Task allocation in case-based recommender systems: a swarm intelligence approach

Fabiana Lorenzi[1, 2]
Rua Carlos von Koseritz, 720/402 – Porto Alegre, RS – Brasil
CEP 90.540-030
+55-51-34774000

Daniela Scherer dos Santos[1]
Rua Milan Krás, 1426/11 - Cachoeira do Sul, RS - Brasil
CEP 96508-011
+55-51-92516146

Denise de Oliveira[1]
Porto Alegre, RS - Brasil - Caixa-Postal: 15064
CEP 91501-970
+55-51-33166823

Ana L. C. Bazzan[1]
Porto Alegre, RS - Brasil - Caixa-Postal: 15064
CEP 91501-970
+55-51-33166823

[1]Universidade Federal do Rio Grande do Sul
[2]Universidade Luterana do Brasil

{lorenzi, dsradtke, edenise, bazzan}@inf.ufrgs.br

Task allocation in case-based recommender systems: a swarm intelligence approach

KEYWORDS: Multiagent systems, Swarm intelligence, Case-based reasoning, Recommender systems, Case-based recommender systems, Task allocation, Negotiation among agents, Self-organization, Clustering.

ABSTRACT

Case-based recommender systems can learn about user preferences over time and automatically suggest products that fit these preferences. In this paper we present such a system, called CASIS. In CASIS we combined the use of swarm intelligence in the task allocation among cooperative agents applied to a case-based recommender system to help the user to plan a trip.

INTRODUCTION

In the tourism branch, travel agencies have a hard task to plan a trip because there are dependencies on information that changes over time. Besides, this information is distributed in the Internet (in sites of flight companies, hotels, city's attractions etc.). Another difficulty is that frequently the trip destination is not fixed. Finally, it is hard to plan according to the user's preferences when one has not all the necessary information.

Thus, a tool is necessary to retain the knowledge about the most popular destinations (e.g. within a season) and the travel agency experience. To solve those questions we propose an approach that combines case-based recommender and multiagent systems. The overall system has two different layers (with different goals): to search the case base and to search new information in the Internet. This work focuses on the former, proposing a metaphor from swarm intelligence to help the negotiation process among agents.

A multiagent system (MAS) needs to deal with dynamic problems such as variation in the number of agents, changes in environment and in the system's goals. These organizational issues depend on the system's goals, the perceived environment, and the relationships among agent's activities, as well as their interactions.

There have been many propositions to tackle these issues whose discussion is outside the scope of this paper. However, we are interested in the use of swarm-based approach. In (Ferreira et al, 2005) for instance, the authors proposed the use of swarm-based approach to adapt organizations in a MAS. This approach is useful because it deals with dynamic organizations, the same kind of problem posed by the tourism scenario just described.

This paper is organized as follow: section 2 presents a background of case-based recommender systems, negotiation among agents and swarm intelligence. Section 3 shows the CASIS system and section 4 presents some experiments and the results. Finally, the future trends and the conclusions are discussed.

BACKGROUND

Case-Based Recommender Systems

Case-based reasoning (CBR) is a problem-solving methodology that deals with a new problem by first retrieving a past, already solved similar case, and then reusing that case for solving the new problem (Kolodner, 1993). A case models a past experience, storing both the problem description and the solution applied in that context. All the cases are stored in the case base. When the system is presented with a new problem to solve, it searches for the most similar case(s) in the case base and reuses an adapted version of the retrieved solution to solve the new problem.

CBR is a cyclic and integrated problem solving process that supports learning from experience (Aamodt & Plaza, 1994) and has four main steps: retrieve, reuse, adaptation and retain. The adaptation phase is split into two sub-steps: revise and review. In the revise step, the system adapts the solution to fit the specific constraint of the new problem, whereas in the review step the constructed solution is evaluated by applying it to the new problem, understanding where it fails, and making the necessary corrections. CBR is one of the most successful machine

learning methodologies that exploit a knowledge-rich representation of the application domain (Aamodt & Plaza, 1994; Watson1997).

In a CBR recommender system (CBR-RS) a set of suggestions or recommendations is retrieved from the case base by searching for cases similar to a case described by the user (Burke, 2000). In the simplest application of CBR to recommend, the user is supposed to look for some product to purchase. He/she inputs some requirements about the product and the system searches the case base for similar products (by means of a similarity metric) that match the user requirements. A set of cases is retrieved from the case base and these cases can be recommender to the user.

If the user is not satisfied with the recommendation he/she can modify the requirements, i.e. build another query, and a new cycle of the recommendation process is started. The case retrieval is typically the main step of the CBR cycle and the majority of CBRRS can be described as sophisticated retrieval engines. For example, in the Order-Based Retrieval (Bridge & Ferguson, 2002) the system uses special operators to retrieve a partially-ordered set, whereas in the Compromise-Driven Retrieval (Mcsherry, 2003) the system retrieves similar cases from the case base but also groups the cases, putting together those offering to the user the same compromise, and presents just a representative case for each group.

Negotiation among agents and swarm intelligence

In a MAS it is necessary that agents coordinate their actions because they do not have a global view, and so the goals and knowledge are local, making it difficult for agents to cooperate. Besides, the system should be able to deal with global constraints, which are not perceived by the individual agents' local view, and with inter-agents dependencies.

In a more general way, coordination increases the MAS performance. There are many ways to coordinate agents in a MAS, classified by (Nwana et al, 1996) in four categories: organizational structuring, contracting, multi-agent planning, and negotiation.

Negotiation is a process where two or more parties make a joint decision. The negotiation can be classified as competitive or cooperative. We are interested here in the cooperative negotiation that can be viewed as a distributed search process where the agents try to reach the maximum global utility.

The negotiation process starts when the agent realizes that the agent is not able to perform a given task. This can happen for many reasons: the agent can be overloaded, the agent does not have capacity to perform the task, or maybe the agent knows that another agent can perform the task with a higher quality. When the agent has to perform the task, it calculates the task's utility (e.g. cost and time) and decides if it is necessary to start negotiation process.

One way to achieve cooperation without explicit communication is to use swarm intelligence, an artificial intelligence technique based around the study of collective behavior in decentralized, self-organized, systems. The use of the social insect metaphor to solve computer problems such as combinatorial optimization, communications networks or robotics is increasing (Bonabeau et al, 1999).

Social insects living in colonies, e.g. ants, termites, bees, and wasps distinguish themselves by their organization skill without any centralized control (Camazine et al, 2003; Gordon, 1996). This self-organization emerges from interactions among individuals, between the individuals and the environment, and from behaviors of the individuals themselves (Bonabeau et al, 1999).

Models based on self-organization assume that it might be possible to explain something apparently complex in terms of simple interacting processes (Bonabeau et al, 1999). There are several approaches inspired in social insects' behavior; the complete list is outside the scope of this paper. However, one example of the self-organization can be seeing in the way in which ants travel to and from a food source. They deposit a certain amount of pheromone while walking, and each ant prefers to follow a direction rich in pheromone. This enables the ant colony to quickly find the shortest route. In Dorigo (1998), the AntNet, an adaptive, distributed routing algorithm for telecommunication networks is described, which was inspired on the ant colony behavior described above.

Another important behavior related to ant colony is the division of labor, which is fundamental to the organization of insect societies (Robinson, 1992). Colonies respond to changing internal and external conditions by adjusting the ratios of individual workers engaged in the various tasks. Based on this behavior, some models were proposed. Ferreira (2005), for instance, applied one of these models to task allocation in dynamic environments where the agents adapt to changes in the organization, just as social insects do.

Another example of self-organization is found in the bees. Bees travel up to 10 km away from the hive to collect nectar. They return with nectar and information about the nectar source (Camazine et al, 2003). This process of dispatching bees into the surrounding countryside to gather the colony's food is called foraging.

In (Seeley et al, 1991) the authors presented an experiment showing the recruitment of nectar forager bees to feeders containing sugar solutions. Two food sources are presented to the colony at 8:00 a.m. at the same distance from the hive: source A is characterized by a sugar concentration of 1.00 mol and source B by a concentration of 2.5 mol. At the noon, the sources are exchanged: source A is now characterized by a sugar concentration 2.5 mol and source B by a concentration of 0.75 mol. In both situations the better source is more visited. This experiment showed that the foraging decision of each bee is based on very limited information of its own particular visited source. This simple behavior allows the colony to select the best quality source of nectar.

Based on these experiments, Camazine and Sneyd (1991) developed a mathematical model, which shows that he colony is capable to select the best nectar source available through three simple probabilistic behavioral rules:

- f_d^a and f_d^b: are the probability of sharing the nectar source information, A and B respectively, by dancing, a behavior in which a bee communicates to other bees the direction, distance, and desirability of the food source, trying to recruit new bees to that food source;
- f_x^a and f_x^b: are the probability of abandoning the food source A and B, respectively, and go to the area inside the hive called the dance floor to observe dancing bees and select its next food source;
- $A(f_l^a)$ and $B(f_l^b)$: is the probability of following a dancer for A and B, respectively.

The probability of following a dancer for source A and B is estimated by:

$$A(f_l^a) = \frac{D_a d_a}{D_a d_a + D_b d_b} \text{ and } B(f_l^b) = \frac{D_b d_b}{D_a d_a + D_b d_b}$$

Where:

D_a and D_b are the number of bees dancing for source A and B, respectively;

d_a and d_b are the time spent dancing to source A and B, respectively.

The Camazine and Syneid's model shows how the properties of the system emerge automatically from the dynamic interactions among the constituent components.

THE CASIS SYSTEM

Many researches are generating new solutions to show how e-commerce can be used for travel and tourism, trying to reproduce the traditional travel agents advice. In (Ricci et al, 2003), the authors describe an intelligent recommender system developed with Case-Based Reasoning approach that helps a traveler to select a tourist destination, building a set of products and composing a plan for the travel. Dietorecs (Fesenmaier, 2003) is also a recommender system that incorporates a human decision model that stresses individual differences in decision styles.

In this paper we present a case based swarming intelligence recommender system (called CASIS) to the forfeit's planning task using agents to recommend travel packages to the user. Forfeit is a special travel package created by the travel agent with flights, hotels and further services that match with the customer's preferences. The process starts with the user's query (that we call demand) and the metaphor of the honey bees is used to select the most similar cases to the user's demand. Similar works were not found in the literature to recommend travel packages based on swarm intelligence and case-based reasoning.

The Camazine and Sneyd's mathematical model (1991) was adapted and used in the negotiation process. Each agent represents a bee with the following behaviors: probability of abandoning the nectar source *i* (P^i_X); probability of recruiting more bees by dancing (P^i_D) to source *I* and probability of following the source *i* (P^i_F). P^i_F was adapted in our approach because we are not taking into account the time the bee spend dancing. Thus, $P_F^i = \frac{D_i}{\sum_{j=1}^{M} D_j}$ where D_i is the number of dancers bees to the source *i* and *M* is the total number of sources. Notice that P^i_F is the proportion of bees dancing to the source *i* related to all the bees that are dancing to the other sources.

There is a case base with past travel cases and the agents search in this case base for possible travel to recommend according to the users' preferences or the user's profile. Each case is viewed as a nectar source for the bees. Given a new user query, the metaphor calls for honey bees leaving the nest looking for nectar sources. In our system the original model was modified and the bee can forage for several different sources (not only two as in the original model).

According to Camazine and Sneyd's model, each bee's choice is calculated by using probabilities. The most important one is the abandon probability (P^i_X), where the bee checks the quality of the nectar source and decides whether or not to continue in this source. This probability was adapted in our model. In a traditional case-based recommender system, the new query is compared with all cases in the case base and the most similar is shown to the user. However, in CASIS, the similarity is used to help the bee to make its decision to continue in the nectar source or not. Randomly, the bee visits a case in the case base, comparing the query with this case and it calculates P^i_X. P^i_X calculates the euclidean distance between the new query and the cases. The cases with the smallest distance to the new query are the most similar, i.e. they represent the best nectar sources.

Figure 1 shows the model with the hive having two compartments (*Dancing for Case* and *Observing a Dancer*), the case base (representing the available nectar sources), and also the bee's decisions possibilities. As we can see, the bee can decide to visit the compartment *Dancing for Case* to perform recruitment's dance before returning to its case C_i , or to continue to forage at the case without recruiting other bees. Another possibility is to abandon the case and then visit the compartment *Observing a Dancer* (that has bees dancing for different cases). Here the bee selects (randomly) a dancer bee and decides to follow C_j (the case visited by the dancer bee) or to select a different dancer bee. In each one of these possibilities, the respective probabilities are calculated.

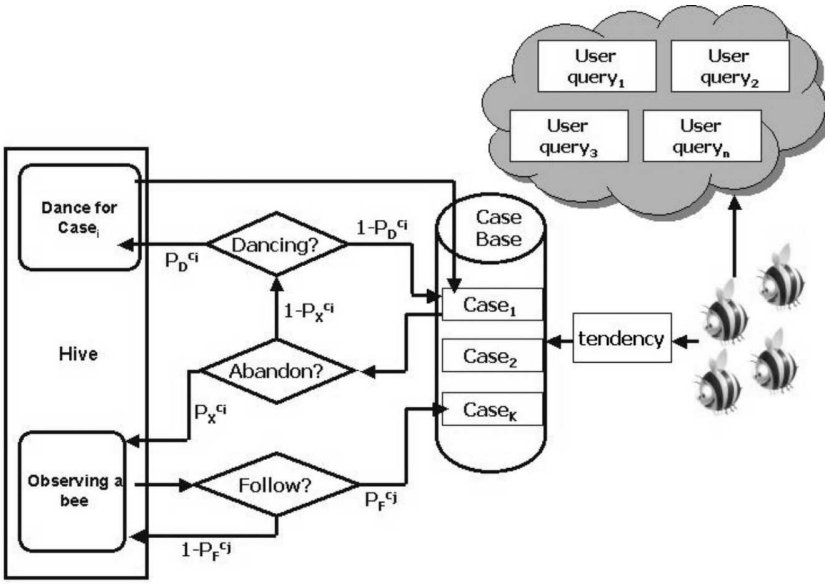

Figure 1: Dancing metaphor – adapted from Camazine & Sneyd

To improve CASIS, allowing that several passengers can request a recommendation at the same time, we found inspiration in the Bonabeau's task allocation model. This adapted model allows the recommender system to have several bees working in different demands.

The bee picks a demand to solve and calculates the *tendency* (T) that is the probability to visit a specific case, instead of visiting all cases from the case base. A bee will visit a case stored in the case base according to the tendency associated to this case. It means that the bee has a tendency associated to each pair of case/demand. This tendency is inspired on Bonabeau's mathematical model (Bonabeau et al, 1999) and it is calculated using the following equation:

$$T_{ij} = \frac{S_{ij}^2}{S_{ij}^2 + \theta_{ij}^2}$$

where:

S_{ij} represents the stimulus associated to the case *i* and to the demand *j*;

θ_{ij} is the bee's answer threshold associated to the case *i* and to the demand *j*.

θ_{ij} and S_{ij} are updated as follow:

- if the bee dance to recruit new bees to the case it is visiting:
  $\theta_{ij} = \theta_{ij} - \alpha$, if $\theta_{ij} > 0$
  $\theta ij = 0$, otherwise

$S_{ij} = S_{ij} + \alpha$

- if the bee does not dance and abandon the case:
  $\theta_{ij} = \theta_{ij} + \alpha$
  $S_{ij} = S_{ij} - \alpha$
  where α is a constant.

θ_{ij} and S_{ij} work as a reinforcement to those cases that are most similar to the user query, avoiding that the bees visit cases not relevant to the demand that is being solved.

The tendency helps the bee to decide which case it should visit, depending on the demand it is solving, reducing the search space and avoiding waste of time. In the moment the bee is working on a demand, it will continue in that demand until the recommendation appears. This behavior guarantees that a recommendation is always given to the user, since the bees will work until the best possible recommendation shows up.

As shown in the Recommendation's algorithm, when the user informs his/her preferences (a new request) the negotiation process is called. All the bees that are not busy (unemployed), i.e., the bees that are not working in any demand, are candidates to work in this new demand.

Let us assume a specific demand, shown in Table 1. When the user informs all his/her preferences, the random process of task allocation (which bees will work in this demand) starts. Each bee now has to select (randomly) which case it will visit. To decide about this, the bee calculates the Tendency (T) for that case (*i*) given demand (*j*).

Table 1: Example of the user preferences.

Destination	Transportation	Hotel	Local	Period	Price
Ibiza	Plane	Five-star hotel	Beach	10/01/2007 to 20/01/2007	Up to U$ 3.000,00

Following the recommendation cycle, the bee visits a case *i* (according to the calculated tendency) and now it has to calculate the probabilities (which means choose an option): abandon the case, follow a case or dancing for a case. The first probability calculated is the *abandon* - P^{cij}_X. With this probability, the bee decides if it has to abandon the case or not. It means that a low P^{cij}_X value indicates a low probability of the bee to abandon the case and a greater similarity between the visited case and the user's preferences.

If the bee decides to abandon case *i*, then it will randomly select a dancer bee to observe. There are always bees in the *Dancing for Case* compartment, dancing to recruit new bees to a specific case (case *j*). When observing the dancer bee, the bee calculates P^{cxj}_F that represents the probability of visiting case *j*. If it decides to visit it, the whole cycle restarts. The bee visits case *j*, calculates the probability of abandon the case and so on. On the other hand, if the bee decides not to visit case *j*, it will select a different dancer bee until P^{cxj}_F is satisfied. When the probability P^{cij}_X leads the bee to decide not abandoning the case it is visiting, it goes to the *Dancing for Case* compartment to recruit new bees to this case. So $P^{cij}_D = 1 - P^{cij}_X$.

The process is repeated until the most similar case regarding the user's preferences emerges and it is immediately recommended to the user in a textual form.

```
Recommendation's algorithm
N: number of bees
M: number of cases (sources)
CB: case base –(where CB = C1, C2, …, CM)
D: number of demands
DB: demand base –(DB = D1, D2, …, DR)
q: user's query
θij = 1
Sij = 1
for b = 1, ..., N do
   dj = random (DB)
   ci = random (CB)
   Tij = S2ij / S2ij + θ2ij
   if (Tij met) then
   begin
      //the bee goes to the case cij
      repeat
         Pcij X = Euclidean Distance (q, cij)
         if ((Pcij X) not met) then
         begin
            Pcij D = 1 - Pcij X
            if (Pcij D met) then
            begin
               //visit the compartment Dancing for Case
               θij = θij – α
               Sij = Sij + α
            end
            else keep foraging
            end if
         end
         else begin
            //visit the compartment Observing a dancer
            θij = θij + α
            Sij = Sij - α
            repeat
               //choose a random dancer bee
               Pcxj F = Dcxj / Σ^M y=1 Dy //Dcxj is the number of bees
                                  //dancing to pair Cx and demand dj
               if (Pcxj F met) then
                  ci = cx //bee will forage cx
               end if
            until (ci = cx)
         end
         end if
      until (number of bees working in case i and demand j ≥ 80%
            of the total number of bees working in the demand j)
   end
   end if
end for
return Most_danced_case //The case with the highest number of
                        //bees dancing.
```

The tourism scenario is a dynamic domain and the recommendation demands are seasonal. As an example, we can say that the travel packages can be divided in two different clusters. In summer, users prefer beaches and sunny places, while in winter, usually the users prefer ski resorts. Following this idea, we can see that using the honey bee dance metaphor this clustering behavior emerges and if the user starts a query asking for some beach package, the bees will concentrate in cases that have summer travel packages, avoiding the cases with winter packages. The advantage of this behavior is that the bees can specialize themselves in some cases, according to the demands. However, it is important to mention that the bees can surprise the user with different cases if he/she does not define some preferences. The bees can use both clusters in this situation.

The advantage of using this metaphor is that the bees always return something to recommend. Despite the controversy that sometimes it is better not recommend instead of recommend wrong products, this is not true in the tourism context, especially in the frequent case where the client does not know exactly where to go or what to do. Our approach always returns some recommendation, which is specific for the current season, given the dynamic nature of the approach (bees adapting to the environmental changes which is here the travel demands of the current season).

EXPERIMENTS AND RESULTS

To validate the algorithm proposed, we have performed some experiments with different users' queries and also different parameters such as number of bees and number of cases in the case base. This section shows some experiments done in CASIS and the results achieved.

Parameters

The experiments were performed with 50 and 100 cases in the base case. Different number of bees was used to evaluate the performance of the system such as 200, 500, 1000, 2000, and 2500. However, the variation of these parameters did not present differences in the results.

We report results related to the experiments with 2500 agents, 50 cases, and with α set to 0.05. The system was allowed to run for 1000 iterations, a time considered enough to analyze the bees' behavior in performing the tasks randomly asked by different users.

Figure 2 shows the variations of the number of bees working in a demand in a part of the simulation. In the moment that some demand is performed (some travel package is recommended to the user), the bees abandon that demand and are idle until a new demand appears. The arrow indicates this situation in figure 2 (between time 30 and 40). When the demand C is performed, there is a considerable increase of bees to demand B. This demand is then quickly finished between time step 40 and 50.

It is important to notice that, after a demand has been performed and returned to the user, any user can request it again. This situation can be seen in figure 2 between time step 40 and 50, when demand C, after has been performed once, is requested by another user's query. During the recommendation process, bees are allocated to the demands at the moment they get into the system.

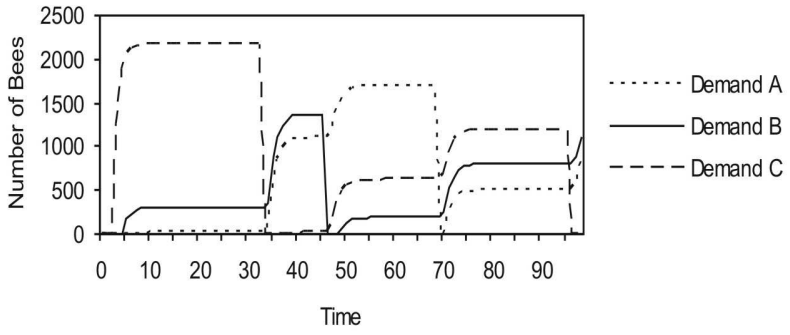

Figure 2: Number of bees versus time, for different demands

Figures 3 and 4 show the number of bees dealing with demand A. In figure 3, N is equal to 200 and in figure 4 N is equal to 2500. The arrows indicate the moments the recommendation was returned to the user. In figure 3, between the time steps 50 and 100, and also between 150 and 200, it is possible to see that Case 1 was the most danced case, i.e. the case that was recommended to the user, but the most similar was Case 0. This situation shows that the model behaves better (i.e. recommends the most similar case) with a big number of bees. However, this recommendation cannot be considered as a totally wrong recommendation because Case 1 is the second most similar.

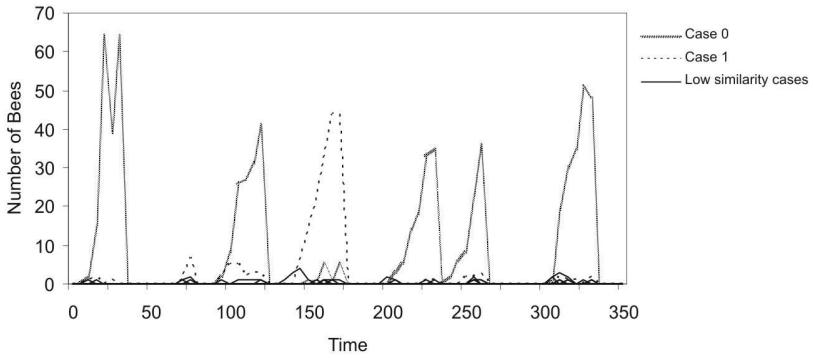

Figure 3: Number of bees dancing for given cases versus time (**N=200**; M=50)

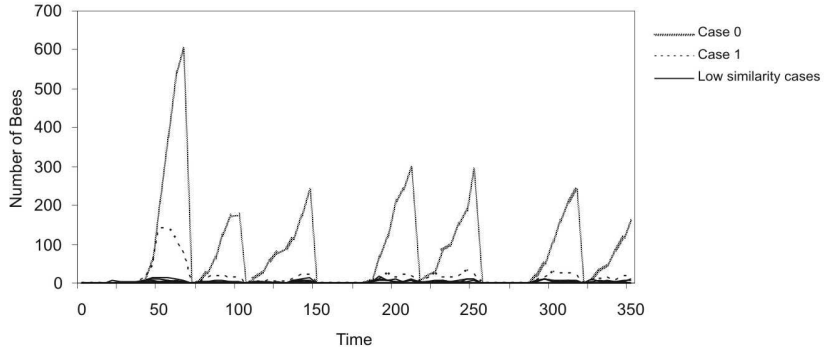

Figure 4: Number of bees dancing for given cases versus time (**N=2500**; M=50)

Figure 5 shows the number of cases visited by bees when they were working in the demand A, i.e. this figure shows how the tendency changes. In the beginning of the simulation, when the tendency of all cases was nearly equal, the bees have visited a high number of cases. However, within time, bees do not waste time in cases not relevant to the demands they are working in. It means that the bees focus on the more similar (appropriate) cases only.

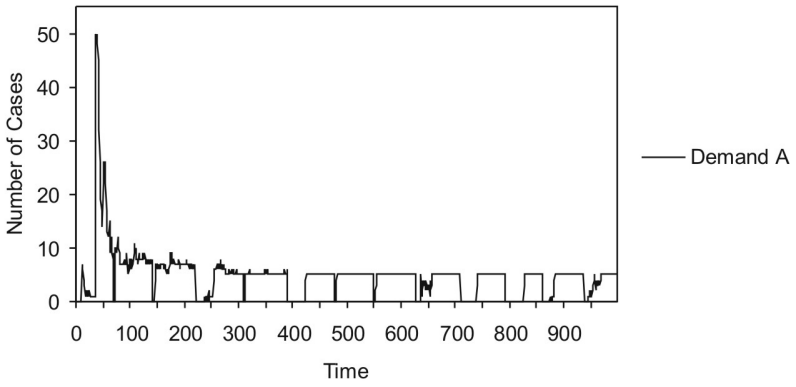

Figure 5: Number of cases that were visited by bees in the demand A

FUTURE TRENDS AND CONCLUSIONS

This paper presented an approach inspired by swarm intelligence applied to a case-based recommender system in the tourism domain. This is justified by the success such agents (social insect inspired) have in dynamic environments. The task of planning a travel needs dynamic information that is not always available to the travel agent. The use of agents combined to case-based recommender systems can help the travel agent in this task. Our experiments' results have shown that using this metaphor the system always return some recommendation to the user, avoiding the user's disappointment with the recommender system.

As future work, we intend to improve the case representation of CASIS, storing the previous queries as cases. Depending on the similarity of the new query, the bees can start not completely randomly. Instead, they may use the information gathered regarding the previous query.

Recommender systems are been used in e-commerce to help users to find out better products according to their needs and preferences. A new kind of recommender system is arising: distributed multiagent recommender system. In these systems the agents work together to search information that fits the users preferences. Each agent has its own knowledge base and all agents communicate with each other to exchange information only when they do not have it in their knowledge base.

REFERENCES

Aamodt, A. & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1), 39–59.

Bonabeau, E., Theraulaz, G. & Dorigo, M. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Univ. Press.

Bridge, D. & Ferguson, A. (2002). Diverse product recommendations using an expressive language for case retrieval. In S. Craw and A. Preece, (Ed.), *Advances in Case-Based Reasoning*,

Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR (pp. 43-57). Aberdeen, Scotland, 4 – 7 September. Springer Verlag.

Burke, R. (2000). Knowledge-based recommender systems. In J. E. Daily, A. Kent, and H. Lancour (Ed.), *Encyclopedia of Library and Information Science*, volume 69. Marcel Dekker.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G. & Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton.

Camazine, S. & Sneyd, J. (1991). A model of collective nectar source selection by honey bees: Self-organization through simple rules. *Journal of Theoretical Biology*, 149(4), 547–571.

Dorigo, M., & Caro, G. D. (1998). AntNet: Distributed Stigmergetic Control for Communications Network. *Journal of Artificial Intelligence Research, JAIR*, 9, 317-365.

Ferreira Jr., P.R., Oliveira, D. & Bazzan, A.L. (2005). A swarm based approach to adapt the structural dimension of agents' organizations. *Journal of Brazilian Computer Society, JBCS Special Issue on Agents Organizations*, 11(1), 63-73.

Fesenmaier, D, Ricci, F, Schaumlechner, E, Wober, K. & Zanella, C. (2003). DIETORECS: Travel advisory for multiple decision styles. In A. J. Frew, M. Hitz, and P. O'Connors (Ed.), *Information and Communication Technologies in Tourism* (pp. 232–241).

Gordon, D. (1996). The organization of work in social insect colonies. *Nature*, 380, 121–124.

Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA.

McSherry, D. (2003). Similarity and compromise. In A. Aamodt, D. Bridge, and K. Ashley (Ed.), *5th International Conference on Case-Based Reasoning*, ICCBR (pp. 291–305). Trondheim, Norway.

Nwana, H. S., Lee, L. C. & Jennings, N. R. (1996). Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4), 79–88.

Robinson, G. E. (1992). Regulation of Division of Labor in Insect Societies. *Annual Review of Entomology*, 37, 637-665.

Ricci, F. Venturini, A. Cavada, D. Mirzadeh, N Blaas, D. & Nones, M. (2003). Product recommendation with interactive query management and twofold similarity. In A. Aamodt, D. Bridge, and K. Ashley (Ed.), *5th International Conference on Case-Based Reasoning* (pp. 479–493). Trondheim, Norway.

Seeley, D. Camazine, S. & Sneyd, J. (1991). Collective decision-making in honey bees: how colonies choose nectar sources. *Behavioral Ecology Sociobiology*, 28, 277–290.

Watson, I. (1997). *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann.