defined sequence of auctions or an interleaving of auctions and negotiations. In this work we have presented an e-sourcing solution that does indeed support customisable multi-stage negotiations.
- *Decision support tools for strategic thinking*. Strategic sourcing is founded on the availability of powerful decision support tools. Nonetheless current vendors' solutions are lacking as to this matter. As an example, most tools share the commonality of not providing support for determining the winner in multi-item, multi-attribute negotiations and auctions. The unavailability of such support poses an intricate, combinatorial problem to professional buyers that leads them to either relinquish or opt for alternative, and less efficient, non-combinatorial protocols. In this work we have tried to make headway in providing decision support

In this paper we have tried to exemplify how the sourcing process can be highly automated, allowing companies to achieve enormous benefits: cost savings, processing time reduction, less time-to-market, and more time left to strategy. We have presented our contribution along this direction by dissecting *Quotes,* an Internet-enabled sourcing solution capable of streamlining the sourcing process. *Quotes*' main strengths can be summarised as follows:

- It allows goods and services to be represented and managed with all their attributes, overcoming rigid and unreal price-discovering approaches.
- It provides a powerful negotiation framework based on the composition of structured negotiation protocols and flexible reverse auctions.
- It provides with the necessary tools to help users manage the complex sourcing mechanisms involved in multi-item, multi-unit, multi-attribute, multi-stage negotiations.

The fundamental contribution of *Quotes* lies on the incorporation of highly valuable decision-making support functionalities targeted at spurring the transition of sourcing processes from transactional to strategy-centered. This paper describes three decision support systems that have been studied and implemented in *Quotes* with satisfactory results, namely:

- *Automated offer generation*. Providers can translate their business knowledge into *bidding* rules that allow instantaneous and automatic construction of recommended offers.
- *Scoring algorithm*. Based on the importance assigned by the buyer to each item attribute in an RFQ and his flexibility to accept offers beyond his preferences, a *fuzzy* offer-matching algorithm scores each offer and ranks it accordingly. Analogously, providers can benefit from the very same algorithm in order to discriminate incoming RFQs.
- *Computation of the optimal bid set for combinatorial negotiations*. Given a set of offers for either a multi-item RFQ or an auction, the iBundler component obtains the optimal subsets of offers according to various criteria (e.g. minimization of price, start/finish date, maximization of product quality, etc.), or a combination of them specified through scoring. Although the used algorithm is not new, the way the problem is formulated can be considered as an important contribution to research in combinatorial negotiations.

Finally, there is an important aspect affecting the successful exploitation of e-sourcing applications: negotiation design matters. Indeed there are many dimensions involved in the design of negotiation scenarios. Thus we must keep in mind that careless, faulty designs of such scenarios may eventually lead to terribly catastrophic outcomes as reported through several case studies in [18].

Acknowledgments

The authors would like to kindly acknowledge iSOCO, S. A., the former company of all of them, where they carried out the research reported in this paper.

References

[1] Aberdeen Group. (2002). "Making E-Sourcing Strategic: From Tactical Technology to Core Business Strategy".

[2] 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), pages 39-46, Boston, MA.

[3] Ariba. http://www.ariba.com.

[4] Baas; Kwakernaak (1977), "Rating and ranking of multiple-aspect alternatives using fuzzy sets", *Automatica* 13 (1977) 47-58.

[5] Bichler, M.; Kalagnanam, J.; Lee, H. S. (2002): "RECO: Representation and evaluation of configurable offers". In: Hemant K. Bhargava and Nong Ye (Eds.): Computational Modeling and Problem Solving in the Networked World: Interfaces in Computing and Optimization, Kluwer Academic Publishers, 2002.

[6] Collins, J.; Gini, M. (2001). "An Integer Programming Formulation of the Bid Evaluation Problem for Coordinated Tasks". In B. Dietrich and R. V. Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, volume 127 of IMA Volumes in Mathematics and its Applications, pages 59-74. Springer-Verlag, New York.

[7] Commerce One. http://www.commerceone.com.

[8] De Vries, S; Vohra, R.. (2001). "Combinatorial Auctions: A survey".

[9] E-negotiation group (2000) "The London Classification" at DEXA workshop on e-negotiations. London, 2000.

[10] Fujishima, Y.; Leyton-Brown, K.; Shoham, Y. (1999) "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches". In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI'99, 548–553.

[11] Godo, L.; López de Mántaras, R.(1993). "Fuzzy Logic". In Enclyclopedia of Computer Science and Technology. Marcel Dekker, Inc. Pennsylvania. Vol 29, pp 211-229.

[12] Holte, R. C. (2001). "Combinatorial Auctions, Knapsack Problems, and Hill-Climbing Search". Lecture Notes in Computer Science 2056.

[13] Hoos, H. H.; Boutilier, C. (2000) "Solving Combinatorial Auctions using Stochastic Local Search". In Proceedings of AAAI-2000.

[14] Jap. S. D. (2002) "Online, Reverse Auctions: Issues, Themes, and Prospects for the Future". *Journal of the Academy of Marketing Science*, Vol. 30, No. 4, 506-525 (2002)

[15] Kalagnanam, J.; Parkes, D. C. (2003) "Auctions, Bidding and Exchange Design". Chapter 10, in Supply Chain Analysis in the eBusiness Era, Edited by David Simchi-Levi, S. David Wu and Z. Max Shen, 2003. (forthcoming)

[16] Keeney, R. L.; Raiffa, H.; Meyer, R. (1993). "Decision Making with Multiple Objectives: Preferences and Value Tradeoffs". Cambridge, UK: Cambridge University Press.

[17] Makhorin, A.(2001) "GLPK – GNU Linear Programming Toolkit". http://www.gnu.org/ directory/GNU/glpk.html. 2001

[18] McAfee, R. P. (1995). "Auction Design for the Real World". http://www.eco.utexas.edu/faculty/McAfee/Papers/PDF/Auction.pdf.

[19] MetalSite. http://www.metalsite.com/.

[20] Reyes-Moro, A.; Rodríguez-Aguilar, J. A.; Cerquides, J.; López-Sánchez, M.; Gutiérrez-Magallanes, D.. (2002). "Quotes: a Negotiation Tool for Industrial Procurement". Fourth International Conference on Enterprise Information Systems (ICEIS 2002), April 4-6, Ciudad Real, Spain.

[21] Reyes-Moro, A.; Rodríguez-Aguilar, J.A.; Cerquides, J.; López-Sánchez, M.; and Gutierrez-Magallanes, D. (2003). "Embedding decision support in e-sourcing tools: Quotes, a case study". *Group Decision and Negotiation Journal*, vol 12, pp. 347-355, 2003.

[22] Ribeiro, R. A. (1996). "Fuzzy multiple attribute decision making: A review and new preference elicitation techniques". *Fuzzy Sets and System*s, 78; 155-181.

[23] Rothkopt, M. H.; Pekec, A.; Harstad, R. M. (1995). "Computationally manageable combinatorial auctions". *Management Science,* 44(8):1131-1147.

[24] Sandholm, T. (2000). "eMediator: A Next Generation Electronic Commerce Server". International Conference on Autonomous Agents (AGENTS), Barcelona, Spain, June 3-8.

[25] Sandholm, T. W. (2002) "Algorithm for optimal winner determination in combinatorial auctions". *Artificial Intelligence Journal* 135:1-54, 2002.

[26] Sandholm, T.; Suri, S.; Gilpin, A.; Levine, D. (2002). "Winner Determination in Combinatorial Auction Generalizations". First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy, 2002, pp. 69-76.

[27] Sandholm, T. and Suri, S. (2005). "Side Constraints and Non-Price Attributes in Markets". To appear in *Games and Economic Behavior.*

[28] Stephens Inc. (2001). "Internet Supply Chain Team. Strategic Sourcing: Applications to Turn Direct Materials Procurement into Competitive Advantage". Industry Report, January 30, 2001.

[29] Van de Walle, B.; Heitsch, S.; Faratin, P. (2001): "Coping with One-to-Many Multi-Criteria Negotiations in Electronic Markets", in Proceedings of the Second International Workshop on Negotiations in Electronic Markets: beyond Price Discovery, held during DEXA 2001 (Munich, Germany, September 2001), 747-751.