A Multi-tiered Recommender System Architecture for Supporting E-Commerce

Luigi Palopoli, Domenico Rosaci, and Giuseppe M.L. Sarné

Abstract. Nowadays, many e-Commerce tools support customers with automatic recommendations. Many of them are centralized and lack in efficiency and scalability, while other ones are distributed and require a computational overhead excessive for many devices. Moreover, all the past proposals are not "open" and do not allow new personalized terms to be introduced into the domain ontology. In this paper, we present a distributed recommender, based on a multi-tiered agent system, trying to face the issues outlined above. The proposed system is able to generate very effective suggestions without a too onerous computational task. We show that our system introduces significant advantages in terms of openess, privacy and security.

1 Introduction

Nowadays, a large number of *Recommender System*s (RSs) is used to promote e-Commerce (EC) activities [11] but often they fall when transactions occur between customers and merchants (B2C), mainly for a inadequate exploration of the market space, ineffective communications between the actors and for lack of security and privacy in the transactions. To solve such issues, new 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 without human intervention.

A RS provides his user with potentially useful suggestions for his purchases [14] based on a representation of his interests and preferences across the phases of a B2C transaction. Different behavioural models describe such phases, as the well known *Consumer Buying Behaviour* (CBB) model [6] base on six stages, namely: *i) Need*

Luigi Palopoli
Università della Calabria, 87036 Rende (CS) (Italy)
e-mail: palopoli@deis.unical.it

Domenico Rosaci · Giuseppe M.L. Sarné
Università Mediterranea, 89122 Reggio Calabria (Italy)
e-mail: {domenico.rosaci,sarne}@unirc.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 71–81.
springerlink.com

Identification; *ii) Product Brokering*; *iii) Merchant Brokering*; *iv) Negotiation*; *v) Purchase and Delivery*; *vi) Service and Evaluation*. Software agents [16] are usually exploited to monitor a user during these stages for building his profile. The RSs present in the EC sites can adopt centralized or distributed architecture. The first generate suggestions only on the server side but their performances are lacking in terms of efficiency and scalability and customers' privacy (due to the centralization of personal information), and this potentially affects the quality of their suggestions. The alternative approach implies distribution [9, 10] but its complexity could generate unacceptable computational overheads on the client as, for instance, a mobile device. Moreover, existing RSs assume homogeneous system components, implying that it is difficult for the users to add personal knowledge in the system.

In this paper, we present a **D**istributed **A**gent **R**ecommender for **E**-**C**ommerce (**DAREC**) based on a multi-tiered agent system. It allows to *i)* increase the distribution degree of the RS, *ii)* generate effective recommendations without any onerous computational task on the client side, *iii)* introduce significant advantages in openess and privacy. The basic idea of this framework (see Figure 1) is that each customer is assisted by three software agents, each of which, autonomously of the other agents, deals with a different CBB stage . Each agent runs on a different thread in the customer's client and this improves the efficiency of the overall process being agent interactions *specialized*. Each customer's agent, during its activity, can interact with the DAREC sellers' sites distributed over the Internet, where each seller site is assisted by a seller agent provided with both a *product catalogue* and *customers' profiles* encoding the preferences of each customer that visited the site in the past. This interaction allows the customer agent to generate content-based (CB) recommendations for the customer and also makes the site able to generate personalized presentations of the products for its visitors to support the site visit. The agents also interact with the seller agents and reciprocally generate 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 of c_1 and c_2 are free to perform other activities improving the system performances with respect to those systems where a unique customer's agent can execute only one activity at time.

The remaining of the paper is organized as follows. In Section 2 we introduce the knowledge representation exploited in DAREC, while Section 3 describes the agents' behaviour. Section 4 deals with some related work. Section 5 presents some experiments to evaluate our proposal and in Section 6 we draw our final conclusions.

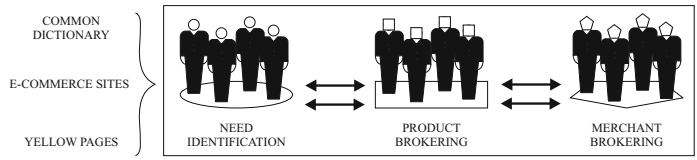

Fig. 1 The DAREC Architecture

2 The DAREC Knowledge Representation

The DAREC community shares a common dictionary storing the names of basic product categories of interest and their reciprocal relationships. Moreover, each customer's agent profile encodes all the information necessary to manage its CBB stage. Similarly, each seller's agent manages a catalogue of offered products organized in categories. Finally, in order to allow a collaboration between agents the information stored in a "yellow page" data structure are exploited (see below).

More in detail, a *Category Dictionary* $\mathscr{D}$ contains (*i*) a set $\mathscr{D}_{\mathscr{C}}$ of *product categories* and (*ii*) a set of *category links* $\mathscr{D}_{\mathscr{R}}$ between categories, denoted by $\langle cat_1, cat_2, t\rangle$ where cat_1, $cat_2 \in \mathscr{D}$ and t is the *type* of the link that can be: *i) isa-linked*, denoted $\langle cat_1, cat_2, ISA\rangle$, iff all the products belonging to cat_1 also belong to cat_2 ; *ii) synonymy-linked*, denoted $\langle cat_1, cat_2, SYN\rangle$, iff both all the products belonging to cat_1 also belong to cat_2 and vice versa; *iii) overlap-linked*, denoted $\langle cat_1, cat_2, OVE\rangle$, iff there exist some product of cat_1 that also belong to cat_2, and viceversa. Note that if two categories are synonymy-linked, they are also overlap-linked; *iv) commercial-linked*, denoted $\langle cat_1, cat_2, COM\rangle$ iff we suppose that the customers usually purchase both products belonging to cat_1 and cat_2.

We represent a category dictionary $\mathscr{D}$ (called COMMON and publicly available) as a direct labeled graph $G(\mathscr{D}) = \langle \mathscr{D}_C, \mathscr{D}_R\rangle$, where; for each category $cat \in \mathscr{D}_C$ there is a node called $name_{cat}$ associated with a label denoted by $info_{cat}$; for each arc $\in \mathscr{D}_R$ there is a link $\langle cat_i, cat_j, t\rangle$ oriented from cat_i to cat_j and labeled by t. Nodes represent product categories, which the products offered by the sellers belong to, and arcs represent existing relationships between categories. Moreover, we say that cat_i and cat_j belong to a *category relationship* (ISA($\mathscr{D}$), SIN($\mathscr{D}$), OVE($\mathscr{D}$) or COM($\mathscr{D}$)) if they are in the same dictionary and the relationships $\langle cat_i, cat_k, t\rangle$ and $\langle cat_k, cat_j, t\rangle$ belong to a more general (ISA, SIN, OVE or COM)-relationship, while they are called *generally related*, denoted by GEN($\mathscr{D}$), if there is in $\mathscr{D}$ a path between their associated nodes independently of the specific arc-labels.

2.0.1 The Personal and the Site Profiles

In a DAREC community ($\mathscr{C}$), in order to handle the Need Identification (resp. Product Brokering, Merchant Brokering) CBB stage, each customer $c \in \mathscr{C}$ is assisted in that stage by an agent, called NI_c (resp. PB_c, MB_c). Each agent stores in a profile, called NI (resp. PB, MB)-profile all the c's information necessary to handle the associated CBB stage. The profile is implemented by a category dictionary $\mathscr{P}(NI_c)$ (resp. $\mathscr{P}(PB_c)$, $\mathscr{P}(MB_c)$) such that its nodes represent categories of interest (resp. product categories relative to suitable products or merchants) for c, arcs represent links between categories and each category (resp. product of interest, merchant) is associated with a quantitative evaluation of the c's interest. Moreover, since a product (resp. merchant) search could be not activated for each category (resp. product) of interest for c, the categories belonging to $\mathscr{P}(PB_c)$ (resp. $\mathscr{P}(MB_c)$) are generally a subset of those in $\mathscr{P}(NI_c)$ (resp. $\mathscr{P}(PB_c)$). As a consequence, each arc of the PB (resp. MB)-profile, between cat_i and cat_j is a copy of the corresponding arc

belonging to the NI (resp. PB)-profile (included the label). Finally, each category belongs to either the common dictionary or to a "personal" customer's category (understandable to the other agents being in a general relationship with at least another category of the common dictionary).

In DAREC, each seller $s \in \mathscr{S}$ is associated with a seller agent that stores in its site profile (SP_s) a catalogue of the offered products and some information about the preferences of its past customers. Also SP_s is represented by a category dictionary whose nodes represent product categories in which s offers products and whose arcs represent relationships between categories. We introduce the mapping $SP_s(cat)$, that accepts the category cat as input and returns the tuple $SP_s(cat) = \langle prod, customers \rangle$, where $prod$ is a list of products that s offers in the category cat. To access the elements of this list, we use a mapping $SP_s(cat).prod(p)$ that accepts as input a product p and returns the tuple $SP_s(cat).prod(p) = \langle price, payment, format \rangle$, such that $price$ is price of p, $payment$ is the set of payment methods available for p and $format$ is the format available for p. The $price$ mean depends if the price is fixed or it is the reserved price in an "auction". Finally, the element $customers$ is a list of customers interested in products of the category cat that, for each customer, stores a list of those products of the site which the customer is interested in.

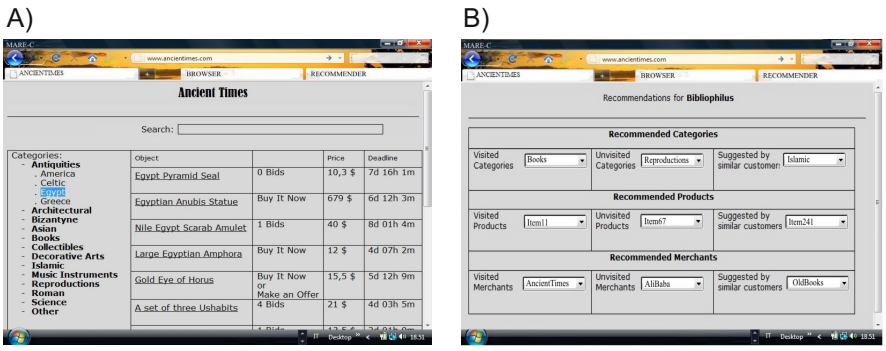

Fig. 2 A)An example of a DAREC site; B)The recommendations provided by the personal agents

2.0.2 The Yellow Pages YP

This data structure allows customers of the DAREC community to publish their interests, in order to facilitate mutual collaboration. YP is a set of category dictionaries $\{YP_c\}$, one for each customer c. In particular, YP_c is a sub-graph of c's NI-profile, containing those categories $cat \in \mathscr{P}(NI_c)$ such that $NI_c(cat).i = public$.

3 Agent's Behaviour in DAREC

The agent running on the client used by the customer c to visit the DAREC EC sites (i) supports the Web site navigation like a normal browser and (ii) generates

suggestions for the user. To this purpose, the client interface is provided with the functionalities *Browser* and *Recommender* described below. If c is a newcomer he should build an initial NI-Profile $\mathscr{P}(NI_c)$ by adding the categories of his interest, together their relationships, from the dictionary COMMON. Moreover, c could add to $\mathscr{P}(NI_c)$ some personal category $cat \notin COMMON$, by specifying its name and path in $\mathscr{P}(NI_c)$ joining cat with at least a category $cat^* \in COMMON$. Besides, for each selected category c should specify his interest degree $NI_c(cat).i$ and the visibility mode $NI_c(cat).mode$.

3.1 The Browser: Agents Working "over the Shoulders"

Each DAREC site allows the customer c *i)* to "navigate" through the categories by clicking on the tab "Browser" (the *Categories* list is on the left in Figure 2-A) or *ii)* to use the *Search* tool to perform a keyword-based search among the products sold at fixed price (i.e., *Buy-It-Now*) or with an auction (i.e., *Make an Offer*). For each product p, belonging to a category cat, c can perform the actions of: *(A1)* selecting the product for examining the offer; *(A2)* watching the product; *(A3)* purchasing the product. Each action A1, A2 or A3 performed by c implies a call to the agents NI, PB and MB that automatically update their profiles and in particular:

- The NI-agent is called, feeding it the category cat. If cat is absent in its profile, it is added therein and its interest value $NI_c(cat)$ is set to an initial value $iniInt$. Then the NI-agent requires to the PB and MB agents to add cat and its interest value to their profiles. Otherwise, if $cat \in$ NI-profile, its interest value is updated to $NI_c(cat) = min(1, NI_c(cat) + \Delta_a)$, where $\Delta_a \in [0,1]$ (with $a = A1, A2, A3$) it is arbitrarily set by c to weight the performed action. The value $NI_c(cat)$ is then passed to the agents PB and MB for updating their profiles.
- The client calls the PB-agent to pass the product p. If $p \notin$ PB-profile then it is added to the list $PB_c(cat).prod$ with $iniInt$ as interest value and their insertion in the the list $MB_c(cat).prod \in$ MB-agent is required. Otherwise, if $p \in$ PB-profile its interest value is updated to $PB_c(cat).prod(p).i = min(1, PB_c(cat).prod(p).i + \Delta_a)$ and passed to MB to be copied in $MB_c(cat).p.i$.
- The client calls the MB agent, passing the seller s. If $s \notin$ MB-profile, then it is added to the list $MB_c(cat).sellers$ with $numT = 1$ and a score of $iniInt$. Otherwise, $numT$ increased by 1 and the score is updated to $MB_c(cat).sellers(s).ev = min(1, MB_c(cat).sellers(s).ev + \Delta_a)$.

Periodically, the $NI_c(cat).i$ (resp. $PB_c(cat).prod(p).i$, $MB_c(cat).sellers(s).ev$) value associated with the NI (resp. PB, MB)-profile, after a τ_{NI} (resp. τ_{PB}, τ_{MB}) time period passed from its last update, is decreased of ρ_{NI} (resp. ρ_{PB}, ρ_{MB}), a c's parameter ranging in $[0,1]$. Also the seller agent of s updates its list $SP_s(cat).customers \in SP_s$ after each customer's action that involves a product $p \in cat$. In particular, if $c \notin SP_s(cat).customers$ a new element $SP_s(cat).customers(c)$ is added to it. Moreover, p is added to the list $SP_s(cat).customers(c).prod$ and the number of transactions $SP_s(cat).customers(c).numT$ is increased. Otherwise, if $c \in SP_s(cat).customers$, the

seller agent updates the element $SP_s(cat).customers(c)$ by increasing the number of transactions $SP_s(cat).customers(c).numT$ and inserting p in $SP_s(cat).customers(c).prod$ if it is absent.

3.2 The Recommender: Exploiting Customer's Profiles

When c selects the tab "Recommender" in his client, then some suggestions are generated for him and visualized in a page having a section for each supported stage (Figure 2-B). Suggestions are generated by the each agent in a "cascade" mode, i.e. firstly the customer chooses a category from those in section "Recommended Categories", then a set of products is suggested in section "Recommended Products" and chosen a product, a set of merchants selling that product is suggested in section "Recommended Merchants".

The **NI-agent** suggests to c a set of categories visualized in the client section "Recommended Categories" in the three distinct list-boxes (Figure 2-B):

• **Visited Categories** contains categories selected with a CB approach from the NI-profile $\mathscr{P}(NI_c)$ built by monitoring the c's activity (see Section 3).
• **Unvisited Categories** lists categories unknown to c, but considered interesting for him by his NI-agent. This agent uses a *relationship-based* mechanism to exploit the interaction between the c's NI-agent and the agent 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. $c \in SP_s(cat).customers$), determines all the categories $cat^* \in SP_s$ such that cat^* is unvisited by c and there exists a path in SP_s between cat and cat^*; these categories are sent to the c's NI-agent, to be added to this list.
• **Suggested by Similar Customers**, where the categories are determined, with a CF technique, on the whole EC customer's navigation history by the c's NI-agent collaborating with the NI-agents of customers similar to c for interests. To this aim, it is exploited the public repository YP (see Section 2) storing, for each DAREC customer x his public interest profile YP_x. The c's NI-agent computes the similarity degree $s(c,x)$ between its profile $\mathscr{P}(NI_c)$ and each YP_x by using the Jaccard measure of the set of nodes of $\mathscr{P}(NI_c)$ and YP_x for n customers (a parameter set by c), that is $s(c,x) = |NODES(\mathscr{P}(NI_c)) \bigcap NODES(YP_x)| / |NODES(\mathscr{P}(NI_c)) \bigcup NODES(YP_x)|$, where NODES(G) returns the set of nodes of its input graph. Then, the c's NI-agent determines, for each similar customer x, those categories stored in the public profile $YP_x \notin \mathscr{P}(NI_c)$, and adds them to this list.

The **PB-agent** (resp. **MB-agent**) suggests, in the section "product recommendations" (resp. "merchant recommendations") 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 listboxes:
• **Visited Products (resp. Merchants)**, it contains products (resp. merchants) of the category $cat \in$ PB-profile $\mathscr{P}(PB_c)$ (resp. MB-profile $\mathscr{P}(MB_c)$), ordered by score.
• **Unvisited Products (resp. Merchants)**, this list is built by exploiting a collaboration between the c's PB (resp. MB)-agent and the seller agent of each site s the

customer c visited in the past. In particular, for a PB-agent each listed $p \in cat$ is unvisited by c and belongs to each site profile, while for the MB-agent the set is formed by those sellers having *cat* in their profiles and are unvisited by c in the past.
• **Suggested by Similar Customers**, these suggestions are based on the collaboration between the PB (resp. MB)-agents of c and of other customers similar to him for interests (their list is provided by the c's NI-agent). Each of the PB (resp. MB)-agent of these customers sends to c's PB (resp. MB)-agent its set of products $PB_x(cat).prod$ (resp. merchants $MB_x(cat).sellers$) that is added to this list. The PB (resp. MB)-agent shows the products (resp. merchants) belonging to the listbox "Visited Products" (resp. "Visited Merchants"), ordered by value, and the products (resp. merchants) belonging to the listboxes "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 SP_s to personalize the site presentation for the each customer c that is visiting it. When c returns to visit the site, the element $SP_s(cat).customers(c)$ already exists in the SA_s profile. Using such information, SA_s personalizes its home page for c by visualizing in a "Shop Window" all the $p \in SP_s(cat).customers(c).prod$, ordered by interest value. SA_s uses this list as a sort of *local profile* related to c and at the same time, it increments the value $SP_s(cat).customers(c).numV$ to consider his current visit. Otherwise, for the first time c's visit the default home page is visualized.

4 Related Work

Centralized RSs are widely used in EC Web sites, for example, the *Amazon* site [1] adopts some recommender tools based on the customer's browsing history, past purchases and purchases of other customers. The system drives a customer to buy something because this is related to something that he purchased before, or because this is popular with other customers. Another case is that of the RSs embedded in auction sites [4], as in eBay [5], that generates recommendations using its feedback profile features. Among the centralized approaches in [9] the authors propose a system which stresses on freshness, novelty, popularity and limitedness in product recommendations. All these centralized approaches generate, similarly to DAREC, both CB and CF recommendations and store all the private information about customers and sellers necessary to generate recommendations. Differently to DAREC is that they are not open and use pre-defined dictionaries of terms for defining the product categories while DAREC allows the users to define new terms in their personal ontologies. Distributed recommender systems (DRSs), increased in diffusion in these later years, share information and computation tasks among more entities. DRSs are more critical in design and performances optimization [8] than centralized RSs but *(1)* promise scalability in time and space complexities, *(2)* avoid the failure risks due to a central database running on a unique server and *(3)* preserve privacy and security. DSRs often adopt peer-to-peer (P2P) and agents technologies to easily *i)* exchange data locally stored on each peer (e.g., Chord [13]) reducing the task to locate a specific resource and *ii)* provide communication, cooperation and

negotiation rules, respectively. In a mall, adaptive learning agents associated with each shop in CASy [3] process shop transactions and analyze information about interests of customers entering the shop that are derived by his profile, keywords, product queries and by other shop agents for proposing to the consumer suitable suggestions in an efficient and adaptive way. A multi-agent system (MAS) implementing a knowledge-based DSR applied to the tourism domain is discussed in [7]. Agents cooperate to suggest travel packages to a user. They are expert in specific domains (hotel, flights, interchanges, etc) and autonomously select the most suitable sub-task of a recommendation request to deal with their competence. Their suggestions are then composed in a final recommendation. All the referred DRSs take advantage of the distributed architectures in terms of scalability, risks failure, privacy and security. Differently from DAREC, none of them deal with the Need Identification, Product or Merchant Brockering phases based on a description of the consumer's interests and preferences (with the partial exception of [2] that can also avoid the use of such a profile). All the described recommenders and DAREC adopt (or can easily adopt, as [12, 15]) agent-based and/or P2P systems to find similar neighbors, resources and exploit predefined services. Agent specialization is introduced in [7] but it is relative just to the item typology, while DAREC is based on agent communities, where each agent is specialized in a different EC phase (as described in the CBB model). A similar concept is also present in [15] but it involves a set of recommenders, each one linked to a different organizations and having a particular point of view in generating suggestions.

5 Efficiency and Effectiveness of DAREC

In this Section, we discuss efficiency and some experimental results about the advantages of our approach. In terms of efficiency, for a community of n customers and m sellers, a unique centralized agent managing all the three phases has computational cost of $NC = \Sigma_{i=1}^{3} n_i \cdot k_i$, where k_i and n_i are the number of contemporary sessions activated and of operations needed for a user to manage the Need Identification, Product and Merchant Brokering phases, respectively. Instead, in DAREC, for a given CBB stage, each user's agent can deal with more different issues running on different threads. Let α_i, be the multi-threading degrees for a specific CBB stage and let $\beta = k_1 + k_2$ be the computational overhead due to the communications between the local agents (i.e., the Need Identification agent calls the Product Broker agent, that in its turn can call the Merchant Broker agent that does not run any additional operation). In this way the computational cost for a CPU will be $ND = \beta + \Sigma_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i$ and the computational advantage (ρ), due to the distribution in DAREC, is equal to $\rho = (\beta + \Sigma_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i) / \Sigma_{i=1}^{3} n_i \cdot k_i$ 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 $\rho = \alpha + 2/N$. Therefore, the advantage of using DAREC is perceivable with a small multi-threading contribution (i.e. high values of α) in presence of a reasonably high number of operations (i.e. an intense EC activity), while for a high multi-threading activity the advantage shows up also for small N values.

In terms of effectiveness, the time exploited to perform B2C processes in serial and multi-threading way has been compared by means of a software appositively designed. To this aim, we considered a period of 2 hours where a set of 500 customers finalize all their B2C processes with a purchase dealing with a merchant population (M) of 10 units. Moreover, the merchant has to satisfy also the requests due to other customers that could absorb significant merchants' servers resources. This is taken into account by means of an overhead (O) of $1 \div 100$ requests for second, randomly shared among the merchants. Finally, a lot of different of communication, computational and behavioural parameters have been tuned to model realistic B2C processes. Obviously, in order to compute in 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, T_m (i.e. 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.

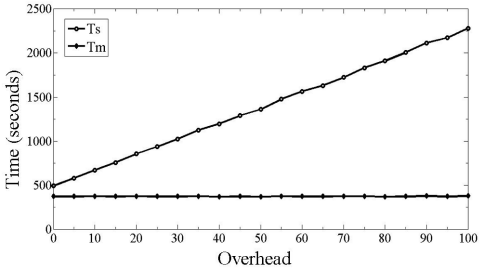

Fig. 3 The average serial (T_s) and multi-threading (T_m) times (in sec.) needed to carry out a B2C process depending on the Overhead by considering 500 Customers and 10 Merchants

The experimental results shown in Fig. 3 confirm that the DAREC approach consumes in average about the 25% of time less then the serial approach in performing a purchase in absence of overhead and when the overhead grows also T_s grows with it while T_m is almost uniform. In Table 1 are reported the average gain (G) in percentage of T_m with respect to T_s for different values of the overhead. This behavior is due to the fact that changes in the number of merchants, overheads and so on, have a minimal impact on T_m and very high impact on T_s. This because, in average, each merchant's server is busy to satisfy the customers' requests and T_s grows with the level of "saturation" of the merchants' servers worsening the quality of their service.

Table 1 The average gain ($G = T_s/T_m$) to carry out a B2C process depending on the number of Overhead by considering 500 Customers and 10 Merchants

O	0	5	10	20	30	40	50	60	70	80	90	100
G	24.61	36.12	44.63	56.43	63.98	69,19	73,05	76,39	78,39	80,75	82,28	83,61

6 Conclusions

In this paper, we have presented advantages and limitations of the DAREC distributed architecture that introduces novel, original characteristics with respect to other recommenders. DAREC allows to 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 cost for the device on which the local agents run and the presence of specialized agents improves the users' knowledge representations, the openness of the system and the privacy degree.

Acknowledgements. This work was partially funded by the Italian Ministry of Research through the PRIN Project "Entity Aware Search Engines" and by the DEIS (Università della Calabria).

References

1. Amazon (2011), `http://www.amazon.com`
2. Awerbuch, B., Patt-Shamir, B., Peleg, D., Tuttle, M.R.: Improved Recommendation Systems. In: Proc. of 16th ACM-SIAM Symp. on Discrete Algorithms, pp. 1174–1183. SIAM (2005)
3. Bohte, S.M., Gerding, E., La Poutré, J.A.: Market-based Recommendation: Agents that Compete for Consumer Attention. ACM Trans. Internet Techn. 4(4), 420–448 (2004)
4. Culver, B.: Recommender System for Auction Sites. J. Comput. Small Coll. 19(4), 355–355 (2004)
5. eBay (2011), `http://www.ebay.com`
6. Guttman, R.H., Moukas, A., Maes, P.: Agents as Mediators in Electronic Commerce. Electronic Markets 8(1), 22–27 (1998)
7. Lorenzi, F., Correa, F.A.C., Bazzan, A.L.C., Abel, M., Ricci, F.: A Multiagent Recommender System with Task-Based Agent Specialization. In: Ketter, W., La Poutré, H., Sadeh, N., Shehory, O., Walsh, W. (eds.) AMEC 2008. LNBIP, vol. 44, pp. 103–116. Springer, Heidelberg (2010)
8. Olson, T.: Bootstrapping and Decentralizing Recommender Systems. Ph.D. Thesis, Dept. of Information Technology. Uppsala Univ. (2003)
9. Parikh, N., Sundaresan, N.: Buzz-based Recommender System. In: Proc. of 18th Int. Conf. on World Wide Web (WWW 2009), pp. 1231–1232. ACM (2009)
10. Rosaci, D., Sarné, G.M.L., Garruzzo, S.: MUADDIB: A Distributed Recommender System Supporting Device Adaptivity. ACM Trans. Inf. Syst. 27(4) (2009)
11. Schafer, J.B., Konstan, J.A., Riedl, J.: E-Commerce Recommendation Applications. Data Min. Knowl. Discov. 5(1-2), 115–153 (2001)
12. Schifanella, R., Panisson, A., Gena, C., Ruffo, G.: MobHinter: Epidemic Collaborative Filtering and Self-Organization in Mobile Ad-Hoc Networks. In: Proc. of ACM Conf. on Recommender Systems (RecSys 2008), pp. 27–34. ACM (2008)
13. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. of SIGCOMM 2001, pp. 149–160 (2001)

14. Wei, K., Huang, J., Fu, S.: A Survey of E-Commerce Recommender Systems. In: Proc. of 13th Int. Conf. on Service Systems and Service Management, pp. 1–5. IEEE (2007)
15. Weng, L.-T., Xu, Y., Li, Y., Nayak, R.: A Fair Peer Selection Algorithm for an e-Commerce-Oriented Distributed Recommender System. In: Proc. of 2006 Conf. on Adv. in Intell. IT, pp. 31–37. IOS (2006)
16. Wooldridge, M., Jennings, N.R.: Agent Theories, Architectures, and Languages: A Survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI 1994 and ATAL 1994. LNCS, vol. 890, pp. 1–39. Springer, Heidelberg (1995)