AntRS: Recommending Lists through a Multi-Objective Ant Colony System

Pierre-Edouard Osche, Sylvain Castagnos, Anne Boyer

AntRS: Recommending Lists through a Multi-Objective Ant Colony System

Pierre-Edouard Osche, Sylvain Castagnos, and Anne Boyer

Univ. of Lorraine - CNRS - LORIA
Campus Scientifique B.P.239, Nancy, France
{pierre-edouard.osche,sylvain.castagnos,anne.boyer}@loria.fr

Abstract. When people use recommender systems, they generally expect coherent lists of items. Depending on the application domain, it can be a playlist of songs they are likely to enjoy in their favorite online music service, a set of educational resources to acquire new competencies through an intelligent tutoring system, or a sequence of exhibits to discover from an adaptive mobile museum guide. To make these lists coherent from the users' perspective, recommendations must find the best compromise between multiple objectives (best possible precision, need for diversity and novelty). We propose to achieve that goal through a multi-agent recommender system, called AntRS. We evaluated our approach with a music dataset with about 500 users and more than 13,000 sessions. The experiments show that we obtain good results as regards to precision, novelty and coverage in comparison with typical state-of-the-art single and multi-objective algorithms.

Keywords: Recommender systems · Multi-agent Systems · Multi-agent reinforcement learning

1 Introduction

Recommending an appropriate list or sequence of items to a specific user can be seen as a multi-objective problem. Let us illustrate this with a use case: Imagine a user who enjoys listening to music while doing sport through a mobile app. Such an online service should generate a playlist that is adapted to her preferences (*precision*). The tempo and the energy of the proposed songs should fit the context (*similarity*). The playlist should offer an appropriate level of *diversity* to avoid boredom. It could also bring *novelty* and *serendipity* according to her desires. The scientific challenge thus consists in taking into account different constraints that are contextualized and potentially not compatible.

In this paper, we propose a new multi-objective recommender system, called AntRS. Our model relies on a Multi-Agent System. The environment is a graph whose nodes are the items in the item set and whose edges connect items that have been co-consulted by several users. An Ant Colony Optimization algorithm allows to explore this environment until the target state is reached for each objective. Our model is generic since it is possible to add as many colonies as the

domain context requires. The paths generated by the different colonies are then merged to offer a good compromise between the objectives. We have validated our approach by choosing 4 objectives which can be antagonist (similarity vs. diversity, preferences vs. novelty). We relied on a music dataset made of 180,000 songs and 500 users, and compared our approach to 4 state-of-the-art algorithms. We measured the performances using several metrics (accuracy, novelty, diversity, coverage...). Results show that AntRS achieves a better accuracy than others, while offering a better compromise to users on other objectives.

This paper is organized as follows: Section 2 presents the related work on multi-objective recommenders and the principle of the Ant Colony Systems from which our system took inspiration. Section 3 describes our AntRS model. Sections 4 and 5 respectively describe the experiments carried out and the results obtained. Finally Section 6 concludes this paper and presents our perspectives.

2 Related Work

2.1 Multi-Objective Recommender Systems

A recommender system can either propose a list of independent items at each time step, or it can propose a sequence of items [20]. In Sequence-Aware Recommender Systems, one can both consider the importance of the order of the past events (by looking for co-occurrence patterns [4] or for sequential patterns [12] in past sessions) and the expected order of the future recommendations (e.g. continuation in playlists [13] or transitions between items [18]). In this paper, we based our experiment on a music dataset. As recent research has found little evidence that the exact order of songs actually matters to users [25], we limited our state-of-the-art to the recommendations of lists.

Transversely a recommender system can be mono-objective or multi-objective. Most recommenders solely focus on the accuracy (precision and recall) [23]. Others attempt to find a compromise between precision, serendipity and novelty [15], or between precision and diversity [32, 17] for example. There are several ways to address a multi-objective optimization problem. One can either look for a set of Pareto solutions, considering that a solution is optimal if it is not possible to make any objective better off, without making at least one objective worse off [33]. In that case, recommender systems aim at producing as many solutions as possible in order to cover as much as possible of the problem's Pareto front. Or one can rank the items in a single list by aggregating or reordering the results of each single objective [21]. This list can be produced in one stage [10], or come out of a 2-step process consisting in generating several lists of candidate items for the active user and in merging them [9, 27–29, 22, 8]. Recommending several Pareto solutions offers the advantage to leave the choice to the active user. It can be interesting in some application domains such as e-commerce where an explicit validation process from the user is mandatory. However, in the context of online music services, it is not conceivable to request a user decision at each timestep. The songs must come one after another without disturbing the user in his/her

main current task. For this reason, we focused this paper on recommenders which produce only one solution (i.e. only one list of recommendations).

The existing multi-objective single-list recommenders suffer from several limitations: they are dependent from the application domain (any change in the set of objectives has a drastic impact on the implementation of the model) and they are very time-consuming. To bypass these difficulties, we propose a new approach relying on a Multi-Agent System (MAS), and more precisely on an Ant Colony System explained below. MAS have multiple advantages in our context:

- they have a relatively low computational complexity;
- they are efficient at tackling multi-objective problems [1, 3];
- they can easily be adapted to new configurations and are resilient to changes.

2.2 Ant Colony Systems

The Ant system algorithm (AS) is inspired by the foraging behavior of ants, specifically the pheromone communication between ants, to find shortest paths in an environment between a starting node and a target node. Dorigo proposed a few different versions of this AS model [7]. Our model took inspiration from one of those variants, the Ant Colony System (ACS) algorithm. In comparison to the classic AS model, ACS proposes a different way for the ants to deposit pheromones. Instead of having all the ants deposit their pheromones at the end of one iteration (i.e. after all the ants have finished their tour), only the ant that found the best path can deposit pheromones. Furthermore, ants perform a so called local pheromone update where, after each construction step, they deposit some pheromone on the last edge they visited. In other words, each time an ant takes an edge, it deposits some pheromones along its way, regardless of the quality of the path. As explained by Dorigo, this version of the ACO algorithms is known to "*diversify the search performed by subsequent ants during an iteration: by decreasing the pheromone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and, hence, to produce different solutions. This makes it less likely that several ants produce identical solutions during one iteration*". As our search space is large (there are millions of songs on an online music service) and as we promote not only the precision but other characteristics in the recommended lists, we chose the ACS algorithm. In the rest of this subsection, we explain the main formulas of the ACS.

State transition rule - The state transition rule uses the pseudo-random proportional rule where a random variable $q \in [0; 1]$ is compared to a parameter q_0 to decide if the ant will explore the graph or if it will exploit the knowledge collected by previous ants. $q_0 = 0$ is equivalent to the AS model where ants only explore the graph while $q_0 = 1$ refers to a pure reinforcement behavior with no exploration. The Equation 1 let the algorithm decide between knowledge exploitation and biased exploration of the graph.

$$\begin{cases} \text{Exploitation (Equation 2) if } q \leq q_0 \\ \text{Biased exploration (Equation 3) if } q \geq q_0 \end{cases} \tag{1}$$

The Equation 2 represents the direct exploitation of the knowledge in the graph where the best edge is always chosen.

$$\arg\max_{l \in V_i} \{\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}\}, \tag{2}$$

where V_i is the set of available nodes from the node i, $\tau_{il} \in [0;1]$ is the amount of pheromones left on an edge (i,l) by previous ants, η_{il} is the heuristic information on an edge (i,l), α and β are two parameters representing respectively the weight of the pheromones and the weight of the heuristic.

The Equation 3 represents the biased exploration where best edges have more chances to be picked and p_{ij} is the probability for an ant at the node i to choose the edge (i,j).

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum\limits_{l \in V_i} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, & \text{if } j \in V_i, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

Global pheromone update - After each iteration, only the ant who found the best tour is allowed to update the pheromone level τ_{ij}:

$$\tau_{ij} = \begin{cases} (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to best tour,} \\ \tau_{ij} & \text{otherwise,} \end{cases} \tag{4}$$

where ρ is the evaporation rate of the pheromones and $\Delta\tau_{ij} = 1/L_{best}$ where L_{best} is the length of the best tour.

Local pheromone update - Another addition of the ACS model over the AS model is the local pheromone update performed after each step by each ant described in Equation 5.

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \tag{5}$$

where τ_0 is the pheromone level set on every edge at the initialization.

Heuristic information - the heuristic information η_{ij} represents the information that ants possess *a priori* on an edge (i,j). In ACS, the heuristic information is computed based on the distance between the two nodes of the edge: the farther both nodes are from each other and the lower η_{ij} will be.

$$\eta_{ij} = \frac{1}{d_{ij}}, \text{where } d_{ij} \text{ is the distance between nodes } i \text{ and } j. \tag{6}$$

3 Our model: AntRS

As previously stated, AntRS has been built with several goals in mind: (1) be as generic as possible; (2) be able to include several competing objectives in a single list; (3) be resilient to changes in the environment (new items, new preferences,...). Our model takes inspiration from the ACS algorithm because the

latter gathers all the quality needed to satisfy those objectives. However, we want to point out the differences between the classic ACS as described in Section 2.2 and our model AntRS. First of all, we had to develop our own method to create a graph to model as best as possible the large environment we were working in without hindering the execution time of the system. In Subsection 3.1 we present our graph creation method. Secondly, we wanted to optimize many objectives while the ACS optimizes only one attribute which is usually the distance. In Subsection 3.2 we introduce more formally the objectives used in our model. Thirdly, as we are generating several paths during the algorithm execution, a merging procedure has to be executed at one point to be able to propose the best possible recommendation list for each user. The Subsection 3.3 explains two tactics we used to do so.

3.1 Graph creation

The first step of our model is the creation of the graph. This is often an overlooked part in the literature as the datasets used are usually small and/or the links between nodes of the graph are manually picked by a field expert. One of the main differences between ACO simulations and real ants is the definition of the search space. Real ants are evolving in a continuous search space without any landmarks (or vertices) and are free to go everywhere whereas agents are released in a discrete environment and have to follow predetermined paths (or edges) between set landmarks. One of the ways to be as close as possible to real ants' behavior would be to compute distances between each and every node of the graph to build a complete graph. It is nonetheless an unpractical solution for more than a few thousands vertices as the number of edges depends on the number of vertices n with $|E| = \frac{n(n-1)}{2}$. As our goal is to use our model in a realistic situation with many potential items represented by vertices, we decided to find a workaround without sacrificing the quality in the solutions found. To do so, we needed to select a few "best" edges between each vertex. At this point, we formulated two hypotheses to help us construct the graph: (1) past sequences created by previous users represent useful domain knowledge which should be exploited; (2) past sequences done by previous users are not always the best possible ones and could have been improved with clever recommendations.

To take into account those two hypotheses, we first computed the number of transitions (i.e. co-consultations) between each pair of items in our dataset and we added (1) all the transitions above a specific threshold, and (2) only some of those below this threshold as edges. Finally, if a given connectivity degree was not reached, we added new edges between items who were not connected in our dataset to allow our model to discover new potential interesting paths not known by users. The process of creating an edge is shown in Equation 7.

$$e_{ij} = \begin{cases} \text{if } t_{ij} \geq m \\ \text{or if } t_{ij} < m \text{ and } q < t_{ij} \text{ where } q \in [min\ t_{il}; \log(max\ t_{il})] \\ \text{or if } t_{ij} = 0 \text{ and if } deg(i) < d \text{ then pick a random} \\ \text{transition until } deg(i) = d \end{cases} \tag{7}$$

where e_{ij} is the presence of an edge between vertices i and j, t_{ij} is the number of transitions performed from item i to item j by the users in the dataset, m is the threshold where transitions are not directly added to the graph as edges, $q \in [min\ t_{il}; \log(max\ t_{il})]$ is a random variable uniformly distributed, $min\ t_{il}$ is the minimal number of transitions between the item i and all the others items $l \in V_i$ where at least one transition has been found, $deg(i)$ is the current degree of the node i in the graph and d is a parameter specifying the minimal degree each vertex must have in the final graph.

3.2 Objectives

It is now widely admitted that the sole precision is not sufficient to produce good recommendations to users. We thus propose to define a set of 4 concurrent objectives that have to often be considered in the literature while recommending a list of items. The objectives we considered are all transposable in different application domains, guaranteeing the genericity of our approach.

Furthermore, the ability to add, to modulate the importance or to remove objectives on the fly was essential for having an adaptive model. To address this issue, we chose to integer as many colonies as objectives in our model, and each colony is specialized in maximizing its own objective. To do so, we modified the way the ACS model computes the distance d between two nodes of the graph while the calculation of the heuristic η_{ij} was left untouched. The rest of this Section describes the equations used to compute the distance for each colony.

Similarity - This is one of the main factor considered by nearly all the recommender systems. The main goal of a recommender system is to propose items similar to what the user liked before. Even if similarity is a well-known and widely used characteristic, we think that a good recommender system cannot overlook it. We also do consider that similarity should not be the cornerstone of each and every recommender system anymore. The goal of this colony is to find a list with items as similar as possible of what the user previously viewed or is currently viewing. A lot of methods exist to compute the similarity of two vectors and, based on our dataset and on the metadata available, we decided to use a cosine similarity measure [24]. To compute the distance value on the edges of the graph, we simply computed the cosine similarity between the two items represented by the vertices. More formally, for an edge (i, j), its associated distance d_{ij} is computed with the cosine similarity between the vectors of the descriptive characteristics of the items i and j.

$$d_{ij} = \frac{1}{sim(C_i, C_j)} \tag{8}$$

where C_i are the characteristics of the item i. The item characteristics depend obviously on the dataset and on the meta-information available but, we can formalize that each item of the dataset is described by n characteristics as follow $C_i = \{c_1, c_2, \ldots, c_n\}$. We used the multiplicative inverse to transform the similarity metric $sim \in [0; 1]$ into a distance $d \in [1; +\infty]$. Therefore, a distance value near 1 on an edge (i, j) means that the two items i and j are similar.

Diversity - This characteristic and the similarity are often described together as they are both related to the distance/correlation between the items liked by the user and his/her recommendations. But unlike similarity, diversity depicts how dissimilar two items are relatively to each other. Similarity and diversity are complementing each other in the sense that they are both needed to adapt the system to the needs of different users [14]. To compute this objective, we chose to apply one of the classic diversity metric which is obtained by computing the inverse of the similarity between two items, as shown in Equation 9. As for the similarity, we used the multiplicative inverse of the diversity to obtain a distance $d \in [1; +\infty]$.

$$d_{ij} = \frac{1}{1 - sim(C_i, C_j)} \tag{9}$$

Novelty - This characteristic represents the items that are not yet known by the user. It could be new items recently added to the system or old but not so popular items that the user missed. Novelty should not be confused with diversity, since novel items could be either similar or dissimilar to what the user usually likes. Novelty is an important characteristic of a recommender system to avoid a potential lack of interest of users due to too much foreseeability in the recommended items [26].

To determine if an item is novel or not relatively to a specific user, we used the work of Zhang [31] who defined the novelty as a notion composed of three characteristics: (1) Unknown: the item is unknown to the user; (2) Satisfactory: the item is liked by the user; (3) Dissimilarity: the item is dissimilar to the other items known by the user. The author proposed to evaluate the novelty of the item i for the user u as follow:

$$novelty(i, u) = p(i|unknown, u) \cdot dis(i, pref_u) \cdot p(i|like, u) \tag{10}$$

where $p(i|unknown, u)$ is the probability that the user u does not know the item i, $dis(i, pref_u)$ is the dissimilarity between i and the set of items in the users' profile and $p(i|like, u)$ is the probability that u will like i. However, the dissimilarity and the satisfaction of the user relatively to i are closely related to other objectives in our model, respectively maximized by the diversity colony and by both the preferences and the similarity colonies. Hence we decided to trim down the Equation 10 to the probability $p(i|unknown, u)$ only (see Equation 11).

$$p(i|unknown, u) = -log(1 - pop_i), \text{ where } pop_i \text{ is the popularity of item } i. \tag{11}$$

Preferences - The preferences characteristic corresponds to what the user really likes. It intersects with the similarity notion but, again as with diversity and novelty, we think that preferences express another aspect of a good recommendation for a user. The similarity characteristic allows the recommender to propose items that are similar to the preferences of the user, but it is not guaranteed that he will like those items. It is for example perfectly common to both like and dislike some songs coming from the same album and artist, yet those songs will probably be treated as very similar relative to each other. The preferences characteristic favors items that are known to be liked by the user.

The goal for this colony is to find a sequence in the graph prioritizing items that are already known to be liked by the user. Thereby, items must have criteria conveying how the user like an item or not. This can be done either with explicit feedback (*e.g.* item rating,...), with implicit feedback (*e.g.* number of times the user viewed an item,...) or with a combination of both. The nature of the feedback will heavily depends on the domain, but we can formalize that each collected information concerning the behavior of a user on an item must be taken into account. Let C_u be the set of criteria representing all the actions that a user u may perform on the items of the system, thus $c_{u,i}$ is the sum of all interactions specific to a single criterion c that a user u performed on an item i (*e.g.* the number of times a user u viewed i). To aggregate all the different interactions possible in a single value, we use the presumed interest formula proposed by Castagnos et al. in [6] and described in the Equation 12.

$$presumed\ interest_{u,i} = v_{min} + \frac{\sum_{c \in C} (w(c) \cdot c(u,i))}{\sum_{c \in C} w(c)} \cdot \frac{(v_{max} - v_{min})}{c_{max}} \tag{12}$$

where $c(u,i)$ corresponds to normalized values given to the item i by the user u to each criterion c, $w(c)$ is the weight of the criterion c, v_{min} and v_{max} are the minimal and maximal expected values for the presumed interest and c_{max} is the maximal value that $c(u,i)$ can take regardless of the criteria. In our case, we considered the following criteria for each song: number of consultations, number of skips, number of bans (when the user do not want to listen to the song ever again) and number of likes.

3.3 Merging tactics

In the previous section, we described four objectives that could provide suitable recommendations for users. Each of these four objectives is associated to a specific ant colony in our model. Thus, after this step, we are left with as many lists of recommendations as colonies, where each one should represent a part of the final recommended list. In order to build it, we needed a tactic to merge all the colonies' lists into one. To do so, we propose two techniques described below.

Merging colony - The first merging tactic relies once more on the ACS algorithm but with one additional colony that we called "merging colony". Starting from the set of items found by the other colonies (**step 1**), the merging colony considers all the objectives at once with a weighted sum to calculate the distances on the graph's edges (**step 2**), as shown in Equation 13.

$$d_{ij} = \sum_{col \in colonies} w(col) \cdot d_{ij}(col) \tag{13}$$

where $w(col)$ is the weight representing the expected importance of the colony's objective in the final recommendation. To estimate those weights, we calculated the average values of each objective (similarity, diversity, novelty and preferences) on the last n sessions processed of the user. This gave us the general

importance of each objective while taking into account contextual information and recent tendencies in the user's behavior. We also built a new graph for this path of the algorithm. To construct it, we used all the items in the lists found by the other colonies as vertices, we added edges to each consecutive pair of items in the lists and finally we added random edges in the same way that is described in the last part of the Equation 7 to give the possibility of new paths to be found and chosen by the merging colony's agents.

Lists merging - For the second merging tactic, we calculated the weight $w(col)$ of each objective in the same way that for the merging colony (see above). We then built the list step by step by considering all the items found by the different colonies. We iterated through all the available items for each step of the list construction and we added to the final recommended list the item which yield the best amelioration towards the expected values. This process was stopped either when the remaining items degraded the list's metrics, when there was no items left or when the last item of the initial listened session was found.

4 Experimentation

4.1 Comparison algorithms

We compared the performances of AntRS with four state-of-the-art algorithms capable of producing lists of items in the same conditions than our model. The first three are classical techniques spanning most of the work in the recommender systems domain: (1) UserKNN [5]; (2) TrustMF [30] and (3) SVD++ [16]. Those three algorithms were implemented using the Java library librec [11]. We also implemented a fourth hybrid multi-objective model named PEH described in [22] to be able to compare AntRS to a state-of-the-art multi-objective recommender system. We used the three algorithms described above in the hybridization process. Furthermore, we also ran several version of AntRS to assess the strenghts and the weaknesses of our model.

4.2 Dataset

For our experimentations, we decided to use a dataset from Deezer as they offer the possibility to get metadata and information on listened tracks with their API. Our dataset spans one month of listenings starting from 5th Dec. 2016.

We split the dataset in listening sessions which corresponded to a listening with a break not longer than $900+track\_duration$ seconds. Among all 1,871,919 consultations, we were able to determine 91,468 unique sessions with a mean length of 18.3 tracks each. The full dataset contains 3,561 unique users, 178,910 unique songs, and 1,871,919 listenings. However, as the PEH algorithm is not highly scalable (similarly to most of multi-objective algorithms) and so as to compare all algorithms in the same conditions, we limited the experiment to 500 randomly chosen users.

Each track is described by a number of metadata provided by Deezer, including song name, artist name, album name, music genre, related artists and

some numerical values like acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo and valence. We used those characteristics to implement the metrics of the four different colonies described in Section 3.2.

Finally, so as to transform consultations (more precisely metadata such as duration, frequency or recency of consultations) into ratings usable by collaborative filtering algorithms (UserKNN, TrustMF, SVD++), we used the Formula 12 proposed by [6]. We recall that this same formula was also used by our model for the objective of preferences, taking no advantage on other algorithms.

4.3 Experimental protocol

We performed several evaluations of our model in diverse configurations to measure its performance relatively to itself and to other models. To guarantee a fair chance for each model and configuration, we set the same starting and stopping conditions and we used the same data for all the experiments we did. For each listening session of the test base users, each algorithm produced one recommended list and its performance was measured in comparison to the initial listening session. The first item of each session was given to the algorithm as starting point for the recommended list. The last item was not given but, if the algorithm reached it during the recommendation process then it was stopped. A minimum size was set for the recommended list which was half of the size of the initial listened session. After the recommendation, the initial listened session was added to the training base to simulate a real-case scenario where a system first has no information on a new user and then gather more and more data on him as he interacts with the system. Finally, all the tests were performed on a cross-validation dataset with a training base of 400 users and a test base of 100 users each time. For each listening session composed of 5 items or more in the test base, a recommended session was produced. The users of the training base had listened to 10,621 sessions while there were 2,569 sessions in the test base.

We used different metrics capturing all the aspects of what we consider a good recommendation: Precision, Recall and F-measure [2], Similarity and Intra-list Similarity [32], Diversity and Relative diversity [19, 26]. We also used the preferences and the novelty metrics of Equations 11 and 12 as well as in [26]. We empirically fixed the meta-parameters values of the baseline algorithm described in 2.2 as follows: $q_0 = 0.3$, $\alpha = 0.1$, $\beta = 0.9$, $\rho_0 = 0.2$, $\tau_0 = 0.1$.

5 Results

5.1 Single-objective AntRS

We first wanted to see how each of our objective performed alone. As our model allows us to change the number of objectives on the fly, we performed four different tests for the four different objectives without any merging step and we measured how each of the tests performed considering the metrics described above. The Table 1 presents those results. We measured the statistical significance of the score of the colony that was supposed to perform best on each

Table 1. Experimentations with AntRS as a single-objective model

	Precision	Recall	F-measure	Similarity	ILS	Diversity	RD	Novelty	Preferences
Similarity colony	0.317	0.126	0.165	**0.952**	**0.923**	0.048 ***	0.049 ***	0.72 ***	0.447 ***
Diversity colony	0.211	0.104	0.127	0.862 ***	0.77 ***	**0.138**	**0.19**	0.806 ***	0.393 ***
Novelty colony	0.132	0.112	0.114	0.895 ***	0.837 ***	0.105 ***	0.144 ***	**0.909**	0.379 ***
Preferences colony	0.296	0.159	0.191	0.946 ***	0.91 ***	0.054 ***	0.08 ***	0.695 ***	**0.804**

Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

metric in comparison with the results obtained by the other colonies. Thus, the similarity and the intra-list similarity of the similarity colony were compared to the similarity and intra-list similarity of the 3 others colonies and so on. As the Shapiro-Wilk test revealed that our data did not follow a normal distribution, we used the non-parametric Wilcoxon test which allowed us to compare the means of two related samples (same users with different algorithms). As expected, we can see that each of our colonies produce lists that are specialized in a single objective. Thus, the similarity colony produces lists that are the most similar relatively to all the other colonies; the novelty colony produces lists that are the most novel, and so on. Those experiments showed that our model was working as intended and that we could combine the four objectives together.

5.2 Multi-objective AntRS

In Table 2, we present the summary of the results obtained for all the models tested and their variations. We also measured the statistical significance of the results of our best performing model, AntRS with the lists merging, in comparison with the results of all the other models. As for the previous subsection, the statistical test used was Wilcoxon signed-rank test.

As explained in Section 3.3, we proposed and tested two merging tactics to combine the results of our four objectives. We also tested to run the merging colony alone, without the **step 1** in the first merging tactic of Section 3.3: in that scenario, the merging colony operates on the whole graph, rather than on the subgraph of items recommended by the four colonies. We hypothesize that running directly the merging colony without the **step 1** will degrade the quality of the final solutions found. This first step with the four colonies gave our model the ability to find very specialized lists of items, which are associated to optimal solutions in a Pareto front, and this process was supposed to help the merging colony to find a better compromise between those solutions. This hypothesis has been confirmed in Table 2 since the two variants of AntRS with the 4 colonies got better results than the sole merging colony on each metric, except for the similarity, thus offering a better deal. The first merging tactic of our model outperforms the second one in terms of precision, but the latter obtains the best relative diversity of all the AntRS variants.

We can also see that both AntRS variations obtain the best precision (up to +78.23%), recall (up to +29.05%) and F-measure (up to +31.28%) of all the models tested, which means that our model was the best to capture the preferences from the lists initially listened by the users in the training set. AntRS

Table 2. Experimentations with AntRS as a multi-objective model

	Precision	Recall	F-measure	Similarity	ILS	Diversity	RD	Novelty	Preferences	Coverage
AntRS (4 colonies + lists merging)	**0.344**	0.131	**0.1838**	0.947	0.908	0.053	0.069	**0.741**	**0.61**	**96.92%**
AntRS (4 colonies + merging colony)	**0.288** ***	**0.151** ***	**0.1836** **	0.945	0.905 ***	0.055	**0.082** ***	0.702 ***	**0.612** **	**96.92%**
AntRS (merging colony alone)	0.197 ***	0.118 ***	0.141 ***	0.953 ***	0.93 ***	0.047 ***	0.068	0.324 ***	0.389 ***	96.92%
UserKNN	0.224 ***	0.138	0.162 ***	0.95 ***	0.905 ***	0.05 ***	0.081 ***	0.68 ***	0.541	61.39%
TrustMF	0.195 ***	0.117 ***	0.14 ***	0.95 ***	0.894 ***	0.058 ***	0.09 ***	0.697 ***	0.458 ***	68.17%
SVD++	0.195 ***	0.12 ***	0.14 ***	0.941 ***	0.892 ***	0.06 ***	0.093 ***	0.70 ***	0.455 ***	68.17%
PEH	0.193 ***	0.118 ***	0.14 ***	0.941 ***	0.892 ***	0.059 ***	0.096 ***	0.697 ***	0.452 ***	97,52%

Significance codes (compared to AntRS with 4 colonies + lists merging): 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

also outperformed the other models for the preferences and the novelty metrics, while still managing to maintain a correct level of similarity and diversity. Let us remind that we deliberately chose non-compatible objectives (Similarity vs. Diversity, Novelty vs. Preferences), which makes the task harder for the multi-objective algorithms (AntRS and PEH) compared to others. Despite a lower diversity, AntRS offers a better compromise between all the objectives than PEH, and in a much shorter execution time. Within the frame of this experiment, we considered that all the objectives had an equal importance. However, it would be easy to weight the different objectives in AntRS according to user expectations, like they did in [8]. Finally, we can notice that AntRS and PEH got a much better coverage compared to other algorithms.

6 Conclusion and Perspectives

In this paper, we showed that our model, AntRS, is able to generate lists with a higher precision than other methods, while still offering a good compromise between similarity, diversity, novelty and preferences. AntRS offers many advantages compared to the state-of-the-art models: (1) the multi-agents part of our model guarantees that it is highly adaptable to changes in the environment, (2) the objective-oriented colonies can be added or removed on the fly, (3) it is generic enough to be adapted in all the domains where a list recommender is relevant, and (4) it is highly parallelizable and resilient to the cold-start problem.

We have some interesting ideas on how to pursue our work in the future. First, we would like to improve the quality of the recommended lists by personalizing the construction of the graph. At the moment, a unique graph is created for all the users while the distances on the edges are recalculated for each user. We would like to improve the personalization by creating a unique graph for each user, and even a unique graph for each colony.

Secondly, we would like to work on the notion of sequence. Instead of recommending simple lists to user, we think that offering a coherent sequence of items with a start, an end and a good progressivity could be beneficial in other domains (a path in a museum, a sequence of courses for a student...). This could be achieved by adding a colony dedicated to the progressivity.

References

1. Ariyasingha, I., Fernando, T.: Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem. Swarm and Evolutionary Computation **23**, 11–26 (2015)
2. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval, vol. 463. ACM press New York (1999)
3. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: Applied informatics. pp. 97–102 (2003)
4. Bonnin, G., Jannach, D.: Automated generation of music playlists: Survey and experiments. ACM Comput. Surv. **47**(2), 26:1–26:35 (2014)
5. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. pp. 43–52. Morgan Kaufmann Publishers Inc. (1998)
6. Castagnos, S., Boyer, A.: A client/server user-based collaborative filtering algorithm: Model and implementation. In: 17th European Conference on Artificial Intelligence (ECAI 2006). pp. 617–621 (2006)
7. Dorigo, M., Birattari, M.: Ant colony optimization. In: Encyclopedia of machine learning, pp. 36–39. Springer (2011)
8. Fortes, R.S., Lacerda, A., Freitas, A., Bruckner, C., Coelho, D., Gonçalves, M.: User-oriented objective prioritization for meta-featured multi-objective recommender systems. In: Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization. pp. 311–316. ACM (2018)
9. Geng, B., Li, L., Jiao, L., Gong, M., Cai, Q., Wu, Y.: Nnia-rs: A multi-objective optimization based recommender system. Physica A: Statistical Mechanics and its Applications **424**, 383–397 (2015)
10. Guimarães, A., Costa, T.F., Lacerda, A., Pappa, G.L., Ziviani, N.: Guard: A genetic unified approach for recommendation. Journal of Information and Data Management **4**("), 295 (2013)
11. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: Librec: A java library for recommender systems. In: UMAP Workshops (2015)
12. Hariri, N., Mobasher, B., Burke, R.: Context-aware music recommendation based on latenttopic sequential patterns. In: Proceedings of the Sixth ACM Conference on Recommender Systems. pp. 131–138. RecSys '12 (2012)
13. Jannach, D., Lerche, L., Kamehkhosh, I.: Beyond hitting the hits: Generating coherent music playlist continuations with the right tracks. In: Proceedings of the 9th ACM Conference on Recommender Systems. pp. 187–194. ACM (2015)
14. Jones, N.: User Perceived Qualities and Acceptance of Recommender Systems: The Role of Diversity. Ph.D. thesis, EPFL (2010)
15. Kaminskas, M., Bridge, D.: Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Trans. Interact. Intell. Syst. **7**(1), 2:1–2:42 (2016)
16. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 426–434. ACM (2008)
17. L'Huillier, A., Castagnos, S., Boyer, A.: Understanding usages by modeling diversity over time. In: 22nd Conference on User Modeling, Adaptation, and Personalization. vol. 1181 (2014)

18. Maillet, F., Eck, D., Desjardins, G., Lamere, P.: Steerable playlist generation by learning song similarity from radio station playlists. In: In Proceedings of the 10th International Conference on Music Information Retrieval (2009)
19. McGinty, L., Smyth, B.: On the role of diversity in conversational recommender systems. In: Proceedings of the 5th International Conference on Case-based Reasoning: Research and Development. pp. 276–290. ICCBR'03 (2003)
20. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. CoRR **abs/1802.08452** (2018)
21. Ribeiro, M.T., Lacerda, A., Veloso, A., Ziviani, N.: Pareto-efficient hybridization for multi-objective recommender systems. In: Proceedings of the Sixth ACM Conference on Recommender Systems. pp. 19–26. RecSys '12 (2012)
22. Ribeiro, M.T., Ziviani, N., Moura, E.S.D., Hata, I., Lacerda, A., Veloso, A.: Multiobjective pareto-efficient approaches for recommender systems. ACM Trans. Intell. Syst. Technol. **5**(4), 53:1–53:20 (2014)
23. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook. Springer, US (2015)
24. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence **2009**, 4 (2009)
25. Tintarev, N., Lofi, C., Liem, C.C.: Sequences of diverse song recommendations: An exploratory study in a commercial system. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. pp. 391–392. UMAP '17 (2017)
26. Vargas, S., Castells, P.: Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the fifth ACM conference on Recommender systems. pp. 109–116. ACM (2011)
27. Wang, S., Gong, M., Ma, L., Cai, Q., Jiao, L.: Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 672–679 (2014)
28. Wang, S., Gong, M., Li, H., Yang, J.: Multi-objective optimization for long tail recommendation. Knowledge-Based Systems **104**, 145 – 155 (2016)
29. Xia, X., Wang, X., Li, J., Zhou, X.: Multi-objective mobile app recommendation: A system-level collaboration approach. Comput. Electr. Eng. **40**(1), 203–215 (2014)
30. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence **39**(8), 1633–1647 (2017)
31. Zhang, L.: The definition of novelty in recommendation system. Journal of Engineering Science & Technology Review **6**(3) (2013)
32. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web. pp. 22–32. ACM (2005)
33. Zuo, Y., Gong, M., Zeng, J., Ma, L., Jiao, L.: Personalized recommendation based on evolutionary multi-objective optimization [research frontier]. IEEE Computational Intelligence Magazine **10**(1), 52–62 (2015)