This is the post print version of the paper: L. Palopoli, D. Rosaci, G.M.L. Sarnè. Introducing Specialization in e-Commerce Recommender Systems. Concurrent Engineering: Research and Applications (**CERA**). Vol. 21, Num.3, pp. 187-196. 2013. Sage.
The editorial version is available at https://doi.org/10.1177/1063293X13493915

Introducing Specialization in e-Commerce Recommender Systems

Luigi Palopoli

University of Calabria, DIMSI Dep., 87036 Rende, Italy

Domenico Rosaci

University Mediterranea of Reggio Calabria, DIIES Dep., 89122 Reggio Calabria, Italy

Giuseppe M.L. Sarnè

University Mediterranea of Reggio Calabria, DICEAM Dep., 89122 Reggio Calabria, Italy

Corresponding author - Tel: +39 0965 875438; Fax: +39 0965 875220; e-mail: sarne@unirc.it

Abstract

Currently, customers' trading activities are supported by recommender tools able to generate personalized suggestions. Many of these recommenders are centralized, lacking in efficiency and scalability, while other ones are distributed and, conversely, imply a computational overhead on the client side being often excessive or even unacceptable for many devices. In this paper, we propose a distributed recommender, based on a multi-tiered agent system, where the agents of each tier are specialized in a different e-Commerce activity. The proposed system is able to generate effective suggestions without a too onerous computational burden. In particular, we show that our system introduces significant advantages in terms of openness, privacy and security.

Key-words: Multi-Agent Systems, Distributed Architectures, e-Commerce

1. Introduction.

Nowadays, a large number of Recommender Systems (RSs) are used to promote e-Commerce (EC) activities [47] but often they fall short when transactions occur between customers and merchants (B2C), mainly for inadequate exploration of the market space, ineffective communications between the actors

and lack of security and privacy in the transactions. To face such issues, new generations of B2C systems, characterized by high levels of automation, exploit software agents that, acting on the customers' behalf, allow a B2C transaction to be carried out largely without human intervention [34, 41].

Usually, RSs adopt a common representation of concepts and their relationships involved in the user domain (ontologies) to facilitate mutual interactions [13,17,35]. In particular, a RS provides his user with suggestions for purchases [52] based on a representation of his interests and preferences (profile) across the different phases of a B2C transaction, that can be built by software agents [18,54]. Different behavioral models describe such phases, such as the Consumer Buying Behavior (CBB) model [21] based on six stages, namely: *i*) Need Identification; *ii*) Product Brokering; *iii*) Merchant Brokering; *iv*) Negotiation; *v*) Purchase and Delivery; *vi*) Service and Evaluation. Moreover, centralized RS architectures generate suggestions only on the server side but their performances are limited in terms of efficiency, scalability and customers' privacy. The alternative approach implies distribution [44] but its complexity could generate unbearable computational overheads on the client side as, for instance, with mobile devices. Besides, existing RSs assume homogeneous system components, implying that it is difficult for users to add personal knowledge to the system.

To face such issues, in this paper, we present a RS for e-Commerce, called **Dl**stributed **REC**ommender **T**iers (**DIRECT**), that allows to: *i*) increase the distribution level of activities; *ii*) generate effective suggestions not implying onerous computational tasks for the client; *iii*) introduce significant advantages in openness and privacy. The basic idea of DIRECT is that each customer is assisted by three software agents, each of which, autonomously from the other agents, is specialized in a different CBB stage. Each agent runs on a different thread on the customer's client and this improves the efficiency of the overall process, making it specialized also the agent interactions. Each customer's agent can interact with the DIRECT sellers' sites over the Web, where each seller site is assisted by a seller agent provided with both a product catalogue and customers' profiles encoding the preferences of each past customer of the site. This interaction allows the customer agent to generate content-based (CB) recommendations for the customer and also visitors to support their site visit. The agents also interact with the seller agents and, in turn, gen-

erate collaborative filtering (CF) recommendations. This way, if a customer c_1 needs to interact with a customer c_2 for need identification purposes, his NI-agent simply interacts with the c_2's NI-agent. The other agents associated with c_1 and c_2 are free to perform other activities improving the system performances with respect to those systems where a unique agent can execute only one activity at time.

2. The DIRECT Knowledge Representation.

To represent a common knowledge, a simple and effective solution [30,39,43], also adopted in DIRECT, consists of exploiting a common dictionary storing the names of all the product categories of interest and their relationships. Furthermore, each customer's agent, associated with a CBB stage, encodes in an agent profile all those information needed to manage its CBB stage [38]. In the same way, each seller's agent encodes in an agent profile all those information needed to manage a catalogue of offered products organized in categories. To promote the agent cooperation, a "yellow page" data structure is available in DIRECT, where that information that agents desire to make public are stored (see below).

The *Dictionary* D consists of the sets D_C of *product categories* and D_R of *category relationships* between categories, denoted by $\langle cat_1, cat_2, r\rangle$, where cat_i is a category stored in D and r is a relationship type that can be: *i*) *isa*, denoted by *ISA*; if all the products belong both to cat_1 also belong to cat_2; *ii*) *synonymy*, denoted by *SYN*, if all the products belonging to cat_1 also belong to cat_2 and vice versa; *iii*) *overlap*, denoted by *OVE*, if there are some product of cat_1 that also belong to cat_2, and vice versa. Categories linked by a synonymy relationship are also linked by an overlap relationship; *iv*) *commercial*, denoted by *COM*, if customers usually purchase both products belonging to cat_1 and cat_2. A dictionary D (called COMMON), publicly available, represented by a direct labeled graph $G(D) = \langle D_C, D_R\rangle$, where for each category $cat \in D_C$ exists a node called $name_{cat}$ and for each $arc \in D_R$ oriented from cat_i to cat_j and labeled by r exists a link $\langle cat_i, cat_j, r\rangle$. In detail, nodes are associated with product categories, which those offered by the sellers belong to, and arcs represent existing relationships between categories.

In a DIRECT community (C), to handle the Need Identification, Product Brokering and Merchant Brokering CBB stages, each customer c is assisted by a specialized agent called NIA_c, PBA_c and MBA_c, respectively. Each one of them assists its customer in the activity related to the CBB stage associated with it and stores in a personal profile, called NI (resp. PB, MB)-profile, all the *c*'s information required to handle its CBB stage, represented by a category dictionary $PROD(NIA_c)$ (resp. $PROD(PBA_c)$, $PROD(MBA_c)$) such that: *i*) each node represents for *c* a category of interest (resp. a product category relative to a suitable product or merchant); *ii*) each arc represents a link between two categories and each category (resp. a product of interest or a merchant) is associated with a quantitative evaluation of the *c*'s interest. Moreover, when a customer performs a search for a product (resp., a merchant), often, he does not activate a search for each category (resp. product) of interest. As a consequence, the categories in $PROD(PBA_c)$ (resp. $PROD(MBA_c)$) usually are a subset of those in $PROD(NIA_c)$ (resp. $PROD(PBA_c)$).

For such a reason, each arc linking two categories in the PB (resp. MB)-profile, is a copy (also including the label) of the equivalent arc belonging to the NI (resp. PB)-profile. Finally, each category stored in an agent profile must belong to either the common dictionary or to a "personal" customer's category. In order to make personal categories understandable to the other agents, each personal category has to be in a general relationship with at least another category belonging to the common dictionary.

Each seller $s \in S$, associated with a DIRECT platform, is supported by a seller agent managing a site profile ($SiteP_s$) where a catalogue is stored of the offered products and some information about the preferences shown by its past customers in visiting the site. Also, $SiteP_s$ is represented by a category dictionary where: *i*) nodes represent product categories within which s offers products; *ii*) arcs represent relationships between categories. To manage this profile, a mapping $SiteP_S(cat)$ is used, that accepts in input a category cat and returns in output the tuple $SiteP_S(cat) = \langle prod, customers \rangle$, where $prod$ is a list of products, belonging to cat, offered by s. The elements of this list can be accessed by means of the mapping $SiteP_S(cat).prod(p) = \langle method, currency, price \rangle$ that accepts as input a product p and returns in output a list $method$ of available payment methods, a list $currency$ of accepted currencies and the $price$

of p, that depends if the price is fixed or it denotes a reserved price, like in an auction. Finally, the element customers consists of a list of *customers* interested in products belonging to the category cat that, for each customer c, stores a list of those products offered in the site which c is interested in.

To facilitate mutual users' collaboration, it is important to know users' orientations about their interests and preferences [4,19,40]. In DIRECT, customers can cooperate by using a Yellow Pages (YP) data structure available on the platform. Using the YP, each customer can publish all those information about his interests that he desires to make public. The YP data structure is realized as a set of category dictionaries YP_c, one for each customer c. More in detail, each dictionary YP_c is a sub-graph of the c's NI-profile and stores those categories belonging to $PROD(NIA_c)$ declared by c as public (i.e., $NIA_c(cat).i = public$.

3. Agent Behavior in DIRECT

A DIRECT agent runs on the client used by the customer c to visit the DIRECT EC sites and assists its customer by: *i*) supporting his Web site navigation like a normal browser; *ii*) generating personalized suggestions. To this purpose, the client interface is provided with two functionalities, namely, Browser and Recommender described below. If c is a newcomer on the DIRECT platform, he should build an initial NI-Profile $PROD(NIA_c)$ by adding those categories (cat) he is interested in, also including their relationships, and belonging to the dictionary COMMON. Moreover, in a simple way c could add some personal categories (cat^*) that are absent in COMMON to his profile $PROD(NIA_c)$. This is done by specifying their name and at least a path in $PROD(NIA_c)$ that joins cat^* with a category cat that is present in COMMON. Besides, for each selected category, c should specify his interest degree $NIA_c(cat).i$ and the visibility mode $NIA_c(cat).mode$ (which is related to using Yellow Pages data structure).

3.1. The Browser and the Recommender

On each site associated with the DIRECT platform, a customer c can exploit his browser to: *i*) navigate through the categories; *ii*) use the *Search* tool to perform a keyword-based search among the products,

both sold at fixed price or with an auction, generating personalized suggestions for him. Moreover, for each product p, belonging to a category cat, the customer c can perform the following actions to: *i*) A1 - select the product for examining the offer; *ii*) A2 - watch the product; *iii*) A3 - purchase the product. After that an action among A1, A2 or A3 is performed by c, it implies an automatic call to the agents NIA_c, PBA_c, and MBA_c takes place, that autonomously will update their profiles and, in particular the NIA_c agent is called, feeding it the category cat. If cat is absent in its profile, it is added therein and its interest value $NIA_c(cat)$ is set to a fixed value $iniInt$. Then NIA_c requires to the agents PBA_c and MBA_c to add cat and its $iniInt$ to their profiles. Otherwise, if $cat \in$ NI-profile, its interest value is updated to $min(1,\ NIA_c(cat) + \varepsilon_a)$, where $\varepsilon_a \in [0,1]$ (with a = A1,A2,A3) it is autonomously set by c to weight the performed action. Then $NIA_c(cat)$ is passed to PBA_c and MBA_c for updating their profiles.

Then the client calls the PBA_c agent to pass the product p. If $p \notin$ PB-profile, then it is added to the list $PBA_c(cat).prod$ with $iniInt$ as the interest value and the insertion in the list $MBA_c(cat).prod \in \mathrm{MBA_c}$ agent is required. Else, if $p \in$ PB-profile, its interest value $min\ (1,\ PBA_c(cat).prod(p).i + \varepsilon_a)$ and passed to MBA_c to be copied in the list $MBA_c(cat).p.i$

Finally, the MBA_c agent is called by the client passing the seller s to it. If $s \notin$ MB-profile, then it is added to the list $MBA_c(cat).sellers$ with $numT = 1$ and a score value $iniInt$. Otherwise, $numT$, increased by 1, and the score $MBA_c(cat).sellers(s).ev$ are updated to $min(1, MBA_c(cat).sellers(s).ev + \varepsilon_a)$.

After a τ_{NIA} (resp., τ_{PBA} o τ_{MBA}) time period is passed from its last update, the value $NIA_c(cat).i$ (resp., $PBA_c(cat).prod(p).i$, $MBA_c(cat).sellers(s).ev$) associated with the NI (resp. PB, MB)-profile is periodically decreased by ϱ_{NIA} (resp., ϱ_{PBA} o ϱ_{MBA}), a parameter c sets in $[0,1[$. Also, the seller agent of s updates its list $SiteP_s(cat).customers \in SiteP_s$ after each customer's action involved a product $p \in cat$ is performed. In particular, if $c \notin SiteP_s(cat).customers$, a new element $SiteP_s(cat).customers(c)$ is added to the SitePs profile and the product p is added to the list $SiteP_s(cat).customers(c).prod$ and the number of transactions $SiteP_s(cat).customers(c).numT$ is increased. On the contrary, the seller agent updates $SiteP_s(cat).customers(c).numT$ and adds p in $SiteP_s(cat).customers(c).prod$.

For receiving personalized suggestions, the customer c exploits a specific functionality of his browser. In this way, some suggestions are generated for him and visualized in a page having a section for each supported stage. Suggestions are generated by each agent in a "cascade" mode: *i*) initially the customer chooses a category from those suggested in the section "Recommended Categories"; *ii*) then the customer can choose a product from the set of products suggested for him in the section "Recommended Products"; *iii*) finally, a set of merchants selling that product is suggested in the section "Recommended Merchants". In details, NIA_c suggests to c a set of categories visualized in the client section "Recommended Categories" organized in three distinct list-boxes, namely: *i*) **Visited Categories**, it contains categories selected with a CB approach from the profile $PROD(NIA_c)$ built by monitoring c's activity (see Section 3); *ii*) **Unvisited Categories**, it lists those categories unknown to the customer c, but considered as potentially interesting to him by his NIA_c-agent. This agent uses a *relationship-based* mechanism to exploit the interaction between the c's NIA_c-agent and the agents of each site that he visited in the past. In particular, the agent of a seller s, for each category cat visited by c (i.e., $c \in SiteP_s(cat).customers$)), determines all the categories $cat^* \in SiteP_s$ such that cat^* is unvisited (for c) and there exists a path in $c \in SiteP_s$ between cat and cat^*; these categories are sent to c's NIA_c-agent to be added to this list; *iii*) **Suggested by Similar Customers**, where the categories are determined adopting a collaborative filtering technique, on the whole EC customer's navigation history, by the NIA_c-agent interacting with the NIA-agents of other customers similar to c for interests. To this aim, the public repository YP is exploited (see Section 2) storing, for each customer x his public interest profile YP_x. The NIA_c-agent computes the similarity degree $s(x,c)$ between its profile $PROD(NIA_c)$ and each YP_x by using the Jaccard measure are the set of nodes of $PROD(NIA_c)$ and YP_x for n customers (n is a parameter autonomously set by c), that is $s(x,c) = \left|NODES\left(PROD(NIA_c)\right) \cap NODES(YP_x)\right| / \left|NODES\left(PROD(NIA_c)\right) \cup NODES(YP_x)\right|$, where $NODES(G)$ returns the set of nodes of its input graph. Then, the NIA_c-agent determines those categories stored in YP_x of each similar customer x that do not belong to $PROD(NIA_c)$ for adding them to this list.

In section "product recommendations" (resp. "merchant recommendations") PBA_c (resp. MBA_c) suggests to its customer c a set of products (resp. sellers) belonging to a category cat selected by c from the recommended categories (resp. products) on his client. These products are visualized in the following listboxes: *i*) **Visited Products** (resp. **Visited Merchants**), that contains products (resp. merchants) of the category $cat \in PROD(PBA_c)$ (resp. $PROD(MBA_c)$) ordered by score; *ii*) **Unvisited Products** (resp. **Unvisited Merchants**), that is built by exploiting collaboration between the PBA_c agent of the customer c and the seller agent associated with each site s that c visited in the past. In particular, the Unvisited Products list is composed of products $p \in cat$ not visited by c in the past, while the Unvisited Merchant list contains those sellers having cat in their profiles and that c did not visit in the past; *iii*) **Suggested by Similar Customers**, these suggestions are generated based on a collaboration occurring between the PBA_c (resp. MBA_c) agent and the corresponding agents of other customers similar to c for interests (their list is provided by the NIA_c-agent). Each of the PBA_c (resp. MBA_c) agent (say x) of these similar customers sends its set of products $PBA_x(cat).prod$ (resp. merchants $MBA_x(cat).sellers$) to the PBA_c (resp. MBA_c) agent of the customer c in order to add it to this list. The PBA_c (resp. MBA_c) agent shows to its customer the products (resp. merchants) belonging to the list-box "Visited Products" (resp. "Visited Merchants"), ordered by value, and the products (resp. merchants) belonging to the list-boxes "Unvisited Products" (resp. "Unvisited Merchants") and "Suggested by Similar Customers", ordered alphabetically. Each seller agent SA_s associated with a seller s exploits its profile $SiteP_s$ to personalize the site presentation for the customer c that is visiting it. When c returns to visit the site, if $SiteP_s(cat).customers(c)$ already exists in the SA_s profile, using the information stored in such element, SA_s personalizes its home page for c by visualizing in a "Shop Window" all the products $p \in SiteP_s(cat).customers(c).prod$, ordered by interest value. SA_s uses this list as a sort of local profile related to c and, at the same time, it increases the value $SiteP_s(cat).customers(c).numV$ to consider his current visit. Otherwise, if it is the first time that c visits the site, the default home page is visualized.

4. Related Work.

Recommender systems (RS) have been considered by a large number of models and architectures proposed in the past and the interested reader might refer to [6,26,29,32,47,52] for a more complete overview of the current state-of-the-art on these tools. In this context, many systems support e-Commerce activities by exploiting different approaches, often agent based, such as in [5,8,14,15,20,22,25,27,31,41,42].

In this Section, we describe some approaches for generating recommendations in an e-Commerce context that are relevant to the work presented in this paper. Some of them, differently from DIRECT, are centralized, but share some similarities with it. Instead, other approaches, likewise DIRECT, are distributed. Below we overview these two kinds of RSs, highlighting both similarities and differences with DIRECT.

Centralized RSs are easy-to-implement and widely used within e-Commerce Web sites, such as that of Amazon [2], to support visitors in their purchases. The Amazon site makes it available to its customers some smart tools that tap into the customer's past purchases and purchases of other shoppers. For example, the site section "What Do Customers Buy After Viewing This Item?" shows collaborative filtering recommendations, providing statistical reasons as to why the customer should buy some items, while the section "News for You" shows content-based recommendations based of new items. To compute all these recommendations, Amazon exploits information derived by customers' behaviors performed on Amazon. Whether a customer likes to buy something because this is related to something that he purchased before, or because it is popular among other customers, the system drives him to add such an item to his basket. Similar tools are also available in many other Web stores such as CDNOW [9] and Dandang [12].

Another case is that of the RSs embedded in auction sites [10], where a well-known and studied example is that of eBay [16]. The RS generates suggestions in eBay by using its feedback profile features. Customers and sellers are allowed to provide feedbacks such as assessment of their satisfaction. Most of the time, buyers and sellers can then use this information as recommendations. eBay also provides a tool, called Gift Finder, which helps customers to find presents by matching the profile of the gift recipient.

Among the centralized RSs discussed in the literature WebSell [11] is an XML agent platform where customers are provided with a set of tools that would extend the range of products and services for trading. In

this platform suggestions are generated exploiting both content-based and collaborative filtering aproaches to bring customers together with products potentially of their interest. The suggestions generated by EC-XAMAS [15] and MASHA [37] take into account the device. EC-XAMAS, working on the client-side, preserves customers' privacy in finding products and/or services of interest, according to their past interests and behaviors. The second one considers the computational limits of devices and, differently from EC-XAMAS, the recommendation are generated on the server side.

All the presented approaches generate, similarly to DIRECT, both content-based and collaborative filtering recommendations. Differently from DIRECT, they are fully centralized recording in a main database all the private information necessary to generate recommendations and are not open, adopting a predefined dictionary of terms for defining categories and product categories of interest, while DIRECT allows the users to define new terms in their personal ontologies.

A different way to think of recommenders is embodied by Distributed RSs (DRSs) that have growing in popularity among researchers, as the large number of models and architectures proposed in these later years' witnesses. Although many DRSs have not been specifically designed for the e-Commerce domain and they often do not consider its specific peculiarities in generating their suggestions [47,55], some of them might be applied in this scenario without significant changes. However, in this section, only those DRSs explicitly designed for e-Commerce scenario will be examined.

The DSRs share information and computation tasks among several entities and, in this way, they are able to respond to different weakness of centralized approaches, mainly due to: *i*) lack scalability [23]; *ii*) the failure risks caused by the presence of a single server managing a central database [50]; *iii*) a potential loss in privacy and security since a single server should manage and defend a significant amount of personal information [1,7,56]. On the contrary, DRSs turn out to be more critical than centralized recommender systems in both design and performances optimization.

DSRs can exploit different recommender techniques even if most of them are based on a collaborative filtering approach. To easily solve some technical issues, DSRs often exploit the opportunities made available by peer-to-peer (P2P) and multi-agents technologies. Specifically, P2P are popular distributed

networks (e.g., CAN [33], Chord [49] and Pastry [46]) and mainly provide DRSs with efficient, scalable and robust routing algorithms to reduce the burden of tasks involved in locating a specific resource on the network. Complementarily, multi-agent systems provide DRSs with communication and negotiation facilities. Moreover, each agent can encapsulate specific functions, even distributed among several hosts, and reciprocally cooperate with other agents to achieve its goals. However, in DRSs, each agent is responsible only for its portion of the computational tasks involved in generating suggestions.

A distributed Competitive Attention-space System (CASy) to recommend shops and bids in competitive markets is presented in [3]. CASy is based on adaptive learning agents associated with each shop. Each agent processes a large number of shop transactions and exploits information about interests of every customer entering the shop. These are derived from profiles, keywords and product queries or provided by other shop agents. As a consequence, the commercial strategy is efficient and adaptive in tracking the consumer for proposing suitable personalized suggestions.

A multi-agent system implementing a knowledge-based DSR developed for the tourism domain is discussed in [28]. These agents are cooperative and recommend travel packages by reciprocally exchanging information derived by their local knowledge bases. Whenever a recommendation request is received, it is decomposed into sub-tasks handled by different agents, each one exploiting its own specialist knowledge base. A peculiarity of this DRS is that agents can autonomously select the most suitable tasks to deal with, based on their past experiences in a particular kind of service (hotel, flights, interchanges, conferences, etc.). This way, each agent can increase its specialization and, consequently, the confidence that, in each well defined field, the overall quality of the generated suggestions will increase. Adaptive distributed CF recommenders implemented by means of multi-agent architectures over a P2P networks for a mobile commerce scenario is proposed in [51] where products and services for marketplace users using mobile devices are suggested. This RS translates recommendation tasks into searching tasks over a P2P topology like Gnutella. Peers are represented as software personal assistant agents of mobile customers.

In [53] a DRS is proposed consisting of multiple RSs from different organizations. The authors introduce a peer selection algorithm that allows a RS peer to select a set of other peers to cooperate with. The paper

shows how it is possible to provide a solution to the problem of resource lacking (cold start problem) and also enables recommender systems to provide effective recommendations. Another distributed collaborative filtering recommender working over mobile ad-hoc networks, called MobHinter, is illustrated in [50]. Devices exploit ad-hoc connections to exchange information without accessing any remote online services, when for some different reasons (e.g. cost, failure of wireless network, etc.), other communication channels are unavailable. MobHinter allows the identification of affinity in the neighbors (usually a narrow portion of the users' community) for obtaining personalized suggestions by the way of direct meetings among users. Users' affinities are modeled by a similarity graph that links users to each other with a configurable affinity threshold. Finally, the collected information is used to locally refine predictions, in an incremental manner and without interacting with a remote server or to access the Internet.

All the cited systems take advantage of the distributed architectures in terms of scalability, risks failure, privacy and security. Differently from DIRECT, none of them expressly considers all the phases involved in an EC process. Most of them deal with the Need Identification, Product or Merchant Brockering phases using some form of profile to describe the user's interests and preferences. Moreover, all these RSs, included DIRECT, adopt (or can easily adopt) agent-based and/or P2P systems to find similar neighbors, interesting resources and advantageous predefined services. Agent specialization is introduced in [28] but it is relative to the item typology, while DIRECT is based on a different B2C phase (as described in the CBB model). A similar concept is also present in [53] but it involves a set of recommenders, each one linked to a different organization and having a particular point of view in generating its suggestions.

5. Efficiency and Effectiveness of DIRECT.

In this Section, we discuss efficiency and some experimental results about the advantages of our proposed approach. In terms of efficiency, for a community of n customers and m sellers, a unique centralized agent managing the Need Identification, Product and Merchant Brokering phases has a computational cost CC^C that is $CC^C = \sum_{i=1}^{3} n_i \cdot k_i$, where k_i is the number of simultaneous sessions activated to manage each of the three phases and n_i is the number of operations needed for a user to manage each phase.

In DIRECT, for a given CBB stage, each distinct agent deals with a distinct task on distinct thread. Let α be the multi-threading degree for a specific CBB stage and let $\beta = k_1 + k_2$ be the computational overhead due to the communications between local agents. The implied computational cost is, in this case $CC^D = \beta + \sum_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i$ and the computational advantage (ρ), caused by to the task distribution in DIRECT, is equal to $\varrho = \left(\beta + \sum_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i\right) / \left(\sum_{i=1}^{3} n_i \cdot k_i\right)$ where if, for simplicity, $\alpha = \alpha_1 = \alpha_2 = \alpha_3$, $k = k_1 = k_2 = k_3$ and $N = n_1 = n_2 = n_3$, the above formula becomes $\varrho = \alpha + 2/N$.

Therefore, the advantage of using DIRECT is perceivable with a limited multi-threading in presence of a reasonably high number of operations (i.e. an intense EC activity). Differently, with high multi-threading levels, the advantage shows up even for small values of N.

To evaluate the proposed multi-tiered recommender system in terms of effectiveness, the time exploited to perform B2C processes in serial and multi-threading mode has been compared by means of software purposely designed to test this recommender. To this aim, we considered a period of 2 hours where a set of 500 customers finalize all their B2C processes dealing with a merchant population (M) of 10 units. Moreover, it is assumed that each merchant has to satisfy also the requests of customers outside the DIRECT platform, which could absorb significant merchants' servers resources. This possibility is taken into account in the simulation by means of an overhead (O) of $1 \div 100$ requests for second, randomly shared among the merchants. Note that the activity period, i.e. the number of customers and merchants and the overhead, can be set in the test software within a significant range. Finally, a lot of different of communication, computational and behavioral parameters have been tuned to model realistic B2C processes. Obviously, in order to compare, on the average, the time (in seconds) needed to perform a purchase process in a multi-threading (T_m) and in a serial (T_s) modality the same values for the parameters have been used. More in detail, $\mathrm{T_m}$ (i.e. $\mathrm{T_s}$) has been computed as $T_m = \sum_{i=1}^{NP} T_m / NP$, where NP is the number of purchases, randomly fixed, performed in the considered test session.

The experimental results shown in Figure 1 confirm that the DIRECT approach consumes, on the average, about the 25% of time less on the serial approach in performing a purchase in the absence of overhead.

When the overhead grows, T_s grows as well, while T_m is almost constant. These results, as average gain (G) in percentage of T_m with respect to T_s ($G = T_s/T_m$), for different values of the overhead from 0 to 100 with a step of 10 request for second and considering 500 Customers and 10 Merchants are [24.61, 44.63, 56.43, 63.98, 69.19, 73.05, 76.39, 78.39, 80.75, 82.28, 83.61] for each step.

This behavior is caused by the fact that changes in the number of merchants, overheads and so on, have a minimal impact on T_m, while it is very high on T_s. This happens because, on the average, each merchant's server results busy to satisfy the customers' requests and Ts grows with the level of "saturation" of the merchants' servers, whereby, in this case, the quality of the service gets worse.

[Figure 1 Inserted Here]

Figure 1. The average serial (T_s) and multi-threading (T_m) times (in sec.) needed to carry out a B2C process depending on the Overhead by considering 500 Customers and 10 Merchants.

6. Conclusions

In this paper, we have presented the DIRECT distributed architecture that introduces original characteristics with respect to traditional EC recommender systems. DIRECT allows the different CBB stages of an EC process to be assigned to a different agent creating a tier of specialized agents. This architecture reduces the computational burden on the device on which the local agents run and the presence of specialized agents improves users' knowledge representations. We have performed an experimental campaign to evaluate the performances of our system in terms of effectiveness by implementing a software specifically designed to this aim.

The main issue of our ongoing research focuses on considering different behavioral model of a B2C process and emerging behaviors, analyzing other relevant properties in a distributed environment, such as reliability, dependability and security also by integrating in DIRECT some techniques we developed in the field of trust and reputation [36,45] in order to highlight other advantages of the proposed approach.

References

[1] Ackerman M.S., Cranor L.F. and Reagle J., (1999). Privacy in e-Commerce: Examining User Scenarios and Privacy Preferences. In Proc. of ACM Conf. on Electronic Comm, 1–8, ACM, NY, USA.

[2] Amazon URL.(2013) http://www.amazon.com.

[3] Bohte S.M., Gerding E. and La Poutrè J.A., (2004). Market-based Recommendation: Agents that Compete for Consumer Attention. ACM Trans. Internet Techn., 4(4):420–448.

[4] Buccafurri F., Palopoli L., Rosaci D. and G.M.L. Sarnè., (2004). Modeling Cooperation in Multi-Agent Communities. Cognitive Systems Research, 5(3):171–190.

[5] Buccafurri F., Rosaci D., Sarnè G.M.L. and Ursino D., (2002). An Agent-Based Hierarchical Clustering Approach for E-Commerce Environments. In E-Commerce and Web Technologies, Proc. of the 3rd, vol. 2455 of Lecture Notes in Computer Science, 109–118, Springer-Verlag, Berlin Heidelberg.

[6] Burke R.D. Hybrid Recommender Systems: Survey and Experiments., (2002). User Modeling and User-Adapted Interaction, 12(4):331–370.

[7] Canny J.F., (2002). Collaborative Filtering with Privacy. In Proc. of IEEE Symp. on Research in Security and Privacy 2002, 45–57, IEEE Press, Washington DC.

[8] Castro-Schez J.J., Miguel R., Vallejo D. and Lopez-Lopez L.M., (2011). A Highly Adaptive Recommender System Based on Fuzzy Logic for B2C e-Commerce Portals. Expert Systems with Applications, 38(3):2441–2454.

[9] CDNOW URL., (2010). http://www.cdnow.com.

[10] Culver B., (2004). Recommender System for Auction Sites. J. Comput. Small Coll., 19(4):355–355.

[11] Cunningham P., Bergmann R., Schmitt S., Traphner R., Breen S. and Smyth B., (2000) WEBSELL: Intelligent Sales Assistants for the World Wide Web. In Proc. of the Work. Programmed at the 4th Int. Conf. on Case-Based Reasoning, 104–109, 2000.

[12] Dandang URL., (2011). http://www.dandang.com.

[13] De Meo P., Quattrone G., Rosaci D. and Ursino D., (2012) Bilateral Semantic Negotiation: A Decentralized Approach to Ontology Enrichment in Open Multi-Agent Systems. International Journal of Data Mining, Modelling and Management, 4(1):1–38.

[14] De Meo P., Rosaci D., Sarnè G.M.L., Terracina G. and Ursino D., (2003). An XML-Based Adaptive Multi- agent System for Handling e-Commerce Activities. In Int. Conf. ICWS-Europe 2003, Proc. of the 1st, vol. 2853 of Lecture Notes in Computer Science, 152–166, Springer-Verlag, Berlin Heidelberg.

[15] De Meo P., Rosaci D., Sarnè G.M.L., Terracina G. and Ursino D., (2007). EC-XAMAS: Supporting e-Commerce Activities by an XML-Based Adaptive Multi-Agent System. Applied Artificial Intelligence, 21(6):529–562, 2007.

[16] eBay URL., (2013). http://www.ebay.com.

[17] Garruzzo S., Quattrone G., Rosaci D. and Ursino D., (2011). Improving Agent Interoperability Via the Automatic Enrichment of Multi-Category Ontologies. Web Intelligence and Agent Systems, 9(4):291–318.

[18] Garruzzo S., Rosaci D. and Sarnè G.M.L., (2006). MAST: an Agent Framework to Support B2C E-Commerce. In Workshop dagli Oggetti agli Agenti, WOA 2006, Proc. of 7th, vol. 204 of CEUR Workshop Proceedings. CEUR-WS.org.

[19] Garruzzo S., Rosaci D. and Sarnè G.M.L., (2007). ISABEL: A Multi Agent e-Learning System That Supports Multiple Devices. In IEEE/WIC/ACM Int. Conf. on Intell. Agent Technology, 485–488. IEEE.

[20] Garruzzo S., Rosaci D. and Sarnè G.M.L., (2007). MARS: An Agent-Based Recommender System for the Semantic Web. In Distributed Applications and Interoperable Systems, vol. 4531 of Lecture Notes in Computer Science, 181–194, Springer, Berlin Heidelberg.

[21] Guttman R.H., Moukas A. and Maes P., (1998). Agents as Mediators in Electronic Commerce. Electronic Markets, 8(1):22–27.

[22] Huang Z., Chung W. and Chen H., (2004). A Graph Model for e-Commerce Recommender Systems. J. American Society Information Science Technology, 55(3):259–274.

[23] Jogalekar P. and Woodside M. Evaluating the Scalability of Distributed Systems. (2000). IEEE Trans. Parallel Distrib. Syst., 11(6):589–603.

[24] Kim J.K., Kim H.K. and Cho Y.H., (2008). A User-Oriented Contents Recommendation System in Peer-to- Peer Architecture. Expert Systems with Applications, (34):300–312.

[25] Kim Y.S., Yum B.J., Song J. and Kim S.M., (2005). Development of a Recommender System Based on Navigational and Behavioral Patterns of Customers in e-Commerce Sites. Expert Systems with Applications, 28:381–393.

[26] Konstan J. and Riedl J., (2012). Recommender Systems: from Algorithms to User Experience. User Modeling and User-Adapted Interaction, 22(1):101–123.

[27] Lax G. and Sarnè G.M.L., (2013). CellTrust: a reputation model for C2C commerce. Electronic Commerce Research, 8(4), 193-216.

[28] Lorenzi F., Correa F.A., Bazzan A., Abel M. and Ricci F., (2008). A Multiagent Recommender System with Task-Based Agent Specialization. Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis, 103-116, Springer Berlin Heidelberg.

[29] Manouselis N. and Costopoulou C., (2007). Analysis and Classification of Multi-Criteria Recommender Systems. World Wide Web, 10(4):415–441.

[30] Palopoli L., Rosaci D. and Sarnè. G.M.L. , (2013) A Multi-tiered Recommender System Architecture for Supporting e-Commerce. In Intelligent Distributed Computing VI, vol. 446 of Studies in Computational Intelligence, 71–81. Springer, Berlin Heidelberg.

[31] Postorino, M.N. and Sarnè, G.M.L., (2011). A Neural Network Hybrid Recommender System. In Frontiers in Artificial Intelligence and Applications, Proc. of the 2011 Conf. on Neural Nets WIRN10, 180–187, IOS Press, Amsterdam.

[32] Pu P., Chen L. and Hu R., (2012). Evaluating Recommender Systems from the Users Perspective: Survey of the State of the Art. User Modeling and User-Adapted Interaction, 22(4):317–355.

[33] Ratnasamy S. and McCanne S., (1999). Scaling End-to-End Multicast Transports with a Topologically- Sensitive Group Formation Protocol, In Proc. of ICNP 99, 79–88, IEEE.

[34] Rosaci D., (2005). Exploiting Agent Ontologies in B2C Virtual Marketplaces. Journal of Universal Computer Science, 11(6):1011–1039.

[35] Rosaci D., (2007). CILIOS: Connectionist Inductive Learning and Inter-Ontology Similarities for Recommending Information Agents. Information Systems, 32(6):793–825.

[36] Rosaci D., (2012). Trust Measures for Competitive Agents. Knowledge-Based Systems, 28:38–46.

[37] Rosaci D. and Sarnè G.M.L., (2006). MASHA: A Multi-Agent System Handling User and Device Adaptivity of Web Sites. User Modeling User-Adapted Interaction, 6(5):435–462.

[38] Rosaci D. and Sarnè G.M.L., (2009). TRES: A Decentralized Agent-Based Recommender System to Support B2C Activities. In Agent and Multi-Agent Systems: Technologies and Applications, vol. 5559 of Lecture Notes in Computer Science, 183–192. Springer, Berlin Heidelberg.

[39] Rosaci D. and Sarnè G.M.L., (2010). Efficient Personalization of e-Learning Activities Using a Multi-Device Decentralized Recommender System. Computational Intelligence, 26(2):121–141.

[40] Rosaci D. and Sarnè G.M.L., (2011). EVA: an Evolutionary Approach to Mutual Monitoring of Learning Information Agents. Applied Artificial Intelligence, 25(5):341–361.

[41] Rosaci D. and Sarnè G.M.L., (2012). A Multi-Agent Recommender System for Supporting Device Adaptivity in e-Commerce. Journal of Intelligent Information System, 38(2):393–418.

[42] Rosaci D. and Sarnè G.M.L., (2013). Cloning Mechanisms to Improve Agent Performances. Journal of Network and Computer Applications, 36(1):402-408.

[43] Rosaci D. and Sarnè G.M.L., (2012). Recommending Multimedia Web Services in a Multi-Device Environment. Information Systems, 38(2):198–212.

[45] Rosaci D., Sarnè G.M.L. and Garruzzo S., (2009). MUADDIB: A Distributed Recommender System Supporting Device Adaptivity. ACM Transaction on Information Systems, 27(4).

[44] Rosaci D., Sarnè G.M.L. and Garruzzo S., (2012). Integrating Trust Measures in Multiagent Systems. International Journal of Intelligent Systems, 27(1): 1-15.

[46] Rowstron A.I. and Druschel P., (2001). Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Proc. of Middleware 2001, IFIP/ACM Int. Conf. on Distributed Systems Platform, vol. 2218 of LNCS, 329–350. Springer, Berlin Heidelberg.

[47] Schafer J.B., Konstan J.A. and Riedl J., (2001). E-Commerce Recommendation Applications. Data Mining Knowledge Discovery, 5(1-2):115–153.

[48] Schifanella R., Panisson A., Gena C. and Ruffo G., (2008). MobHinter: Epidemic Collaborative Filtering and Self-Organization in Mobile Ad-Hoc Networks. In Proc. of 2008 ACM Conf. on Recommender Systems, RecSys 2008, 27–34, ACM, New York.

[49] Stoica I. and Morris R. and Karger D.R. and Kaashoek M.F. and Balakrishnan H., (2001). Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In Proc. of the ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Comm., 149–160, ACM, New York.

[50] Tanenbaum A.S. and Van Steen M., (2001). Distributed Systems: Principles and Paradigms. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[51] Tveit A., (2001). Peer-to-Peer based Recommendations for Mobile Commerce. In Proc. of the 1st Int. Work. on Mobile Commerce, 26–29, ACM, New York.

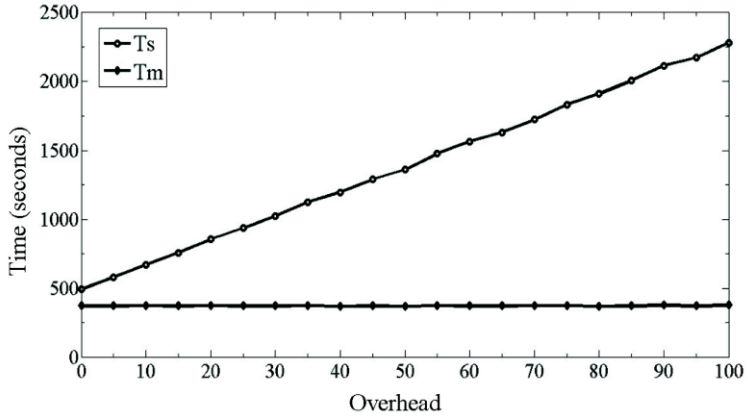
[52] Wei K., Huang J. and Fu S., (2007). A Survey of e-Commerce Recommender Systems. In Proc. of 13th Int. Conf. on Service Systems and Service Management (ICSSSM07), 1–5, IEEE Washington, DC.

[53] Weng L.T., Xu Y., Li Y. and Nayak R., (2006). A Fair Peer Selection Algorithm for an E-commerce Oriented Distributed Recommender System. In Proc. of the 4th Conf. on Advances in Intelligent Information Technology, 31–37, IOS Press. Amsterdam.

[54] Wooldridge M. and Jennings N.R., (1995). Agent Theories, Architectures, and Languages: A Survey. In Proc. of Work. on Agent Theories, Architectures, and Languages (ECAI-94), vol. 890 of LNCS, 1–39. Springer, Berlin Heidelberg.

[55] Yang X., Qiang Z., Zhao X. and Ling Z., (2007). Research on Distributed E-Commerce System Architecture. In Proc. of 8th ACIS Int. Conf. on Software Engineering (SNPD'07), Artificial Intelligence, Networking, and Parallel/Distributed Computing, 821–825, IEEE, Washington, DC.

[56] Zhong S., (2007). Privacy-Preserving Algorithms for Distributed Mining of Frequent Item Sets. Information Science, 177(2):490–503.


Ts
Tm
Time (seconds)
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90
100
Overhead