Implementation of a Competitive Market-Based Allocation of Consumer Attention Space

Report by Pieter Jan 't Hoen Sander Bohte Enrico Gerding Han La Poutré
Hoen@cwi.nl S.M.Bohte@cwi.nl Enrico.Gerding@cwi.nl Han.La.Poutre@cwi.nl

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT

In [Bohte *et al.*, 2001] a competitive distributed recommendation mechanism is introduced based on adaptive software agents for efficiently allocating the "consumer attention space", or banners. In the instance of an electronic shopping mall, the task of correctly profiling and analyzing the consumers is delegated to the individual shops that operate in a distributed, remote fashion. The evaluation and classification of consumers for the bidding on banners is not handled by a central agency as is customary, but is a distributed process where all shops bidding for a consumer partake. This allows each shop to apply its own private strategy, learning-mechanism and specific domain knowledge. Here we present a a scalable, and extensible software agent architecture and prototype for distributed market-based allocation of consumer attention space. The prototype has an architecture suitable for supporting agents in a distributed bidding application where agents run on dedicated machines for maximum computational resources. Furthermore, as an extension, the agents can operate in multiple independent markets concurrently.

1. Introduction

Electronic markets such as recommender systems, electronic auctions, and virtual shops are increasing in economic significance [Jeffrey O. Kephart, 2001]. As these systems grow in size and the number of consumers, the problem of how to maintain an efficient marketplace where consumers are correctly matched to providers increases in complexity and cost. This is especially the case for agent-based systems where heavy-weight learning algorithms and communication can become a bottleneck when using current centralized electronic applications. As such, these systems face built-in limitations that hinder their scaling.

In [Bohte *et al.*, 2001], a novel mechanism was introduced that is well suited to match consumers to providers. The mechanism is based on a market-mechanism where agents bid for placing ads for individual customers, about which some information is available. These agents can operate in a distributed setting, avoiding the bottleneck of a centralised application. It was shown in [Bohte *et al.*, 2001] that agents are able to learn the niche in the market of the advertisers they represent.

In this paper, we present an architecture and prototype of a system of software agents that extends the approach of [Bohte *et al.*, 2001] to a system of real, efficient, and distributed software agents. As a case, we use the shopping mall example application used in [Bohte *et al.*, 2001] as a scalable and efficient demonstrator of the concepts introduced therein.

In the shopping mall example, consumers visit an electronic shopping mall where banners are presented. These banners are presented only after successful bidding by the shops for the attention of the visiting consumer. We focus on an architecture and implementation where parallelism in the behavior of the agents is maximised and communication overhead is constrained to ensure minimal limitations in the performance of the system. Furthermore, we strive to keep the architecture open for future extensions.

The rest of the document is organized as follows. In Section 1.1 we outline the concepts introduced in [Bohte *et al.*, 2001] in considerable detail. In Section 1.2 focus on the specific aspects of our

prototype. We continue in Section 2 with an overview for the architecture of the prototype and the communication amongst the agents. In Section 3 we work out an example of how a consumer interacts with the shopping mall and arrives at a purchase. In Section 4 we discuss the implementation language of choice. In Section 5 we cover the applied learning mechanism for the bidding for attention space by the shops. Lastly, in Section 6 we conclude.

1.1 The CASy *System*

With the advent of electronic marketplaces, scale limitations as encountered in the brick-and-mortar world no longer apply. At the same time novel problems are encountered, like how consumers can find their way in large marketplaces.

A currently preferred solution is to have a central party propose relevant suppliers and products to a consumer. This central filtering mechanism uses knowledge of users, shops, and the product domain to determine recommendations. Such recommender systems work well within limited domains, e.g. a book or music store. However, to maintain accuracy in a large marketplace with many consumers and suppliers, the latter will need to reveal detailed and perhaps sensitive business information to the central party. Central filtering-mechanisms may thus suffer from objections from the suppliers. Furthermore, the amount of information to be processed and maintained by the central party can become unmanageable. Other approaches are needed to complement centralised systems.

In [Bohte *et al.*, 2001], a novel mechanism called Competitive Attention-space System (*CASy*) is introduced to handle the above problems, via the techniques of dynamic market-based control [Clearwater, 1995] and adaptive software agents [Weiss, 1999]. In the used example of an electronic shopping mall, the greater part of the task of matching a specific online consumer to a set of suitable shops is delegated to the individual shops. Each shop evaluates the information that is available about the current consumer: the consumer's current interests and other information that the consumer is willing to provide or has provided; e.g. keywords, product queries, and available parts of a personal profile. Based on this information and on their own domain knowledge, shops can make a monetary bid in an auction where a limited amount of consumer attention space, or banners, for the particular momentary consumer is sold. To facilitate CASy, the system can be designed as a multi-agent system where each shop is represented by a software agent. This agent executes the task of bidding for the attention of each individual consumer. The use of learning software agents allows shops to rapidly adapt their bidding strategy such that they only bid for consumers that are likely to be interested in their offerings. Furthermore, the shops can incorporate up-to-date and detailed domain knowledge into their agents, which allows for accurate and time-dependent targeting of a shop's audience, taking into account sensitive information and issues like service, quality, price, product diversity, sales, etc ....

As a concrete example of the system, we take the sale of airline tickets. A consumer visits the site and enters the desired travel plan along with additional data like, for example, restrictions in price and dates of desired departure. The shops, in this case airlines, can evaluate the desired transportation and place a bid for presenting a banner to the consumer. These bids are ranked and the higher the bid, the higher the banner is placed for presentation to the consumer. This system allows for airlines to bid strongly for customers who are likely to purchase their flight and less for unlikely consumers. For example, an airliner with only luxury seats still available will bid highly for a passenger who has a tight schedule and is willing to spend a large amount. The bid will however be low if the consumer is looking for a budget flight and not in a hurry. Through this mechanism, the consumer is presented with likely banners advertising products in decreasing order of likelihood to meet his needs.

For various basic models for on-line consumers, shops, and profiles, the feasibility of the system has been demonstrated in [Bohte *et al.*, 2001]. We note that the mechanism described is not limited to the above example of the airline and the electronic shopping mall, but can easily be applied to other domains where (pre) selection of possibilities has to be guided, like banners on more general websites, or other types of marketplaces such as job agencies or B2B platforms.

1.2 Focus of the Implementation

In [Bohte *et al.*, 2001], the *CASy* mechanism is described and simulated. It allows agents to bid for the attention space of a stream of consumers with varying profiles. In this report, we present a scalable architecture and demonstrator for *CASy* in the form of a distributed, scalable prototype.

Furthermore, the demonstrator has interesting extensions in two directions. The architecture is based on the *CASy* mechanism is scaled and several new agent types are defined to incorporate segmentation of the product space into multiple markets. For computational efficiency, we enable the distribution of the shop agents over a network of agent servers.

In [Bohte *et al.*, 2001], the shops operate in the context of one homogeneous market. In the prototype, we take a more complex view as multiple markets are introduced with each their own, possibly overlapping, products. Shops hence now must analyze consumers from the perspective of their behavior with respect to multiple markets. Each market is handled in the spirit of [Bohte *et al.*, 2001].

The individual shops in a full-blown mall implementation will require heavy computational resources for their classification of consumers. A strong point of market-based programming techniques is that the correct bidding strategy is learned and executed by each of the individual shops without need of a central server that runs the algorithms of many shops in parallel. This mechanism makes it possible for each shop to be run on its own dedicated hardware, for maximum computational scalability. We developed our prototype to allow each agent to be run on its own machine. This enables for the support of numerous shopping agents where scaling in the number of heavy-weight shop agents is realized by the addition of hardware.

2. Agent Architecture

In this section we outline the agents in our implementation in a first approximation without going into full detail. The different types of agents are introduced and their role in the whole is discussed, along with an example scenario. For the prototype, we focus mainly on the choice in the role and number of types of agents and their required (future) communication.

2.1 Previously Introduced Type of Agents

In [Bohte *et al.*, 2001], a single market is described where all consumer attention space is auctioned off to all participating shops. In [Bohte *et al.*, 2001], there are two main type of agents:

- A "mall manager" agent that runs the mall and hosts the visiting consumers.
- A "shop" agent that implements the bidding strategy for the attention space of a consumer.

2.2 Extension to Multiple Markets

Although communication and computational resources currently available suffice to sustain a significant number of shops, the concept can be generalized in a distributed manner by segmenting the single market into multiple markets, where the mall manager decides where to auction a particular consumer attention space. This would allow shops to only participate in a number of the available markets, in particular those that address their own business areas.

If the mall manager performs a rough first classification – separating the car-buyers from say the garden-equipment buyers – the assigned market will then perform the appropriate allocation. The mall manager could possibly also assign an item to multiple markets, to allow the segmented markets to have some overlap. Many choices for the role of a market in the shopping mall are possible. Of importance is that for the choices made, the *CASy* mechanism is still applicable.

For the role of managing the individual markets, we introduce an *Auctioneer* agent.

2.3 Refinement of the Shop Agent

To allow for increased granularity in the markets, the shop agent representing the real-life business as introduced in [Bohte *et al.*, 2001] is split into two parts.

One part of the shop agent architecture, called the set of *ShopNegotiator*s, runs the actual bidding for the consumer attention for the markets. Each real-life shop has one such bidder running in parallel for each market, allowing for maximum parallelism in the bidding on the markets. Note that assigning one instance of a dedicated shop negotiator to each market allows differentiation between strategies used for each market.

As a second part of the shop agent architecture, we introduce the high-level part of the shop called *ShopManager*. This agent, as one part of its role, handles the actual purchases by a consumer. Secondly, it will in extensions of the prototype manage the complexities of the interactions between the concurrent bidding for multiple markets by the *ShopNegotiator*s. In the prototype we handle each market as a separate, individual scenario. If however, for example, consumer profiling information between the markets could be exchanged, then this is a task that can be assigned to the "ShopManager" in the sense of collaborative filtering [Sorensen *et al.*, 1997] . As another example, budget considerations for learning over all the markets can become an issue that is monitored by this high-level agent.

2.4 The Agents in the Prototype

In summary, for our prototype implementation, we develop the following main type of agents:

- A *MallManager* agent that runs the mall and hosts the visiting consumers.
- An *Auctioneer* agent that facilitates the auction for a specific market in which a business bids for consumer attention space.
- A *ShopManager* which handles the purchases of a consumer.
- A *ShopNegotiator* agent, one per market for each business, that implements the bidding strategy for the attention space of a consumer.

Each of these agents is implemented to run on its own dedicated hardware for maximum performance.

All the agent types used in the simulation are depicted in Figure 1. The next section describes the communication between the agents and gives an example of a visit of a consumer to the shopping mall. We note that the total amount of communication is linear in the number of consumers who visit the mall.

3. Agent Communication

Figure 1 shows a representation of the communication channels and agents and actors in the prototype. The input and output channels are detailed with a brief description of their role. Furthermore, a consumer is described in a scenario in which an auction is actualized.

Note that Figure 1 is a representation of the agents for *one* market. The one dedicated auctioneer agent is represented for this one market along with the mall manager. The shop manager agents along with their shop negotiators for this market are also shown. For additional markets, one new auctioneer agent and one shop negotiator for each shop manager agent are added to make the figure complete.

Initially, the agents are created and distributed over their respective dedicated hardware. In the initialization phase of the prototype, each *ShopManager* registers through an *apply_subject_area* to the *MallManager* for the various markets (handled by the *Auctioneer* Agents). The shop requests for registration are then forwarded to the relevant *Auctioneer* Agents using the link *shop_info*. Each *ShopManager* also sets up its *ShopNegotiator*s to handle the actual bidding for the attention space of an individual consumer for each market. The shopping mall is at this stage ready to receive consumers.

Below, we sum up step by step a typical scenario. In the scenario one consumer visits the mall, is presented with a list of banners from the shops and possibly makes a purchase. The links are

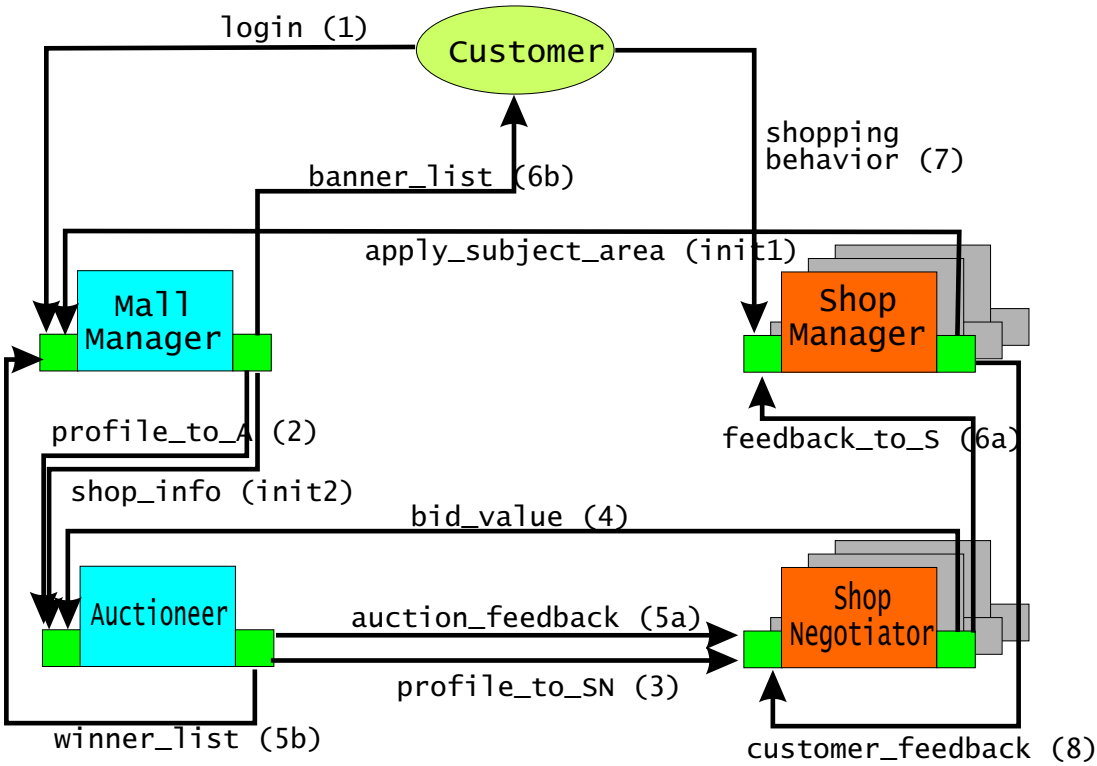

Figure 1: Overview of the Architecture

numbered in the order in which they are activated. Where two links can be activated independently and in parallel the suffix a or b has been added.

- 1. A *Consumer* enters the mall for shopping. The *login* message to the *MallManager* registers the consumer and a profile for the consumer is registered.
- 2. The relevant part of the profile is sent to the appropriate *Auctioneer*s through the *profile_to_A* channel to be able to process the consumer. This information is private and confidential within the context of the *MallManager* and its auctioneers that run the separate markets.
- 3. A partial profile, possibly augmented by aggregate information, is sent to the *ShopNegotiator*s so that each shop can determine their interest in the consumer. *ShopNegotiator*s receive only the necessary information on the consumer if the consumer is assigned to the specific market/*Auctioneer* the shops are operating with. This allows for privacy considerations for the consumer where the *MallManager* and the *Auctioneer* together operate as an intermediary.
- 4. The *ShopNegotiator*s each place their bids for consumer attention space. In the prototype implementation this is a second price Vickrey auction. Future implementations may include multi-round negotiation protocols.
- 5a. The *Auctioneer* calculates the results from the bids of the individual *ShopNegotiator*s and through *auction_feedback* the feedback is given to the *ShopNegotiator*s. This entails whether they have won the bid and of the costs attached to this result. This feedback is one part of the data available for the for the correct bidding for the attention space of a consumer.
- 5b. The feedback to the *ShopNegotiator* is also passed on to its *ShopManager* for high-level consideration, for example cross-market bidding budget considerations.
- 6a. The *Auctioneer* communicates to the *MallManager* the resulting banner list which should be presented to the consumer.
- 6b. The *Consumer* is finally presented with a list of banners from which to choose for the purchase of goods as indicated by his profile.
- 7. The *Consumer* decides whether to purchase a product based on the presented banners. The appropriate *ShopManager* is visited.
- 8. The *ShopManager* gives feedback to the *ShopNegotiator* on the actual actions of the consumer (ignored/browsed/bought). This is the second part (next to step 5) of the data available for the *ShopNegotiator* for the correct bidding of the attention space of a consumer.

In the next section we briefly discuss the programming language used for our prototype and the implementation of the communication channels between the agents.

4. Implementation Platform

For the implementation we used Java, version 1.3 JDSK [Java, 2001a]. This choice was motivated by several considerations, many of which are standard reasons for using Java.

4.1 Object-Oriented

First of all, Java as a modern object-oriented language lends itself well to implementation of agents. We use a relatively straightforward approach where each agent is modeled by a separate class. Each class encapsulates the behavior and data of the implemented agent. Each persistent object/agent is equipped with its own thread of control as supported by the Java *Thread* class for fully autonomous, parallel behavior. Furthermore, the inheritance mechanism allows for extensions of the agents in the prototype to future needs.

4.2 Platform Independence

Secondly, the Java language is platform independent. This allows our implementation to be run on any platform that supports the Java runtime environment, important for deployment of the prototype and future releases. Furthermore, the platform independence allows for easy distribution of the agents in a full-blown implementation where each agent runs on dedicated hardware.

4.3 Communication

Furthermore, Java supports several dedicated class libraries for connections between remote objects running on different machines, in our case agents. Java supports sockets, CORBA, RMI (remote method invocation) and other mechanisms for inter-network communication between objects. We chose RMI as it supports a straightforward, flexible mechanism that can also easily be integrated into code designed to be run on only one machine but has to be upgraded to be run over a network.

Basically, RMI consists of a name server where objects can register themselves and acquire a reference to another object, possibly residing on a remote machine. Objects do not interact directly but communicate through references received from the name server. Interfaces for the objects specify the (method) calls they accept from remote objects. The RMI mechanism allows us to develop the prototype of the shopping mall with parallel threads on one machine, test it thoroughly and without extensive changes in the code modify the agents to communicate through RMI references. The distribution of the agents to be run on dedicated hardware is straightforward.

In the prototype implementation, only the $MallManager$ agent platform is known to all the agents. All other agents register locally with the RMI name server on their machine and then with the $MallManager$. This approach allows for the flexible distribution of agents over the available machines in the network.

4.4 Security

As a further reason for chosing Java is its granular security model [Java, 2001b] to enable the applications to be run in remote environments with limited, configurable system access. Furthermore, additional security aspects with an agent can be handled. For example, for the RMI service, a custom security manager can be installed. This is an important issue for a full-blown implementation where privacy considerations of a consumer and secure shop-data are an issue, e.g. [Stallings, 1998].

In the next section we discuss the learning mechanism for the shops in their bid for the attention space of the consumer.

5. Learning Mechanism

In the spirit of [Bohte *et al.*, 2001], we built a learning mechanism that exploits the feedback loops available in the shop negotiation mechanism. We use these feedback loops to maximize the profits for the shop negotiator participating in the bidding.

We used a second-price Vickrey auction where the dominant strategy is the bidding of the actual expected profit (valuation) [Vickrey, 1961]. For this auction the agent cannot gain by using a different strategy. From the auction, we receive the feedback whether we won the bid or not.

For a winning bid, we have a consumer who may make a purchase at the shop for which the shop negotiator presented the banners. From this information the shop negotiator agents can derive the expected profit from a consumer making a purchase given the consumer profile and the profile of the shop negotiator. These two sources of information are combined in one strategy for the bid placed for the attention space of one consumer.

5.1 The Profiles

As in [Bohte *et al.*, 2001], the consumers and the shops are encoded using a profile upon which we base the stochastic purchase behavior of the consumers. The greater the distance between a consumer profile c and a shop profile s, the smaller the chance of purchase.

Both the consumer and the shop profile are encoded as a number in the interval $[0, 1]$, for n dimensions. The distance $||\vec{c} - \vec{s}||$ between two profiles is the euclidean distance given by

$$\sqrt{(\sum_{k=1}^{n}(\vec{c}_k - \vec{s}_k)^2)} \tag{5.1}$$

where $\vec{c}$ and $\vec{s}$ are vectors representing profiles in n dimensions and $\vec{x}_i$ is the ith component of the vector $\vec{x}$. In [Bohte *et al.*, 2001], the learning algorithms are explored for one and two dimensions. In this paper, we generalise our code to use one or more dimensions.

5.2 Collection of Training Data

In the prototype, we administrate the data of the customers to whom a banner is presented. Provisionally, we initially see a customer as a non-buyer until observed otherwise, i.e. a purchase is made. We correct our training data when, after a random delay, the consumer actually makes the purchase. This delay is built in between the presented banner and the time that a purchase is actually made. The shop negotiators have to deal with a delay and permutation in the order of the consumers to who they present a banner.

A buying customer is coded as a 1 for its point in profile space while a non-buying customer is coded as a 0. For a given shop negotiator, data is accumulated as successful bids are made for the attention space of a consumer. The data points close to the shop negotiator profile will predominantly be 1s while further away (in euclidean distance) the number of "0" data points will increase in relative frequency while the total of number of data points further away from the shop negotiator will drop under intense competition. As such, from the accumulated data of buying consumers, an area of profile space can be derived in which the shop is competitive; this is its niche in the market.

The next section introduces Radial Basis Function networks which we use to learn the consumer behaviour from the accumulated data.

5.3 Fitting the Chance of Purchase

Radial Basis Function (RBF) networks are a class of neural networks in which the activation of a hidden unit is determined by the *distance* between the input vector and a prototype vector [Bishop, 1995]. Each hidden unit represents a single RBF. The main advantage of such networks is that using several RBFs different RBF functions can be used to represent areas of the profile space that can be updated more or less locally. RBF networks with multiple centers are well-suited if more than one niche in the profile space by a shop negotiator must be learned simultaneously. Additionally, efficient algorithms exist to capture clustering using RBF networks [Asim Roy, 1995] in case customers are not distributed homegeneously over the profile space.

A RBF, ϕ, is one whose output is symmetric around an associated *center*, μ_j [Bishop, 1995]. That is, $\phi_j(\vec{x}) = \phi(||\vec{x} - \vec{\mu}_c||)$, where we chose ϕ to be the Gaussian function. A set of k weighted RBFs can serve as a basis for representing a wide class of functions that are expressible as linear combinations of the chosen RBFs [Bishop, 1995]:

$$y(\vec{x}_i) = \sum_{j=1}^{k} w_j \phi_j(\vec{x}_i), \tag{5.2}$$

A RBF network is an embodiment of Equation 5.2 as a feedforward network with three layers: the inputs, the hidden layer representing the RBFs and, in our implementation, a single output node.

The training of the basis functions ϕ_k consists of updating their centers $\vec{\mu}_j$, their radius σ_j and their weights w_j to minimize the error. If we choose an equal number of centers as the number of samples, we can even arrive at a perfect fit for the data. A regularisation term may have to be added to ensure smoothness of the derived function (y), although this is not the case in our application.

We take k ($k \geq 1$) basis functions ϕ_j with centers $\vec{\mu}_j$ and radius σ_j. The output of the RBF for $\vec{x}_i$ is given by Eq. 5.2. The error is the summation over all the n patterns of the data, i.e. the standard mean-square error $E_n = \frac{1}{2}\sum_k \{\vec{t}_n - y(\vec{x}_n)\}^2$ [Haykin, 1994], where $\vec{x}_n$ is the value of sample n, t_n is the desired function value for sample n and y is the function that we use to approximate the data given the $< \vec{x}_i, t_i >$ pairs.

In the following, we present the training rules based on method of steepest descent [Haykin, 1994]. The Δ for each of the basis functions for reducing the error in each of the patterns is given by the equations below. η_1 to η_3 are the learning rates for the different aspects. In our implementation they are set to the commonly used value of 0.001.

The Δ rules are then as follows:

$$\Delta w_j = -\eta_1 (y(\vec{x}_n) - t_n)\phi_j(\vec{x}_n),$$
$$\Delta \vec{\mu}_{j,l} = -\eta_2 \phi_j(\vec{x}_n) w_j \frac{(\vec{x}_{n,l} - \vec{\mu}_{j,l})}{\sigma_j^2}(y(\vec{x}_n) - t_n) \text{ and}$$
$$\Delta \sigma_j = -\eta_3 \phi_j(\vec{x}_n) w_j \frac{||\vec{x}_n - \vec{\mu}_j||}{\sigma_j^3}(y(\vec{x}_n) - t_n).$$

Parameters $\vec{\mu}_{j,l}$ and $\vec{x}_{n,l}$ are the lth component of the vectors for the centers and training samples. Repeated iteration over batch training for all the samples converges the RBF function-approximation to the sought for chance-of-purchase function y.

5.4 Correct Bids

As we use a second-price Vickrey auction, the dominant strategy is to bid the valuation of the shop negotiator for a given consumer. The expected profit for a shop negotiator is its *margin* times the estimated chance of purchase of a given customer. The *margin* is the average profit in case of a sale.

This gives as bid for the attention space of a customer:

$$bid = margin * ChancePurchase(consumerprofile, shopnegotiatorprofile)$$

where the chance of purchase depends on the distance between the profile of the consumer and that of the shop. The chance function has to be initially approximated and learned over time. Note that we here, as opposed to [Bohte *et al.*, 2001], for the sake of simplicity only assume competition for the placement one banner and a simple consumer model. The RBF networks due to their general nature are however suited for the more complex consumer models of [Bohte *et al.*, 2001].

Initially, we assume some imprecise, but globally accurate, market knowledge of the purchase chance of a consumer. This is not unrealistic as we would not expect a large number of shops to compete for customers if there is no expectation of possible profits. The chance function is improved through learning the actual customer purchasing behaviour. The learning algorithm of the shop negotiator can fine-tune the bids for the attention space as the precise purchase behavior is learned. Hence a shop can enter a new market based on prior expectations, and after some time learns to compete optimally.

In our simulations, we postulate a standard *margin* for all the participating shop negotiators. This simplifies the comparison of successful bidding behavior between the different shop negotiators within one market.

Like in [Bohte *et al.*, 2001], we define the chance of purchase of a consumer with respect to a shop negotiator as exponentially decreasing with the distance between the profile of the consumer and the shop. We use as function a Gaussian $e^{-\frac{(||\vec{c}-\vec{s}||)}{2\sigma^2}}$ where $\vec{c}$ is the profile of the consumer and $\vec{s}$ the profile of the shop and $||\vec{c} - \vec{s}||$ the (euclidean) distance between the profiles. σ is the radius of the Gaussian that determines the rate of drop with distance. However, any bell shaped function would be realistic; we simply chose the Gaussian as it is well known. Note that we assume that all consumers have an identical chance-of-purchase function. Furthermore, the buying behaviour of the customers is static, as is quite common in consumer modelling [Jeffrey O. Kephart, 2001].

In the prototype, we integrate the knowledge of the shape of the function of the chance of purchase by a consumer; the shop negotiator knows that this is a Gaussian, a bell-shaped curve dependent

on the distance in profile space. Furthermore, the shop negotiators are provided with some upper bound for the radius of this Gaussian. The above information may, for example, originate from initial market research and can lead to a choice for a shop to participate in a market as a potentially fruitful enterprise.

The actual optimal bid for a *ShopNegotiator* with profile $\vec{s}$ for a consumer $\vec{c}$ that has to be learned is:

$$bid = margin * e^{-\frac{(||\vec{c}-\vec{s}||)}{2\sigma^2}}$$

The shop negotiator knows its own profile, it knows the profile of the consumer for who it has to make a bid as given by the auctioneer and it knows its own margin. It however needs to learn the value σ of the consumer.

For the learning of σ, we currently need only one RBF function, a Gaussian. The center is set to the profile of the shop negotiator. The weight of this center is set to 1. Note that the weight update rules should be used if the chance of purchase for a consumer is not a 100% if the profile of the consumer and shop match exactly.

5.5 On-line Learning

The consumer behavior is learned on-line. Data is accumulated and once sufficient consumers are targeted by a winning bid and the accompanying banner, adaptation of the parameters of the RBF is started. At the beginning of learning, we work with all the data we collected.

Continueing with the learning using all collected data means that a shift in the behavior of consumers (in terms of buying chance) will become harder to detect as it can disappear as noise in the large amount of total data. As an initial, effective means of on-line learning we use a sliding window approach and only train with a fixed number p of data points for the last p consumers for who a successful bid for the attention space has been made. The size of the time window p is chosen empirically for our experiments as sufficiently large to be able to derive the σ of the Gaussian that determines the buying behavior of the consumers. Using this approach, a gradual shift in the σ of the consumer is detected at the latest when p or more new consumers have been bid for after a shift in the consumer behavior has started.

This on-line learning technique, although straightforward, is trivially correct. It also serves to demonstrate the flexibility of the prototype. More advanced, existing techniques can for example also be used to adjust the RBF network to online-learning, e.g. [Saad, 1998].

5.6 Experimental Results of the Learning of the Chance of Purchase

For the experiments we used a consumer with a σ for 0.3, although varying the radius in the interval $[0.1, 0.8]$ did not result in extreme changes in the results. We used 1 to 7 Gaussian base functions for the RBF, but this did not make a significant difference for the matching of the chance of purchase by the consumer. This will however become an issue if more complex models are used for the purchase behavior and hence we have set up the prototype to support multiple RBF functions.

We found that reasonable estimation of σ was possible after registering 25 buying consumers. A perfect fit was achieved with a minimum of 30 purchasing consumers.

In a run with 20 shops for a consumer profile in $[0, 1]^3$, the shop negotiators learned σ to within 0.05 after 30 purchasing customers per shop negotiator were successfully bid for.

We stress that these results are for *ShopNegotiators* operating in one, independent market. Learning in multiple markets occurs in parallel and at the same rate.

6. Conclusion

We presented an agent architecture and a prototype of a distributed shopping mall. Within the shopping mall, shops purchase the attention space of the consumer by participating in an auction. A mall manager acts as an intermediate between consumers and shops and facilitates the auction mechanism.

The actual auctioning is delegated to specialised auctioneer agents, which operate on specific market segments. This enables simultaneous bidding in multiple, independent markets. Similarly, the shops bid on the attention space using a shop negotiator agent, which is also specialized to handle customers from a particular category or market.

The shop negotiators in the implementation can learn their niche and adapt their bidding strategies to the actual customer purchasing behavior. This learning occurs online as different types of customers subsequently arrive at the mall.

The chosen architecture enables shops and the mall manager to be run on different machines. The agents in the application can be distributed over dedicated machines for maximum availability of computational resources. This is of importance as we envision that heavy-weight learning algorithms will be employed in future full-blown implementations of the shop agents. Through their implementation in Java, the agents can easily run remotely, using RMI, on any machine supporting a Java virtual machine. Communication between the agents is kept to the acceptable low order of linear in the total number of customers that visit the shopping mall.

Furthermore, we also simulated the customer behavior. In contrast to previous work [Bohte *et al.*, 2001], the purchases of the customers are not immediate but delayed in time. This provides new challenges for the learning mechanism within the shop negotiators.

Using the feedback from the auction (win/loss) and of the customer behavior, the shops adapt their bids. The purchasing behavior of the customers is estimated using radial basis function networks. Preliminary results show that the shops are capable of finding their niche and adapting to the consumer behavior for simple settings (20 shops, a single banner and a three-dimensional customer profile). In future work we aim at extending the results for additional settings of the simulation and at augmenting the current learning mechanism.

References

[Asim Roy, 1995] Raymond Miranda Asim Roy, Sandeep Govil. An algorithm to generate radial basis function (rbf)-like nets for classification problems. *Neural Networks*, 8(1):179–201, 1995.

[Bishop, 1995] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[Bohte *et al.*, 2001] S.M. Bohte, E.H. Gerding, and H. La Poutré. Competitive market-based allocation of consumer attention space. In M. Wellman, editor, *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC-01)*, pages 202–206. The ACM Press, 2001.

[Clearwater, 1995] S.H. Clearwater, editor. *Market based Control of Distributed Systems.* World Scientific Press, Singapore., 1995.

[Haykin, 1994] S. Haykin. *Neural Networks-A Comprehensive Foundation, Macmillan And Ieee Press*, volume York. Macmillan College Publishing Compagny Inc., 1994.

[Java, 2001a] Java, 2001. http://www.java.sun.com.

[Java, 2001b] Java, 2001. Java security issues, http://java.sun.com/docs/books/tutorial/security1.2/index.html.

[Jeffrey O. Kephart, 2001] Rajarshi Das Jeffrey O. Kephart, Christopher H. Brooks. Pricing information bundles in a dynamic environment, 2001. Third ACM Conference on Electronic Commerce Tampa, Florida, USA.

[Saad, 1998] D. Saad, editor. *On-Line Learning in Neural Networks.* Cambridge University Press, 1998.

[Sorensen *et al.*, 1997] H. Sorensen, A. O'Riordan, and C. O'Riordan. Profiling with the INFOrmer text filtering agent. *J.UCS: Journal of Universal Computer Science*, 3(8):988–999, 1997.

[Stallings, 1998] William Stallings. *Crytography and Network Security, Principles and Practice, second edition.* Prentice-Hall, Inc., 1998.

[Vickrey, 1961] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[Weiss, 1999] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, 1999.