# MULTI-OBJECTIVE PARETO-EFFICIENT ALGORITHMS FOR RECOMMENDER SYSTEMS

MARCO TÚLIO CORREIA RIBEIRO

# MULTI-OBJECTIVE PARETO-EFFICIENT ALGORITHMS FOR RECOMMENDER SYSTEMS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Nívio Ziviani
Co-Orientador: Adriano Veloso

Belo Horizonte
Abril de 2013

MARCO TÚLIO CORREIA RIBEIRO

# MULTI-OBJECTIVE PARETO-EFFICIENT ALGORITHMS FOR RECOMMENDER SYSTEMS

Dissertation presented to the Graduate Program in Ciência da Computação of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Ciência da Computação.

Advisor: Nívio Ziviani
Co-Advisor: Adriano Veloso

Belo Horizonte
April 2013

Ribeiro, Marco Túlio Correia
R484m Multi-Objective Pareto-Efficient Algorithms for Recommender Systems / Marco Túlio Correia Ribeiro. — Belo Horizonte, 2013
xviii, 41 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.
Orientador: Nívio Ziviani

1. Computação - Teses. 2. Sistemas de Recomendação
- Teses. 3. Recuperação de Informação - Teses.
I. Orientador. II. Título.

CDU 519.6*73 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# FOLHA DE APROVAÇÃO

Multi-objective pareto-efficient approaches for recommender systems

**MARCO TÚLIO CORREIA RIBEIRO**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. NIVIO ZIVIANI - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. BERTHIER RIBEIRO DE ARAÚJO NETO
Departamento de Ciência da Computação - UFMG

PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 23 de abril de 2013.

# Acknowledgments

Agradeço a Deus, por sustentar toda a minha vida. Agradeço aos meus pais e à minha noiva pelo suporte.

Quanto ao trabalho em si, agradeço aos colegas Itamar Hata e Anísio Lacerda, pela grande ajuda oferecida. Agradeço por fim aos orientadores, Nívio Ziviani e Adriano Veloso.

*"O conhecimento traz orgulho, mas o amor edifica."*
(Paulo, 1Co 8:1)

# Resumo

Sistemas de recomendação tem se tornado cada vez mais populares em aplicações como e-commerce, mídias sociais e provedores de conteúdo. Esses sistemas agem como mecanismos para lidar com o problema da sobrecarga de informação. Uma tarefa comum em sistemas de recomendação é a de ordenar um conjunto de itens, de forma que os itens no topo da lista sejam de interesse para os usuários. O conceito de interesse pode ser medido observando a acurácia, novidade e diversidade dos itens sugeridos. Geralmente, o objetivo de um sistema de recomendação é gerar listas ordenadas de forma a otimizar uma dessas métricas. Um problema mais difícil é tentar otimizar as três métricas (ou objetivos) simultaneamente, o que pode levar ao caso onde a tentativa de melhorar em uma das métricas pode piorar o resultado nas outras métricas. Neste trabalho, propomos novas abordagens para sistemas de recomendaççao multi-objetivo, baseadas no conceito de Eficiência de Pareto − um estado obtido quando o sistema é de tal forma que não há como melhorar em algum objetivo sem piorar em outro objetivo. Dado que os algoritmos de recomendação existentes diferem em termos de acurácia, diversidade e novidade, exploramos o conceito de Eficiência de Pareto de duas formas distintas: (i) agregando listas ordenadas produzidas por algoritmos existentes de forma a obter uma lista única - abordagem que chamamos de ranking Pareto-eficiente, e (ii), a combinação linear ponderada de algoritmos existentes, resultado em um híbrido, abordagem que chamamos de hibridização Pareto-eficiente. Nossa avaliação envolve duas aplicações reais: recomendação de música com *feedback* implícito (i.e., Last.fm) e recomendação de filmes com *feedback* explícito (i.e., Movielens). Nós mostramos que as abordagens Pareto-eficientes são efetivas em recomendar items com bons niveis de acurácia, novidade e diversidade (simultaneamente), ou uma das métricas sem piorar as outras. Além disso, para a hibridização Pareto-eficiente, provemos uma forma de ajustar o compromisso entre acurácia, novidade e diversidade, de forma que a ênfase da recomendação possa ser ajustada dinamicamente para usuários diferentes.

**Palavras-chave:** Sistemas de recomendação híbridos, Eficiência de Pareto, Diversidade, Novidade.

# Abstract

Recommender systems are quickly becoming ubiquitous in applications such as e-commerce, social media channels and content providers, acting as enabling mechanisms designed to overcome the information overload problem by improving browsing and consumption experience. A typical task in recommender systems is to output a ranked list of items, so that items placed higher in the rank are more likely to be interesting to the users. Interestingness measures include how accurate, novel and diverse the suggested items are, and the objective is usually to produce ranked lists optimizing one of these measures. Suggesting items that are simultaneously accurate, novel and diverse is much more challenging, since this may lead to a conflicting-objective problem, in which the attempt to improve a measure further may result in worsening other measures. In this thesis we propose new approaches for multi-objective recommender systems based on the concept of Pareto-efficiency − a state achieved when the system is devised in the most efficient manner in the sense that there is no way to improve one of the objectives without making any other objective worse off. Given that existing recommendation algorithms differ in their level of accuracy, diversity and novelty, we exploit the Pareto-efficiency concept in two distinct manners: (i) the aggregation of ranked lists produced by existing algorithms into a single one, which we call Pareto-efficient ranking, and (ii) the weighted combination of existing algorithms resulting in a hybrid one, which we call Pareto-efficient hybridization. Our evaluation involves two real application scenarios: music recommendation with implicit feedback (i.e., Last.fm) and movie recommendation with explicit feedback (i.e., MovieLens). We show that the proposed approaches are effective in optimizing each of the metrics without hurting the others, or optimizing all three simultaneously. Further, for the Pareto-efficient hybridization, we allow for adjusting the compromise between the metrics, so that the recommendation emphasis can be set dinamically according to the needs of different users.

**Keywords:** Hybrid Recommender Systems, Pareto-Efficiency, Diversity, Novelty.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Recommender systems are increasingly emerging as enabling mechanisms devoted to overcoming problems that are inherent to information overload. The rise of the Web, amongst other factors, made it the case that there is a variety of content domains where users have the time (and patience) to browse through all the possible options. Netflix[1], for example, provides thousands of movies and TV shows available for streaming. Another astounding example is Last.fm[2]: in 2009, Last.fm radio worked with over 80,000 labels and artists, which resulted in a catalogue of over 7 million tracks[3]. Finally, running a simple query like "book", on Amazon's[4] catalogue search, yields 42,705,980 results[5], an amount that no sane user would dare to browse in search of something new. Recommender systems help users navigate through this avalanche of items, (ideally) presenting to them only items that will be of interest, in a personalized manner.

Personalization is what separates the recommender system from information retrieval systems and traditional search engines (modern search engines are blurring this distinction, though). In traditional search engines, the user expresses an information need (usually through a textual query), and the system retrieves the most relevant items according to that query. In recommender systems, user behaviour and other sources or data are used in order to present to the user items that will be of interest to that particular user (or going even further, to that particular user in a particular context).

[1] www.netflix.com
[2] www.last.fm
[3] http://blog.last.fm/2009/03/24/lastfm-radio-announcement
[4] www.amazon.com
[5] This query was performed on the 15th of January, 2013

Historically, much of recommender systems research focused on the task of rating prediction - that is, trying to predict the rating that an user will give to a given item. Certainly, the Netflix challenge (as presented by Bennett et al. [2007]) had its contribution to this, as they provided a very large dataset to the community, with this task in mind. However, as argued by Cremonesi et al. [2010] and Herlocker et al. [2004], the top-$N$ recommendation task (where a list of the "best bet" recommendations are shown, without the predicted rating values) better represents the reality of many commercial systems. Cremonesi et al. [2010] showed that algorithms optimized for minimizing rating prediction metrics do not necessarily perform well in terms of top-$N$ recommendation. In this work, we focus our efforts on the top-$N$ recommendation task.

## 1.2 Objectives

Even considering the top-$N$ task, the typical goal of a recommender system still is to maximize accuracy. More recently, however, there is increasing consensus that the success of a recommender system depends on other dimensions of information utility. More specifically, even being accurate, obvious and monotonous recommendations are generally of little use, since they don't expose users to unprecedent experiences (see Celma and Herrera [2008]; Herlocker et al. [2004]; McNee et al. [2006]; Vargas and Castells [2011]).

In particular, as noted by Vargas and Castells [2011], "*novelty and diversity are being identified as key dimensions of recommendation utility in real scenarios*". A novel recommendation would be of an item that is not amongst the most popular items, and therefore is less likely to be already known by the user. A diverse recommendation list would be a list where the items recommended are not alike one another.

In this work, we call the three dimensions (accuracy, novelty and diversity) *objectives*. The main goal of this work is to propose a multi-objective algorithm for recommender systems.

## 1.3 Hybrid Multi-Objective Recommender Systems

Increasing novelty and diversity by completely giving up on accuracy is straightforward - and meaningless, since the system will not meet the users needs anymore. In fact, there is an apparent trade-off between these dimensions, which becomes evident by inspecting the performance of existing top-$N$ recommendation algorithms (as

we show in Chapter 4). An easy conclusion is that different algorithms may perform distinctly depending on the dimension of interest (i.e., the best performer in terms of accuracy is not the best one in terms of novelty and diversity), and thus it is hard to point to a best performer if all the dimensions are considered simultaneously. In traditional (accuracy-focused) recommender systems research, the need to combine recommendation techniques to achieve peak performance is apparent (one notorious example is the work by Bell et al. [2007]). Such combinations are called hybrids. The potential synergy between different recommendation algorithms is of great importance to multi-objective recommender systems, since they must achieve a proper level of each dimension (i.e., objective).

In this work we tackle this problem by proposing algorithms based on the concept of *Pareto Efficiency*. This is a central concept in Economics, which informally states that "*when some action could be done to make at least one person better off without hurting anyone else, then it should be done.*" This action is called Pareto improvement, and a system is said to be Pareto-Efficient if no such improvement is possible. The same concept may be exploited for the sake of devising multi-objective recommender systems that are built by combining existing recommendation algorithms, as it fits perfectly with trying to optimize multiple objectives at the same time. In this case, the most efficient recommender system is the one which cannot improve an objective further (i.e., accuracy, diversity or novelty) without hurting the other objectives.

Given that existing recommendation algorithms are complementary in the sense that they greatly differ in their level of accuracy, novelty and diversity, we exploit the Pareto-Efficiency concept in two distinct manners:

1. Pareto-Efficient Ranking: Each possible item is associated with a point in an $n$-dimensional scattergram, which we call the user-interest space. In this case, a point is represented as $[c_1, c_2, \ldots, c_n]$, where each coordinate $c_i$ corresponds to the relevance score estimated by a different recommendation algorithm. Points that are not dominated by any other point in the scattergram compose the Pareto frontier (Goldberg [1989]; Zitzler and Thiele [1999]). Points lying in the frontier correspond to cases for which no Pareto improvement is possible, being therefore items more likely to be simultaneously accurate, novel and diversified. This part of the work, was published in Ribeiro et al. [2012].

2. Pareto-Efficient Hybridization: The final relevance score of an arbitrary item is estimated using a linear combination $(\alpha \times c_1 + \beta \times c_2 + \ldots + \theta \times c_n)$ of the relevance scores estimated by $n$ different existing recommendation algorithms. In this case, we have a 3-dimensional scattergram, which we call the objective

space. Each point in this scattergram corresponds to the level of accuracy, novelty and diversity achieved by a possible hybrid recommendation algorithm. We may search for weights (i.e., $\alpha, \beta, \ldots, \theta$) for which the corresponding points lie in the Pareto frontier, and then choose the hybrid that best fits the system priority. This part of the work was publiched in Ribeiro et al. [2013].

Both algorithms use machine learning components: Pareto Efficient Ranking uses a machine learning based ranking technique in order to try to estimate the likelihood that an item is in a Pareto Frontier, while Pareto Efficient Hybridization uses a genetic algorithm in order to find the appropriate weights. These procedures are run in an off-line setting, that is, the ranked lists are produced before the user interacts with the system. We conducted a systematic evaluation involving different recommendation scenarios, with explicit user feedback (i.e., movies from the MovieLens dataset, presented by Miller et al. [2003]), as well as implicit user feedback (i.e., artists from the last.fm dataset, made available by Celma and Herrera [2008]).

The experiments showed that it is possible to (i) combine different algorithms in order to produce better recommendations and (ii) control the desired balance between accuracy, novelty and diversity. In most cases the proposed algorithms produce systems that improve diversity and novelty without compromising accuracy, when compared against the results obtained with the best algorithms in isolation, or improve accuracy without compromising the other two objectives. Further, the comparison against multi-objective baselines indicates the superiority of our proposed algorithms, which provide significant gains in terms of all three criteria considered in our analysis.

To the best of our knowledge, the algorithms we introduce in this work differ from all existing multi-objective recommendation algorithms. We exploit the notion of Pareto-Efficiency in order to sort items that balance accuracy, novelty and diversity. The Pareto-Efficiency concept was already employed in recommender systems that must cope with additional dimensions such as user privacy (Dokoohaki et al. [2010]) and friendship (Naruchitparames et al. [2011]), but our scenario is much more challenging, involving competing objectives. Our first algorithm employs the Pareto frontier to find a partial ordering between items, and by avoiding items located at the extreme positions of the frontier it finds items that are likely to be simultaneously interesting in terms of accuracy, diversity and novelty. Our second algorithm employs the Pareto frontier to find hybrids that are more likely to perform suggestions that are simultaneously interesting in terms of accuracy, diversity and novelty. Our proposed algorithms are highly practical and effective for multi-objective recommender systems, as shown in our experiments.

## 1.4 Outline of the Thesis

The remainder of this work is organized as follows:

- Chapter 2 [Background]: Summarizes existing work related to evolutionary algorithms, multi-objective optimization and their applications to recommender systems, as well as existing work on hybrid recommender systems.
- Chapter 3 [Pareto-Efficient Algorithms]: Describes our algorithms for combining multiple recommendation algorithms.
- Chapter 4 [Experimental Evaluation]: Presents an experimental evaluation of the proposed algorithms in two different application scenarios.
- Chapter 5 [Conclusions]: Discusses the main conclusions, contributions and limitations of the proposed work.

# Chapter 2

# Background and Related Work

In this section we review the main concepts about evolutionary algorithms and multi-objective optimization. Finally, we discuss related work on hybrid and multi-objective recommender systems.

## 2.1 Evolutionary Algorithms

Evolutionary algorithms are meta-heuristic optimization techniques that follow processes such as inheritance and evolution as key components in the design and implementation of computer-based problem solving systems (Eiben and Smith [2003]; Holland [1975]). In evolutionary algorithms, a solution to a problem is represented as an individual in a population pool. The individuals may be represented as different data structures, such as vectors, trees, or stacks, according to Michalewicz [1996]. If the individual is represented as a vector, for example, each position in the vector is called a gene.

Typically, evolutionary algorithms employ a training and a validation set, as described in Algorithm 1. Initially, the population starts with individuals created randomly (line 6). The evolutionary process is composed of a sequence of candidate solution generations. The process evolves generation by generation through genetic operations (lines 7-12). The goal of this process is to obtain better solutions after some generations. A fitness function is used to assign a fitness value to each individual (line 9), which represents how well it performed on the training set or in a cross validation set. To produce a new generation, genetic operators are applied to individuals with the aim of creating more diverse and better individuals (line 12). Typical operators include reproduction, mutation, and crossover. The reproduction operator is used to breed new individuals identical to their parents, favoring those parents with highest

**Algorithm 1:** Evolutionary Algorithm.

```
1  Let M be a training set
2  Let V be a validation set
3  Let N_g be the number of generations
4  Let N_I be the number of individuals
5  S ← ∅
6  P ← initial random population of individuals
7  For each generation g of N_g do
8      For each individual i ∈ P do
9          fitness ← fitness(i, M, V)
10     S_g ← N_I top-ranked individuals of generation g
       according to their fitness
11     S ← S ∪ S_g
12     P ← New population created by applying genetic
       operators to individuals in S_g
13 BestIndividual← SelectionMethod(S)
```

values of the fitness function. The crossover operator takes two individuals (parents) to breed a new one by exchanging subparts of the parent chromosomes (roughly mimicking a mating process). The mutation operator simulates the deviations that may occur in the reproduction process by a random perturbation of the chromosomes (e.g. replacing a gene by another).

## 2.2 Multi-Objective Optimization

Since we are interested in maximizing three different objectives for the sake of recommender systems (i.e. accuracy, novelty, and diversity), we model our algorithm using a multi-objective evolutionary algorithm. In multi-objective optimization problems there is a set of solutions that are superior to the remainder when all the objectives are considered together. In general, traditional algorithms for multi-objective optimization problems are very limited because they become too expensive as the size of the problem grows (see the work by Cecchini et al. [2010]). Multi-objective evolutionary algorithms are a suitable option to overcome such an issue.

Typically, multi-objective evolutionary algorithms are classified as Pareto or non-Pareto, according to Zitzler and Thiele [1999]. In the non-Pareto optimization case, the objectives are combined into a single evaluation value that is used as fitness value (i.e., average of the objectives). In Pareto algorithms, on the other hand, a vector of objective values is used (i.e., the individual is given as an objective vector). The evaluation of Pareto algorithms follows the *Pareto dominance* concept. An individual

dominates another if it performs better in all of the objectives considered. Given two arbitrary individuals, the result of the dominance operation has two possibilities: (i) one individual dominates another, or (ii) the two individuals do not dominate each other. An individual is denoted as *non-dominated* if it is not dominated by any other individual in the population, and the set of all non-dominated individuals compose the *Pareto frontier*.

The reason we use approaches based on the Pareto dominance concept instead of combining the objectives into a single evaluation measure is twofold: (1) the concept lends itself nicely to the multi-objective scenario, where one wants to keep the balance between the different objectives, and (2) by using the Pareto dominance concept we are able to tweak the compromise between the objectives according to different needs, which we do by using the Pareto Frontier in the Pareto-Efficient Hybridization algorithm (see Section 3.2).

In this work we use a second version of the strength Pareto evolutionary algorithm (SPEA-2), proposed by Zitzler et al. [2001]; Zitzler and Thiele [1999]. The The aim is to find or approximate the Pareto-optimal set for multi-objective problems. The main features of this algorithm are: (i) the fitness assignment scheme takes into account how many individuals each individual dominates or is dominated by, (ii) it uses a nearest neighbour density estimation technique to break ties in solutions with the same fitness, (iii) the size of the population of non-dominated solutions is a fixed value $\eta$. Thus, we have two situations. First, when the actual number of non-dominated solutions is lower than $\eta$, the population is filled with dominated solutions; second, when the actual number of non-dominated solutions exceeds $\eta$, some of them are discarded by a truncation operator which preserves boundary conditions, even though we always keep the current Pareto Frontier in a list separate from the population, so we can later retrieve the individuals in it.

## 2.3 Related Work

Traditionally, hybrid recommender strategies are the combination of two different families of algorithms - namely, content-based and collaborative filtering (see Adomavicius and Tuzhilin [2005]). In this work, we combine many (up to eight) recommendation algorithms - different content-based and collaborative filtering algorithms that deal with explicit and implicit feedback, etc. We treat each recommendation algorithm as a black-box, so adding or removing recommendation algorithms is easy.

Burke [2002] identified seven types of hybrids:

- Weighted: The scores from each of the different recommendation components are combined numerically.
- Switching: The system chooses among recommendation components and applies the selected one.
- Mixed: Recommendations from different recommendation components are presented together.
- Feature Combination: Features from different knowledge sources are combined together and given to a single recommendation algorithm.
- Feature Augmentation: One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
- Cascade: Recommenders are applied in a certain order, such that each recommender breaks ties from the previous ones.
- Meta-level: One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

Out of these seven types, three are not appropriate for combining many recommendation algorithms. The feature combination type is a hybrid that does not even combine recommendation algorithms. The cascade type is appropriate for few recommenders, as there are only so many ties the next algorithm can break. Finally, the meta-level type requires very specific algorithm sequences, which are hard enough for two algorithms, let alone many.

Different hybridization strategies have been proposed in the literature. Examples of weighted hybrids can be found in the works by Claypool et al. [1999] and Pazzani [1999] (which is a voting mechanism). Switching examples are found in the works by Billsus and Pazzani [2000]; Lekakos and Caravelas [2008]. An example of the cascade type was proposed by Burke [2002] himself. A notable example of feature augmentation is the work by Bao et al. [2009], where the predictions scores of a group of algorithms become features to another learning algorithm, together with additional meta-features. The authors show that using two additional meta-features - namely the number of users that rated each movie and the number of movies that each user rated - gave them excellent results. Their hybrid, called STREAM, is used as a baseline in this work.

A prominent use of hybridization in recommender systems is the Belkor system that won the Netflix competition, presented by Bell et al. [2007]; Bennett et al. [2007].

Their method is a weighted linear combination of 107 collaborative filtering engines. There are important differences between their work and ours: (i) their solution is single-objective (accuracy), (ii) they combine only collaborative filtering information, and (iii) the recommendation task is rating prediction, focused on RMSE (Root Mean Squared Error) - which makes the aggregation simpler, since all of the ratings are on the same scale and consist of the same items.

There have been several research efforts to apply machine learning / artificial intelligence methods to the problem of combining different recommendation algorithms. STREAM (Bao et al. [2009]), which has been discussed before, is a notable example - it is an application of the traditional ensemble strategy *stacking* (see Polikar [2006]) applied to recommender algorithms. The authors combine the recommenders with either Linear Regression (Mitchell [1997]), model tree (Wang and Witten [1997]) or bagged model trees (see Hastie et al. [2001]). In their case, the task was rating prediction, and the best machine learning algorithm was bagged model trees. In our top-$N$ case, the algorithm that did the best was linear regression, so we reported the results for STREAM using linear regression as a baseline.

The Belkor system is also a case of using machine learning (linear regression) in order to combine a variety of different recommendation algorithms. In a similar fashion, Basu et al. [1998] applied the inductive rule learner Ripper in order to combine user ratings and content features. Basilico and Hofmann [2004] designed an SVM-like model for a feature combination algorithm, with a kernel function that is based on joint features of user ratings as well as user or item attributes. We use machine learning in both of the proposed algorithms - a genetic algorithm for Pareto-Efficient Hybridization and SVM-rank for Pareto-Efficient Ranking. However, we use these learning algorithms (and the Pareto concept) in order to achieve better results on the whole objective space (more specifically, on accuracy, novelty and diversity), instead of using them to improve accuracy, as is the case with the aforementioned works.

There has been an increasing consensus in the recommender systems community about the importance of proposing algorithms and methods to enhance novelty and diversity, as seen by Ge et al. [2010]; Vargas and Castells [2011]. As showed by Ziegler et al. [2005], user satisfaction does not always correlate with high recommender accuracy. Thus, different multi-objective algorithms have been proposed to improve user experience considering either diversity or novelty. For instance, Ziegler et al. [2005] define a greedy re-ranking algorithm that diversifies baseline recommendations. Another strategy to improve diversity is presented by Zhang and Hurley [2008], where they suggest an optimization method to improve two objective functions reflecting preference similarity and item diversity.

On the other hand, novelty has been understood as recommending long-tail items, i.e., those items which few users have accessed. Vargas and Castells [2011] present hybrid strategies that combine collaborative filtering with graph spreading techniques to improve novelty. Celma and Herrera [2008] take an alternative approach: instead of assessing novelty in terms of the long-tail items that are recommended, they follow the paths leading from recommendations to the long tail using similarity links. As far as we know, this is the first work that proposes a hybrid method that is multi-objective in terms of the three metrics, i.e., accuracy, diversity and novelty.

As for previous research involving multi-objective algorithms in the context of recommendation, Agarwal et al. [2011] present a multi-objective approach to optimize jointly for clicks and post-click downstream utilities (such as revenue, time spent, etc), in the context of content recommendation. However, they handle the trade-off by showing a portion of their users results optimized for relevance, while showing results optimized for another downstream measure for another group of users - which is something that makes sense for such a business, but not so much for movie and music recommendation. Jambor and Wang [2010], on the other hand, formulate the problem as a simple linear optimization problem, where each objective is considered as a constraint. Rodriguez et al. [2012] expand the previous formulation, allowing for nonlinear objective and constraint functions, for a user recommendation scenario on LinkedIn[1].

Extensive research has also been performed exploiting the robust characteristics of genetic algorithms in recommender systems. For instance, Pagonis and Clark [2010] build a content-based recommender system and use genetic algorithms to assign proper weights to the words. Such weights are combined using the traditional IR *vector space model* (see Baeza-Yates and Ribeiro-Neto [tion]) to produce recommendations. Minchul Jung et al. [2008] use a genetic algorithm to build a recommender method that considers the browsing history of users in real-time. In contrast to our method (which uses a Genetic Algorithm (GA) to combine multiple recommender methods), they use a GA to build a single-method.

Hwang [2010], presents a implementation of GA for optimal feature weighting in the multi-criteria scenario. Their application of GA consists in selecting features that represent users' interest in a collaborative filtering context, in contrast to our method, which focuses on assigning weights to different recommendation algorithms in order to improve he overall performance in terms of accuracy, novelty and diversity.

[1] www.linkedin.com

# Chapter 3

# Pareto-Efficient Algorithms

In this chapter, we present the Pareto-efficient concept in two distinct manners: first, we introduce Pareto-efficient ranking, where each item is associated with a point in a n-dimensional scattergram called the user-interest space. Next, we present Pareto-efficient hybridization, where the final score of an item is estimated using a linear combination of recommenders.

## 3.1 Pareto-Efficient Ranking

In this section we introduce our algorithm for Pareto-efficient ranking. We start by discussing how possible items are disposed in a user-interest space by exploiting different recommendation biases within existing recommendation algorithms. Then, we discuss how the user-interest space is used in order to aggregate multiple ranked lists into a final, Pareto-efficient, ranked list.

### 3.1.1 Recommendation Bias and User-Interest Space

Typically, a recommender system arranges items into a ranked list, so that the top-$k$ items are those most interesting to the user. Although being naturally subjective, the potential interest a user will have in the top-$k$ items may be approximated by the following interestingness measures:

- Accuracy: returns how well the top-$k$ items meet the user's information need.
- Novelty: is inherently linked to the notion of discovery and returns how novel to the user are the top-$k$ items. Further, top-$k$ items are assumed to be accurate (i.e., relevant).

- Diversity: returns how different with respect to each other are the top-$k$ items. Further, top-$k$ items are assumed to be accurate.

Existing recommendation algorithms differ by large in their level of accuracy, novelty and diversity. The difference is due to a distinct recommendation bias which is followed by each algorithm, that is, existing algorithms may favor different interestingness measures. However, it is already a consensus that all three interestingness measures are essential to effective recommendation, since together these measures have a complementary effect which is highly desirable for recommender systems. That is, accurate suggestions are of little value if they are obvious to the user. Besides, suggesting items that are too similar to each other leads to monotonous and ineffective recommendations. Therefore, in order to ensure effective results, the top-$k$ items within a ranked list must be as accurate, novel and diverse as possible.

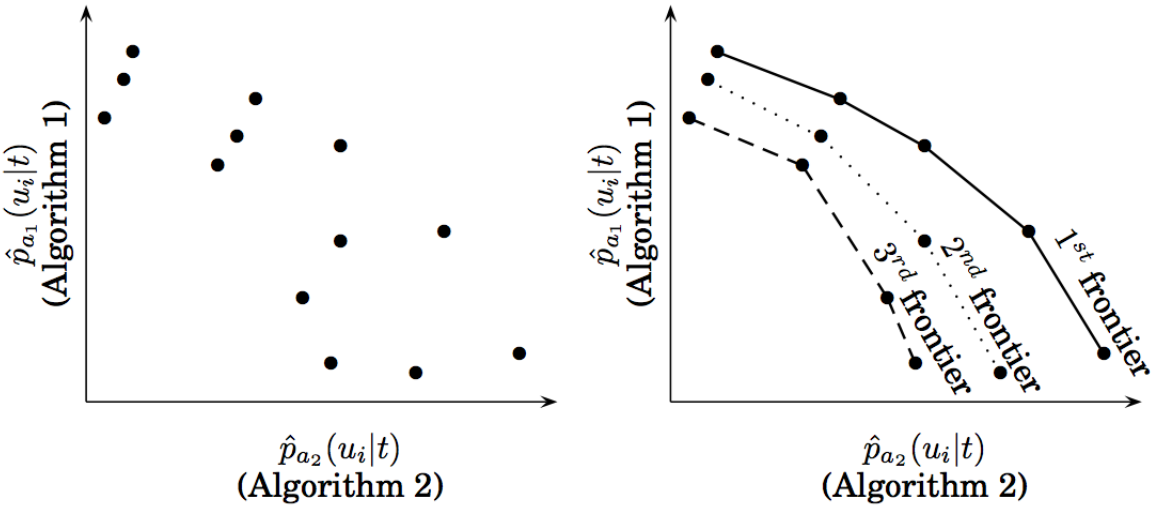


Figure 3.1: Left − User-Interest space according to two different recommendation biases (i.e., different recommendation algorithms). Points are possible items and are represented by the relevance level estimated by different algorithms. Right − Non-dominated items form successive Pareto frontiers.

Consider the set of constituent recommendation algorithms $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ and assume that these algorithms assign to each possible item a score $\hat{p}_{a_j}(u_i|t)$ corresponding to the potential interest user $t$ has on item $u_i$, we may represent each item $u_i$ as a point in a $n$-dimensional user-interest space: $\mathcal{S}_t = [\hat{p}_{a_1}(u_i|t), \hat{p}_{a_2}(u_i|t), \ldots, \hat{p}_{a_n}(u_i|t)]_{i=1}^{m}$, where $m$ is the number of possible items and each $\hat{p}_{a_j}(u_i|t)$ is calculated using one out of $n$ different constituent recommendation algorithms (i.e., following different recommendation biases). Figure 3.1 (Left) depicts a 2-dimensional user-interest

space. The dominance operator relates two items in such space, so that the result of the dominance operation has two possibilities: (i) one item dominates another or (ii) the two items do not dominate each other. We need now the following definition.

**Definition 1:** *A Pareto-Efficient ranked list for user $t$ is an ordered list of $m$ items $\mathcal{L}_t = \{u_1, u_2, \ldots, u_m\}$ such that there is no pair $(u_i, u_j) \in \mathcal{L}_t$ for which $u_i$ dominates $u_j$, given that $i > j$.*

### 3.1.2 Building Pareto-Efficient Ranked Lists

Algorithm 2 builds a Pareto-Efficient ranked list for user $t$. Items that are not dominated by any other item in $\mathcal{S}_t$ lie on the Pareto frontier, as shown in Figure 3.1 (Right). Stripping off an item from the Pareto frontier, and building another frontier[1] from the remaining items in $\mathcal{S}_t$ reveals a partial ordering between the items, which we call a Pareto-Efficient ranking.

**Algorithm 2:** Pareto-Efficient Ranking.

**Input**: $\mathcal{S}_t$ (the $n$-dimensional interest space for user $t$), and $k$ (the number of suggested items).
**Output**: $\mathcal{L}_t$ (a Pareto-Efficient ranked list for user $t$).
Build all the Pareto frontiers in $\mathcal{S}_t$;
**repeat**
  include an item $x$ into $\mathcal{L}_t$;
  remove $x$ from $\mathcal{S}_t$;
**until** $|\mathcal{L}_t| = k$;

Next, we discuss different strategies for building Pareto-Efficient ranked lists for each user $t$. These strategies are based on Algorithm 2, and the only difference between them resides on the item that is selected at each iteration (i.e., item $x$ on Algorithm 2). Still, our strategies try to avoid selecting items located at extreme positions of the frontier, since such items may privilege a specific measure. Instead, highly dominant items, or items that are representative of other items in the frontier, are more likely to balance multiple objectives.

[1] There are efficient algorithms for building and maintaining the Pareto frontier, such as the ones based on skyline queries Lin et al. [2007]; Papadias et al. [2003]. In particular, we employed the skyline operator algorithm proposed in Börzsönyi et al. [2001], ensuring $O(n \times m \times k)$ complexity.

#### Most Dominant Items First

This strategy aims at selecting the item lying in the current Pareto frontier which dominates more items in the user-interest space $\mathcal{S}_t$, as given by:

$$u_i \text{ such that } \arg\max(dom(u_i)), \forall\ u_i \in \mathcal{S}_t$$

where $dom(u_i)$ is the number of items dominated by $u_i$.

The number of items that are dominated by an arbitrary item $u_i$ is easily obtained while building the Pareto frontier, and it remains unchanged as most dominant items are removed from $\mathcal{S}_t$, ensuring the efficiency of the process. It is worth noting that the first frontier is exhausted before items in the second frontier are selected, and so on.

#### Learning to Rank

This strategy aims at selecting the item which is more likely to be located in the first Pareto frontier. To this end, when training, we label items according to the frontier they are located (i.e., first, second, ..., frontiers), so that items lying in the first frontiers are labeled as more relevant than items lying in subsequent frontiers. Then, when testing, we apply a well-known learning to rank algorithm, SVM-Rank (Joachims [2002]), in order to sort items accordingly to their potential relevance level. Specifically, we model the training data as item-user pairs, and each pair is labeled with the Pareto frontier in which the corresponding item is located. SVM-Rank formalizes the ranking problem as a binary classification problem on instance pairs, and then solve the problem using SVMs (Joachims [2006]).

## 3.2 Pareto-Efficient Hybridization

In this section we introduce a weighted Pareto-efficient hybridization algorithm. We start by discussing how different recommendation algorithms are combined in a standard weighted manner. Then we describe the evolutionary search for Pareto-Optimal hybrids. Finally, we discuss a strategy to deal with the compromise between accuracy, novelty and diversity, so that the system is able to adjust itself for different user perspectives.

### 3.2.1 Weighted Hybridization

Our hybridization algorithm is based on assigning weights to each constituent algorithm. We denote the set of constituent algorithms as $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, and we suppose that these algorithms assign to each possible item a score $\hat{p}_{a_j}(u_i|t)$ corresponding to the potential interest user $t$ has on item $u_i$. Since the constituent algorithms may output scores in drastically different scales, a simple normalization procedure is necessary to ensure that all algorithms in $\mathcal{A}$ operate in the same scale. The aggregated score for each item $i$ is calculated as follows:

$$\hat{p}(u_i|t) = \sum_{j=1}^{n} \hat{p}_{a_j}(u_i|t) \times w_{a_j} \tag{3.1}$$

where $w_{a_j}$ is the weight assigned to algorithm $a_j \in \mathcal{A}$. The assignment of weights to each algorithm is formulated as a search problem which we discuss next.

### 3.2.2 Searching for Pareto-Efficient Hybrids

Finding a suitable hybrid, represented as a vector of weights $W = \{w_{a_1}, w_{a_2}, \ldots, w_{a_n}\}$, can be viewed as a search problem in which each $w_{a_i}$ is selected in a way that optimizes a established criterion. We consider the application of evolutionary algorithms for searching optimal solutions. These algorithms iteratively evolve a population of individuals towards optimal solutions by performing genetic-inspired operations, such as reproduction, mutation, recombination, and selection (Goldberg [1989]). Next we precisely define an individual.

**Definition 2:** *An individual is a candidate solution, which is encoded as a sequence of n values* $[w_{a_1}, w_{a_2}, \ldots, w_{a_n}]$*, where each* $w_{a_i}$ *indicates the weight associated with algorithm* $a_i \in \mathcal{A}$*.*

Each constituent algorithm $a_i$ assigns scores to items using a cross-validation set. Finally, weights are assigned to each recommendation algorithm and their scores are aggregated according to Equation 3.1, producing an individual (i.e., an hybrid). A fitness function is computed for each individual in order to make them directly comparable, so that the population can evolve towards optimal solutions (i.e., individuals located closer to the Pareto frontier).

**Definition 3:** *An optimal solution is a sequence of weights* $W = \{w_{a_1}, w_{a_2}, \ldots, w_{a_n}\}$*, satisfying Equation 3.2*:

$$\text{maximize } \phi(o_i) \; \forall o_i \in \{\text{accuracy}, \text{novelty}, \text{diversity}\} \tag{3.2}$$

where $\phi(o_i)$ is the value of an objective $o_i$, which can be either accuracy, novelty or diversity. Thus, the performance of each individual is given by a 3-dimensional objective vector, containing the average accuracy, novelty and diversity over all users in the cross validation set. Searching for optimal hybrids is a multi-objective optimization problem, in which the value of $\phi(o_i)$ must be maximized for each of the three objectives that compose an optimal solution. Therefore, multiple optimal individuals are possible. It is worth noticing that different datasets and combinations of constituent algorithms will generate different optimal individuals.

Again, we exploit the concept of Pareto dominance for solving the multi-objective optimization problem. As a result, given the 3-dimensional objective space, the evolutionary algorithm evolves the population towards producing individuals that are located closer to the Pareto frontier, as illustrated in Figure 3.2.

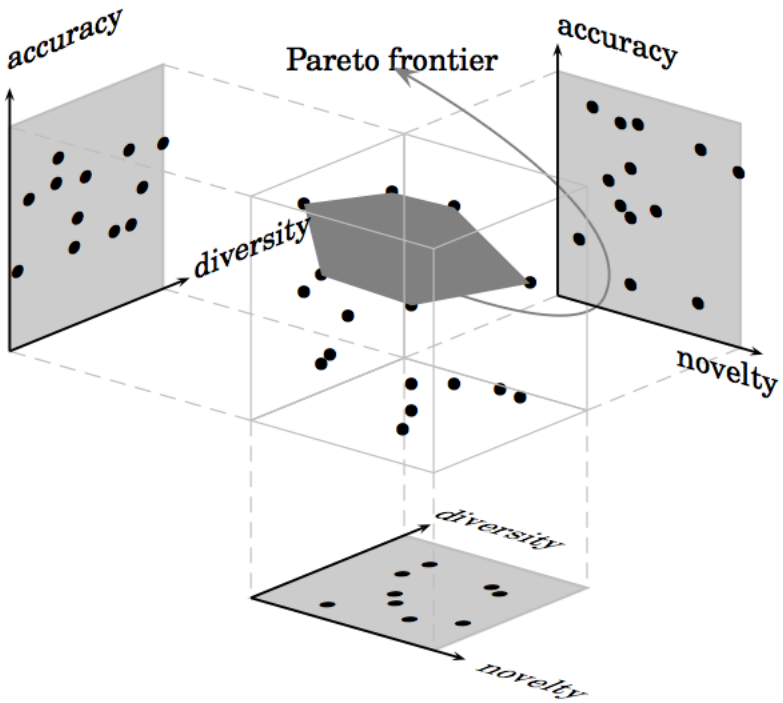


Figure 3.2: A 3-dimensional objective space. Points are possible hybrids and are represented by the corresponding level of accuracy, novelty and diversity. Hybrids lying in the Pareto frontier are not dominated by any other hybrid.

The result is a set of Pareto-efficient hybrids. Under this strategy, we follow the well-known Strength Pareto Evolutionary Algorithm approach (Zitzler and Thiele [1999]; Zitzler et al. [2001]), which has shown to be highly effective and also because it provides more diverse individuals when compared to existing algorithms (such as

those proposed by Corne et al. [2000]; Deb [1999]; Srinivas and Deb [1994]) for many problems of interest. The Strength Pareto approach isolates individuals that achieve a compromise between maximizing the competing objectives by evolving individuals that are likely to be non-dominated by other individuals in the population. Algorithm 3 shows the basic steps of our Pareto-efficient hybridization algorithm.

**Algorithm 3:** Pareto-Efficient Hybridization.

**Input**: $\mathcal{P}$ (the current population of individuals), $p$ (the next population of individuals), and $g$ (the maximum number of generations).
**Output**: Hybrids lying in the Pareto frontier.
**repeat**
  include the best individuals from $\mathcal{P}$ into $p$ (those closer to the frontier);
  apply genetic operators to individuals in $p$;
  update $\mathcal{P}$ with individuals in $p$;
**until** *$g$ generations are produced*;

### 3.2.3 Adjusting the System Priority

It is well recognized that the role that a recommender system plays may vary depending on the target user. For instance, according to Herlocker et al. [2004], the suggestions performed by a recommender system may fail to appear trustworthy to a new user because it does not recommend items the user is sure to enjoy but probably already knows about. Based on this, a recommender system might prioritize accuracy instead of novelty or diversity for new users, while prioritizing novelty for users that have already used the system for a while. This is made possible by our hybridization algorithm, by searching which individual in the Pareto frontier better solves the user's current needs.

The choice of which individual in the Pareto frontier is accomplished by performing a linear search on all of the individuals, in order to find which one maximizes a simple weighted mean on each of the three objectives in the objective vector, where the weights in the weighted mean represent the priority given to each objective. It is worth noting that fitness values are always calculated using the cross-validation set. Therefore, considering a 3-dimensional priority vector $Q = \{q_1, q_2, q_3\}$, that represents the importance of each objective $j$, the individual in the Pareto frontier $\mathcal{P}$ is chosen as follows:

$$\underset{i \in \mathcal{P}}{arg\,max} \sum_{j=1}^{3} q_j \times \phi(o_j) \tag{3.3}$$

Figure 3.3 illustrates this process, in two dimensions (for simplicity). The grey individuals represent the Pareto Frontier. If the system places a higher priority on novelty, the individual in grey on the top left will be chosen as the vector of weights. However, if accuracy receives a higher priority, the individual on the bottom right will be chosen.

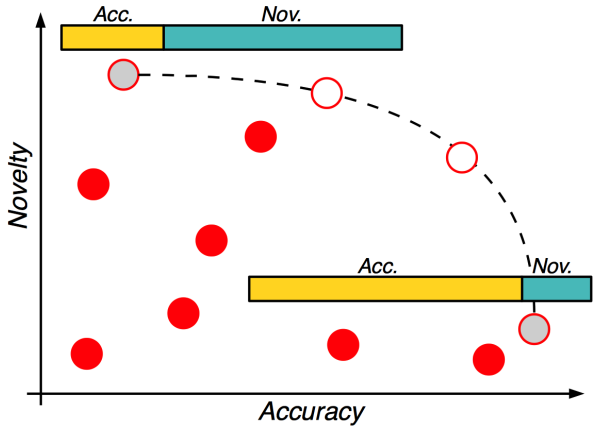


Figure 3.3: An illustration of adjusting the system priority.

# Chapter 4

# Experimental Evaluation

In this chapter we empirically analyze the effectiveness of our proposed Pareto-efficient algorithms for the sake of multi-objective recommender systems. We assume an evaluation setting where recommendation algorithms are compared without user interaction (i.e., offline setting). The experiments were performed on a Linux-based PC with a Intel I5 4.0 GHz processor and 4.0 GBytes RAM.

## 4.1 Evaluation Methodology

The evaluation methodology we adopted in this work is the same as the one proposed by Cremonesi et al. [2010], which is appropriate for the top-$N$ recommendation task. For each dataset, ratings are split into two subsets: the training set (denoted as $\mathcal{M}$), and the test set (denoted as $\mathcal{T}$). The training set $\mathcal{M}$ may be further split (if necessary) into two subsets: the cross-validation training set (denoted as $\mathcal{C}$), and the cross-validation test set (denoted as $\mathcal{V}$), which are used in order to tune parameters or adjust models (when applicable). The test set $\mathcal{T}$ and the cross-validation test set $\mathcal{V}$ only contain items that are considered relevant to the users in the dataset. For explicit feedback (i.e., MovieLens), this means that the sets $\mathcal{T}$ and $\mathcal{V}$ only contain 5-star ratings. An illustration of this procedure can be seen in Figure 4.1.

In the case of implicit feedback (i.e., Last.fm), we normalized the observed item access frequencies of each user to a common rating scale [0,5], as used by Vargas and Castells [2011]. Namely, $r(u,i) = n * F(\mathit{frec}_{u,i})$, where $\mathit{frec}_{u,i}$ is the number of times user $u$ has accessed item $i$, and $F(\mathit{frec}_{u,i}) = |j \in \mathbf{u} | f_{u,j} < f_{u,i}|/|\mathbf{u}|$ is the cumulative distribution function of $\mathit{frec}_{u,i}$ over the set of items accessed by user $u$, denoted as $\mathbf{u}$. In this case, the test set and the cross validation test set only contain ratings such that $r(u,i) >= 4$, since the number of 5-star ratings is very small using this

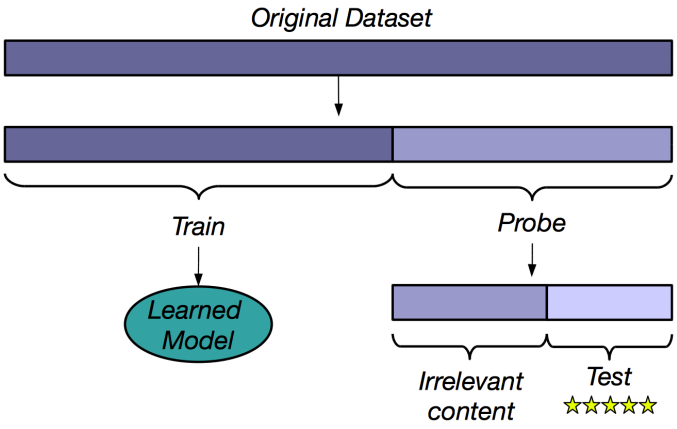


Figure 4.1: Dataset split illustration.

mapping of implicit feedback into ratings. It is worth noting that all the sets have a corresponding implicit feedback set, used by the recommendation algorithms that can deal with implicit feedback.

The detailed procedure to create $\mathcal{M}$ and $\mathcal{T}$ is the same used by Cremonesi et al. [2010], in order to maintain compatibility with their results. Namely, for each dataset we randomly sub-sampled 1.4% of the ratings from the dataset in order to create a probe set. The training set $\mathcal{M}$ contains the remaining ratings, while the test set $\mathcal{T}$ contains all the 5-star ratings in the probe set (in the case of explicit feedback) or 4+ star ratings (in the case of implicit feedback mapped into explicit feedback). We further divided the training set in the same fashion, in order to create the cross-validation training and test sets $\mathcal{C}$ and $\mathcal{V}$. The ratings in the probe sets were not used for training.

In order to evaluate the algorithms, we first train the models using $\mathcal{M}$. Then, for each test item $i$ in $\mathcal{T}$ that is relevant to user $u$:

- We randomly select 1,000 additional items unrated by user $u$. The assumption is that most of them will not be interesting to $u$.
- The algorithm in question forms a ranked list by ordering all of the 1,001 items (relevant test item $i$ + 1000 unrated items). The most accurate result corresponds to the case where the test item $i$ is in the first position.

Since the task is top-$N$ recommendation, we form a top-$N$ list by picking the $N$ items out of the 1,001 that have the highest rank. If the test item $i$ is among the top-$N$ items, we have a *hit*. Otherwise, we have a *miss*. An illustration of the evaluation procedure can be seen in Figure 4.2, where the test item $i$ is ranked 4th by

the algorithm in question. Recall and precision are calculated as follows:

$$recall@N = \frac{\#hits}{|T|} \tag{4.1}$$

$$precision@N = \frac{\#hits}{N * |T|} = \frac{recall@N}{N} \tag{4.2}$$

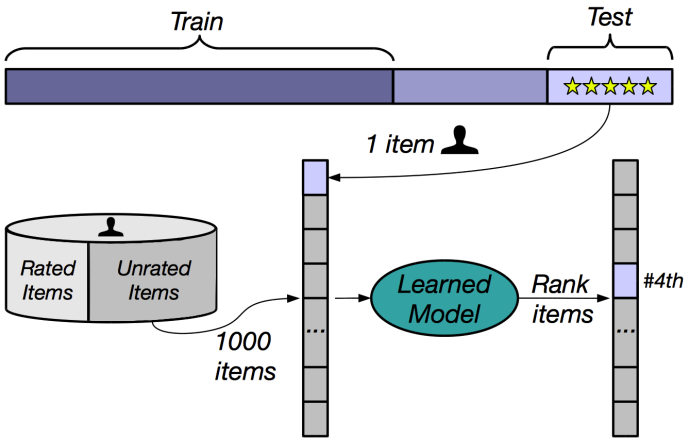


Figure 4.2: Evaluation procedure.

In order to measure the novelty of the suggested items, we used a popularity-based item novelty model proposed by Vargas and Castells [2011], so that the probability of an item $i$ being seen is estimated as:

$$P(seen|i) = \frac{|u \in U | r(u,i) \neq \emptyset|}{|U|} \tag{4.3}$$

where $U$ denotes the set of users. Since the evaluation methodology supposes that most of the 1,000 additional unrated items are not relevant to user $u$, we used the metrics in the framework proposed by Vargas and Castells [2011] without relevance awareness. Finally, the measure of novelty within a top-$N$ recommendation list $R$ presented to user $u$ is therefore given by:

$$EPC@N = C \sum_{i_k \in R}^{i_N} disc(k)(1 - p(seen|i_k)) \tag{4.4}$$

where $disc(k)$ is a rank discount given by $disc(k) = .85^{k-1}$ (following Vargas and Castells [2011]) and $C$ is a normalizing constant given by $1/\sum_{i_k \in R}^{i_N} disc(k)$. Therefore, this metric is rank-sensitive (i.e. the novelty of the top-rated items counts more than the novelty of other items). As is the case with precision and recall, we average the

EPC@N value of the top-$N$ recommendation lists over the test set.

We used a distance based model Vargas and Castells [2011] in order to measure the diversity of the recommendation lists without relevance-awareness. The recommendation diversity, therefore, is given by:

$$EILD@N = \sum_{i_k \in R, i_l \in R, l \neq k}^{i_N, l_N} C_k disc(k) disc(l|k) d(i_k, i_l) \tag{4.5}$$

where $disc(l|k) = disc(max(1, l-k))$ reflects a relative rank discount between $l$ and $k$, and $d(i_k, i_l)$ is the cosine distance between two items, given by:

$$d(i,j) = 1 - \frac{|\mathbf{U_i} \cap \mathbf{U_j}|}{\sqrt{|\mathbf{U_i}|}\sqrt{|\mathbf{U_j}|}} \tag{4.6}$$

such that $\mathbf{U_i}$ denotes the users that liked item $i$, and $\mathbf{U_j}$ denotes the users that liked item $j$.

## 4.2 Experimental Setup

### 4.2.1 Datasets

We apply the methodology presented in the previous section to two different scenarios in order to evaluate the Pareto-efficient algorithms: movie and music recommendation. For movie recommendation, we used the MovieLens 1M dataset (see Miller et al. [2003]). This dataset consists of 1,000,209 ratings from 6,040 users on 3,883 movies. For music recommendation, we used an implicit preference dataset provided by Celma and Herrera [2008], which consists of 19,150,868 user accesses to music tracks on the website Last.fm[1]. This dataset involves 176,948 artists and 992 users, and we considered the task of recommending artists to users. Mapping the implicit feedback into user-artist ratings yielded a total of 889,558 ratings, which were used by the algorithms that cannot deal with implicit feedback, and to separate the dataset into the training and test sets $\mathcal{M}$ and $\mathcal{T}$.

### 4.2.2 Recommendation Algorithms

We selected seven well-known recommendation algorithms to provide the base for our Pareto-efficient algorithms. To represent latent factor models, we selected PureSVD

[1] www.Last.fm

with 50 and 150 factors (PureSVD50 and PureSVD150), described by Cremonesi et al. [2010]. These were the only algorithms we used that are based on explicit feedback. To compute the scores for the items in the Last.fm dataset, we used the mappings of implicit feedback into ratings explained in Section 5.1.

As for recommendation algorithms that use implicit feedback, we used algorithms available in the MyMediaLite package (made available by Gantner et al. [2011]). We used WeightedItemKNN (WIKNN) and WeightedUserKNN (WUKNN) as representative of neighborhood models based on collaborative data (Desrosiers and Karypis [2011]) (we only used WeightedItemKNN on the MovieLens dataset, as MyMediaLite's implementation cannot yet handle datasets where the number of items is very large, which is the case in the Last.fm dataset). Further, we also used MyMediaLite's MostPopular implementation, which is the same as TopPop in Cremonesi et al. [2010]. We also used WRMF − a weighted matrix factorization method based on the work by Hu et al. [2008]; Pan et al. [2008], which is very effective for data with implicit feedback. Finally, we used UserAttributeKNN (UAKNN), a K-nearest neighbor user-based collaborative filtering using cosine-similarity over the user attributes, such as sex, age etc. (which both datasets provide).

### 4.2.3 Baselines

We employed three baselines for the sake of comparison. The first baseline is a voting-based approach based on Borda-Count (BC) which is similar to the method by Pazzani [1999], where each constituent algorithm gives $n$ points to each item $i$ such that $n = |R| - p_i$, where $|R|$ is the size of the recommendation list and $p_i$ is the position of $i$ in $R$. The second baseline is STREAM, a stacking-based algorithm with additional meta-features, proposed by Bao et al. [2009]. We used the same additional meta-features as Bao et al. [2009], namely, the number of items that a certain user has rated and the number of users that has rated a certain item (denoted as $RM_1$ and $RM_2$). We tried the learning algorithms proposed by Bao et al. [2009], and Linear Regression yielded the best results, so the results presented for STREAM are generated using Linear Regression as the meta-learning algorithm. Our last baseline is the weighted hybrid we proposed in Section 4.1, using equal weights for each constituent algorithm. We called this baseline Equal Weights (EW).

### 4.2.4 Pareto Efficient Hybridization Details

We apply the algorithm described in section 3.2 to both datasets, combining all of the recommendation algorithms described in subsection 4.2.2. We used an open-source implementation of SPEA2 (Zitzler and Thiele [1999]; Zitzler et al. [2001]) from DEAP[2]. We used a two points crossover operator (see Holland [1975]), and a uniform random mutation operator with probability .05. Table 4.1 presents SPEA-2's parameters, which were sufficient for convergence.

| Parameters | MovieLens | Last.fm |
|---|---|---|
| Population Size | 100 | 100 |
| Gene dimension | 7 algorithms | 6 algorithms |
| # of Objectives | 3 | 3 |
| # of Generations | 300 | 300 |
| Mutation Rate | .2 | .2 |
| Crossover Rate | .5 | .5 |

Table 4.1: Parameters of the SPEA2 Algorithm

In order to speed up the fitness calculations, we ran all of the constituent algorithms on the cross validation test set and stored their predictions. Then, in order to evaluate the fitness of each individual, we combine the constituent algorithms with the appropriate weights and evaluate the results on the cross validation test set $\mathcal{V}$. It is worth remembering that $\mathcal{V}$ is a list of triples $(u, i, s)$, where $u$ is an user, $i$ is an item that is relevant to $u$ and $s$ is a set of 1,000 items that are unrated by $u$.

Each objective in the fitness function of a certain ranking $R$ of the items $\{i\} + s$ provided by a certain individual is given by:

$$O(R) = \sum_{(u,i,s)\in\mathcal{V}} f(u, i, s, R) \tag{4.7}$$

For the accuracy objective, $f(u, i, s, R)$ is defined as follows:

$$f(u, i, s, R) = 21 - max(21, R_i) \tag{4.8}$$

where $R_i$ is the position of item $i$ in the ranking. This equation provides a way to value hits up to the 20th position, with more value being given to positions closer to the top.

[2]Freely available at http://deap.googlecode.com

| | | Accuracy | | | | | | | | Novelty | Diversity |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Algorithm | R@1 | R@5 | R@10 | R@20 | P@1 | P@5 | P@10 | P@20 | EPC@20 | EILD@20 |
| Const. Algorithms | PSVD50 † | **.1900** | **.4155** | **.5402** | **.6643** | **.1900** | **.0831** | **.0540** | **.0332** | .8070 | .8650 |
| | PSVD150 •◇ | .1237 | .3203 | .4450 | .5658 | .1237 | .0641 | .0445 | .0283 | .8519 | **.8881** |
| | TopPop | .0722 | .2061 | .2895 | .3994 | .0722 | .0412 | .0289 | .0200 | .7079 | .7905 |
| | WRMF | .1513 | .3453 | .4545 | .5674 | .1513 | .0691 | .0455 | .0284 | .7847 | .8394 |
| | WIKNN | .1529 | .3564 | .4624 | .5806 | .1529 | .0713 | .0462 | .0290 | .7744 | .8257 |
| | WUKNN | .1510 | .3364 | .4437 | .5707 | .1510 | .0673 | .0444 | .0285 | .7560 | .8216 |
| | UAKNN | .0614 | .1762 | .2504 | .3387 | .0614 | .0352 | .0250 | .0169 | .7386 | .8173 |
| Baselines | STREAM | **.1792** | **.3961** | **.5169** | **.6426** | **.1792** | **.0792** | **.0517** | **.0321** | .8078 | .8454 |
| | BC | .0473 | .1657 | .2639 | .4352 | .0473 | .0331 | .0264 | .0218 | **.8210** | **.8698** |
| | EW | .1562 | .3574 | .4752 | .5980 | .1562 | .0715 | .0475 | .0299 | .7441 | .8160 |
| Our Algorithms | PEH-mean † • ◇ | .1776 | .4175 | .5379 | .6656 | .1776 | .0835 | .0538 | .0333 | .8361 | .8696 |
| | PEH-acc † | .1959 | .4161 | .5399 | **.6689** | .1959 | .0832 | .0540 | **.0334** | .8188 | .8565 |
| | PEH-nov • | .1415 | .3656 | .4857 | .5917 | .1415 | .0731 | .0486 | .0296 | .8649 | .8964 |
| | PEH-div ◇ | .1309 | .3223 | .4263 | .5297 | .1309 | .0645 | .0426 | .0265 | **.8828** | .9047 |
| | PER-dom † • ◇ | **.1979** | .3722 | .4368 | .4910 | **.1979** | .0744 | .0437 | .0245 | .8549 | **.9060** |
| | PER-SVM † • ◇ | .1953 | **.4296** | **.5540** | .6554 | .1953 | **.0852** | **.0554** | .0328 | .8341 | .8699 |

Table 4.2: Results for Recommendation Algorithms on the MovieLens dataset, with the three objectives (i.e., accuracy, novelty, and diversity). The recommender methods variants are grouped into: (i) constituent algorithms, (ii) multi-objective baselines, and (iii) our proposed Pareto-efficient algorithms. We used the symbols: †, •, ◇ to point out our method and the respective baseline. For each group, the best results for each metric are in bold. Underlined values means that the selected algorithm and the respective baseline are statistically different (95%).

As for the novelty objective $f(u, i, s, R)$ is simply $EPC@20(R)$. Similarly, for the diversity objective, $f(u, i, s, R)$ is equal to $EILD@20(R)$.

## 4.3 Results and Discussion

The results achieved by each of the constituent recommendation algorithms can be seen in Tables 4.2 and 4.3. There is a clear compromise between accuracy, novelty and diversity of these algorithms. For the MovieLens dataset (Table 4.2), the constituent algorithm that provides the most accurate recommendations is PureSVD50. The constituent algorithm that provides the most novel and diverse recommendations, with an acceptable level of accuracy, is PureSVD150, but its accuracy is much worse than the accuracy obtained by PureSVD50. TopPop provided the worst performance numbers in all criteria used.

On the Last.fm dataset (Table 4.3), the constituent algorithm that provides the most accurate recommendations is WRMF. This is expected, as Last.fm is originally an

| | | Accuracy | | | | | | | | Novelty | Diversity |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Algorithm | R@1 | R@5 | R@10 | R@20 | P@1 | P@5 | P@10 | P@20 | EPC@20 | EILD@20 |
| Const. Algorithms | PSVD50 | **.3859** | .5997 | .6649 | .7178 | **.3859** | .1199 | .0665 | .0359 | .8878 | .9561 |
| | PSVD150 •◇ | .3265 | .5241 | .6055 | .6667 | .3265 | .1048 | .0605 | .0333 | **.8998** | **.9617** |
| | TopPop | .1879 | .4114 | .5198 | .6224 | .1879 | .0823 | .0520 | .0311 | .8508 | .9405 |
| | WRMF † | .3834 | **.6148** | **.7073** | **.7858** | .3834 | **.1230** | **.0707** | **.0393** | .8735 | .9471 |
| | WUKNN | .3272 | .5662 | .6562 | .7340 | .3272 | .1132 | .0656 | .0367 | .8481 | .9352 |
| | UAKNN | .1922 | .3790 | .4712 | .5328 | .1922 | .0758 | .0471 | .0266 | .8605 | .9424 |
| Baselines | STREAM | **.3898** | **.6022** | **.6685** | **.7185** | **.3898** | **.1204** | **.0668** | **.0359** | **.8882** | **.9563** |
| | BC | .2973 | .5346 | .6026 | .6692 | .2973 | .1069 | .0603 | .0335 | .8606 | .9414 |
| | EW | .3017 | .5850 | .6785 | .7595 | .3017 | .1170 | .0679 | .0380 | .8473 | .9363 |
| Our Algorithms | PEH-mean † • ◇ | .4230 | **.6505** | **.7250** | **.7829** | .4230 | **.1301** | **.0725** | **.0391** | .8908 | .9514 |
| | PEH-acc † | **.4323** | .6476 | .7232 | .7819 | **.4323** | .1295 | .0723 | .0391 | .8820 | .9484 |
| | PEH-nov • | .3751 | .5911 | .6659 | .7246 | .3751 | .1182 | .0666 | .0362 | .9219 | .9643 |
| | PEH-div ◇ | .3139 | .5184 | .5943 | .6573 | .3139 | .1037 | .0594 | .0329 | **.9388** | **.9713** |
| | PER-dom † • ◇ | .3866 | .6310 | .7127 | **.7829** | .3866 | .1262 | .0713 | .0388 | .9016 | .9561 |
| | PER-SVM † • ◇ | .3851 | .6062 | .6972 | .7264 | .3851 | .1212 | .0691 | .0363 | .8838 | .9516 |

Table 4.3: Results for Recommendation Algorithms on the Last.fm dataset, with the three objectives (i.e., accuracy, novelty, and diversity). The recommender methods variants are grouped into: (i) constituent algorithms, (ii) multi-objective baselines, and (iii) our proposed Pareto-efficient algorithms. We used the symbols: †, •, ◇ to point out our method and the respective baseline. For each group, the best results for each metric are in bold. Underlined values means that the selected approach and the respective baseline are statistically different (95%).

implicit feedback dataset, to which WRMF is more suitable. Once again, PureSVD150 proved its bias to suggest novel and diverse items, being the best constituent algorithm both in terms of novelty and diversity. In this dataset the compromise between the three objectives is once again illustrated by the fact that there is no algorithm that dominates the others in every objective.

Regarding the performance of the baselines in the MovieLens dataset, STREAM performs worse then PureSVD50 on accuracy and diversity, maintaining the same level of novelty. Borda Count performed poorly on accuracy, reasonably well in terms of novelty and diversity. Equal Weights performed poorly on accuracy, novelty, and diversity. On the Last.fm dataset, STREAM performed slightly worse than WRMF in accuracy, and slightly better in terms of diversity and novelty. Once again, Borda Count performed poorly on accuracy. Finally, Equal Weights performed poorly on accuracy, diversity and novelty.

### 4.3.1 Pareto-Efficient Ranking

Now we turn our attention to the evaluation of our Pareto-efficient ranking algorithm. First, we evaluate the simpler approach, which we call PER-dom (Pareto-Efficient Ranking with most dominant items first). Considering the Movielens dataset, we directly compared PER-dom against two different baselines: PSVD50 and PSVD150, since these algorithms were the best performers in terms of accuracy, novelty and diversity. PER-dom is significantly superior than PSVD50 in the top of the rank, but becomes significantly worse than PSVD50 as $k$ increases. On the other hand, PER-dom greatly outperformed PSVD50 in terms of diversity and novelty. Also, PER-dom is better than PSVD150 in terms of novelty, and it greatly outperforms PSVD150 both in terms of accuracy and diversity. In fact, Per-dom was the best performer in terms of diversity. The more sophisticated approach, which we call PER-SVM (Pareto-Efficient Ranking with SVM), was evaluated using the same procedure as to PER-dom. PER-SVM is slightly superior than PVSD50 in all three objectives considered. Also, PER-SVM is much better than PSVD150 in terms of accuracy and diversity, and slightly better in terms of novelty. In summary, PER-SVM is a good choice for cases where all objectives are simultaneously important: it was not the best performer in any of the objectives, but its performance is close to the best performers in any of the objectives.

A similar trend is observed for the Last.fm dataset. We directly compared PER-dom against two different baselines: WRMF and PSVD150, since these algorithms presented the best numbers in terms of accuracy, novelty and diversity. PER-dom is significantly superior than WRMF, in terms of all objectives considered, and particularly better in terms of novelty. Further, PER-dom is much better than PSVD150 in terms of accuracy, and slightly better in terms of novelty, but it is significantly worse than PSVD150 in terms of diversity. PER-SVM performed similarly to PER-dom, both in terms of accuracy and diversity. Also, PER-SVM greatly outperforms PVSD150 in terms of accuracy, but PSVD150 is significantly better in terms of diversity and novelty. Finally, PER-SVM is slightly better than WRMF in all objectives considered. The same conclusion holds for Last.fm, that both PER-dom and PER-SVM are good choices if all objectives must be maximized simultaneously.

### 4.3.2 Pareto-Efficient Hybridization

Now, with our hybridization algorithm, we could reach any of the individuals in Figures 4.3 and 4.4, which represent the accuracy (in this case, Recall@10) and novelty (EPC@20) of the recommendations in $x$ and $y$ axes, and diversity (EILD@20) with a

color scale. It is clear that there is a compromise between accuracy and the other two objectives: the individuals with the most accurate recommendations provide less novel and diverse lists, and so on. This compromise can be adjusted dynamically with little extra cost, since the cost of reaching these individuals is as low as a linear search (for the individual that maximizes a weighted mean, as described in Section 4.3) over the Pareto frontier individuals' scores. The Pareto frontier consists of 510 individuals in the MovieLens dataset, and of 318 individuals in the Last.fm dataset, so a linear search can be done very quickly. We chose to demonstrate a few of these individuals in Tables 4.2 and 4.3. First, PEH-mean (Pareto-Efficient Hybrid with mean weights) represents the individual that optimizes the mean of the three normalized objectives, assuming each of them are equally important. This would be an option if personalization was not desired, or if the designers of the recommender system do not know which combination of the three objectives would result in higher user satisfaction. However, in a more realistic scenario, the system designer would most likely want to select different individuals for different users. We selected as examples the following individuals, which were found by the process explained in Section 4.3 with the represented associated weighted vectors:

- PEH-acc:[Accuracy:0.70, Novelty:0.30, Diversity:0.00]
- PEH-nov: [Accuracy:0.15, Novelty:0.50, Diversity:0.35]
- PEH-div: [Accuracy:0.10, Novelty:0.35, Diversity:0.55]

These objective weights led to the algorithm weights presented in Table 4.4. It is worth noticing that even though some algorithms are always highly weighted (PSVD50, for example) and others are always weighted negatively (TopPop), there are significant differences between the weights of different individuals, which lead to completely different objective values. It is interesting to notice that weaker algorithms (such as WRMF, which in this dataset is worse than PSVD50 in all three objectives) are still able to play a significant role when the algorithms are combined.

| Individual | PSVD50 | PSVD150 | TopPop | WRMF | WIKNN | WUKNN | UAKNN |
|---|---|---|---|---|---|---|---|
| PEH-mean | 21.60 | 20.19 | -14.91 | 8.83 | 0.36 | 13.92 | -3.10 |
| PEH-acc | 21.55 | 7.80 | -11.10 | 10.20 | 5.47 | 10.86 | -3.98 |
| PEH-nov | 25.95 | 22.43 | -5.19 | 0.04 | -5.07 | 8.18 | -7.48 |
| PEH-div | 25.95 | 23.43 | -26.94 | 1.20 | -5.86 | 16.90 | -1.89 |

Table 4.4: Constituent algorithms' weights for different individuals, Movielens

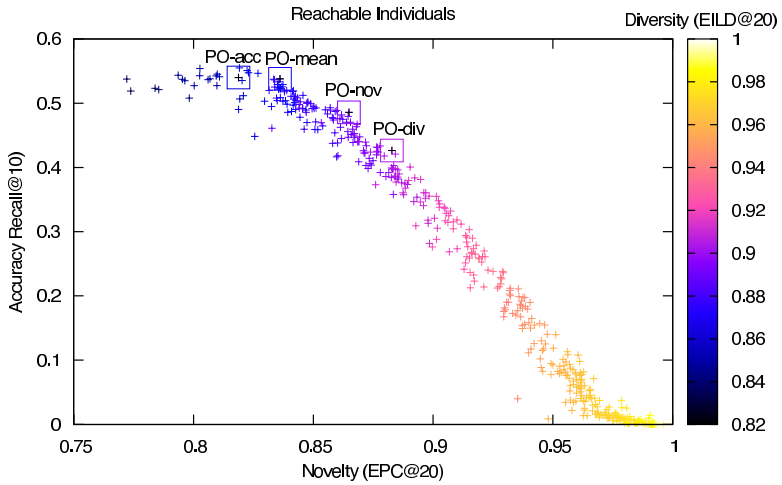


Figure 4.3: Individuals lying in the Pareto frontiers for Movielens.

We compared PEH-acc against PureSVD50, which is the most accurate constituent algorithm. It perform equally well or better than PureSVD on accuracy, but PEH-acc performs better on novelty and worse on diversity. We compared PEH-nov against PureSVD150, which presented the most novel recommendations to the users, with reasonable accuracy. PEH-nov performs better on all three objectives, when compared to PureSVD150 - particularly accuracy and novelty. Finally, we compared PEH-div with PureSVD150, the algorithm with the most diverse recommendations. PEH-div maintains (or slightly improves)the accuracy level, while improving a lot on both novelty and diversity. PEH-mean was an individual that balanced the three objectives, performing much better than PureSVD150, but worse than PureSVD in accuracy, and better than PureSVD50 on novelty and diversity, but worse than PureSVD150. We were able to find individuals in the Pareto Frontier that performed at least as well as the best algorithms in each individual objective, but better on the other objectives. Once again, we could have chosen to compromise more accuracy if we desired even more novelty and diversity, as it is shown in Figure 4.3.

As for the Last.fm dataset, we selected the following individuals:

- PEH-acc: [Accuracy:0.70, Novelty:0.30, Diversity:0.00]
- PEH-nov: [Accuracy:0.15, Novelty:0.85, Diversity:0.00]
- PEH-div: [Accuracy:0.05, Novelty:0.45, Diversity:0.50]

These objective weights led to the algorithm weights presented in Table 4.5. Once again, we notice that different priorities lead to very diverse algorithm weights, and

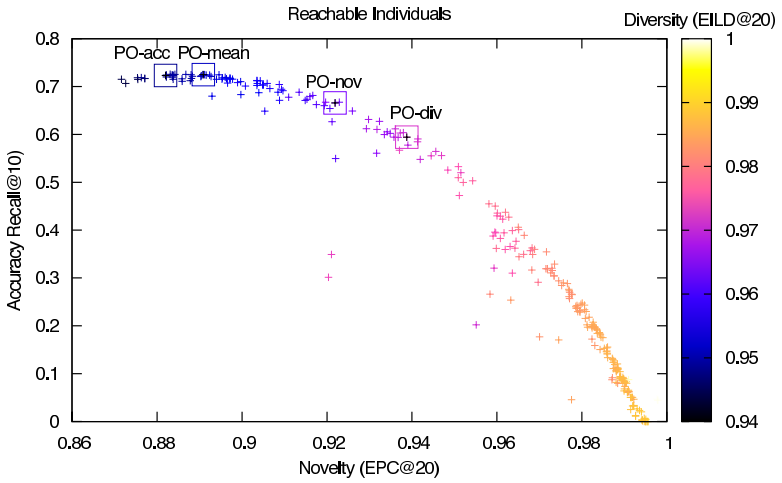


Figure 4.4: Individuals lying in the Pareto frontiers for Last.fm.

that weaker algorithms (such as UAKNN) are able to play an important role when the algorithms are combined.

| Individual | PSVD50 | PSVD150 | TopPop | WRMF | WUKNN | UAKNN |
|---|---|---|---|---|---|---|
| PEH-mean | 26.02 | 24.57 | -10.53 | 23.21 | 2.77 | -7.91 |
| PEH-acc | 26.02 | 22.43 | -6.24 | 24.14 | 4.93 | -7.36 |
| PEH-nov | 27.94 | 26.97 | -9.51 | 13.01 | -5.49 | -8.60 |
| PEH-div | 26.02 | 21.81 | -9.27 | 4.19 | -1.90 | -8.15 |

Table 4.5: Constituent algorithms' weights for different individuals, Last.fm

This time, we compared PEH-acc against WRMF, which is the most accurate constituent algorithm on this dataset. PEH-acc is much more accurate than WRMF, while also improving on novelty and performing almost as well on the diversity level. PEH-nov was compared against PureSVD150, and it performed much better on accuracy and novelty, while losing on the diversity. PEH-div was compared against PureSVD150, and it faired slightly worse on accuracy, while greatly improving on both novelty and diversity. PEH-mean was once again a balanced individual, although this time its accuracy was much better than any of the constituent algorithms. Once again, we were able to find effective individuals in the Pareto frontier, but we could have reached any of the individuals in Figure 4.4 by tweaking the weight value for each objective.

In summary, our proposed algorithms are able to provided significant improvements when compared against other multi-objective algorithms. Specifically, a compar-

ison involving our best performers and the best hybrid baselines, reveals improvements in terms of accuracy (R@1), with gains ranging from 10.4% (on Last.fm) to 10.7% (on MovieLens), in terms of novelty, with gains ranging from 5.7% (Last.fm) to 7.5% (MovieLens), and also in terms of diversity, with gains ranging from 1.6% (Last.fm) to 4.2% (MovieLens).

### 4.3.3 Reproducibility

The datasets we have used in our experiments are freely available, and can be obtained following instructions provided by Miller et al. [2003]; Celma and Herrera [2008]. All constituent algorithms, except PureSVD are implemented in the MyMediaLite package Gantner et al. [2011]. The SVD implementation used for PureSVD50 and PureSVD150 is freely available at `https://github.com/ocelma/python-recsys`. The SVM-Rank implementation used in PER-SVM is freely available at `http://svmlight.joachims.org/svm_rank.html`. The evolutionary algorithm implementation we used to find Pareto-efficient hybrids is available at `http://deap.googlecode.com`.

# Chapter 5

# Conclusions and Future Work

In this work we propose Pareto-efficient algorithms for recommender systems where objectives such as accuracy, novelty and diversity must be maximized simultaneously. We show that existing recommendation algorithms do not perform uniformly well when evaluated in terms of accuracy, novelty and diversity, and thus we propose algorithms that exploit the Pareto efficiency concept in order to combine such recommendation algorithms in a way that a particular objective is maximized without significantly hurting the other objectives.

The Pareto-efficiency concept is exploited in two distinct manners: (i) items are placed in an $n$-dimensional space (i.e., $n$ constituent algorithms) in which the coordinates are the scores assigned to the item by the algorithms. In this way, combining the constituent algorithms means maximizing all objectives simultaneously; (ii) hybrid algorithms (i.e., linear combination of the constituent algorithms) are placed in a 3-dimensional space in which the coordinates are the level of accuracy, novelty and diversity associated with each hybrid. Different hybrids may give emphasis to a particular objective, provided that this will not significantly hurt the other objectives.

Our proposed Pareto-efficient algorithms may be very useful in different scenarios. An obvious scenario is to provide better suggestions to the users, recommending items that are simultaneously accurate, novel and diverse. Another example is the personalization of recommendations according to particular users. For instance, new users may benefit from an algorithm which generates highly ratable items, as they need to establish trust and rapport with the recommender system before taking advantage of the suggestions it offers. The costly part of our Pareto-efficient algorithm is performed entirely offline, and the online cost of choosing items or hybrids in the Pareto frontier is almost negligible, since the Pareto frontier is comprised of few items or hybrids.

We performed highly reproducible experiments on public datasets of implicit

and explicit feedback, using open-source implementations. In our experiments, we demonstrated that the proposed algorithms have either the ability to balance each of the objectives according to the desired compromise, or the ability to maximize all three objectives simultaneously. Finally, we show that the proposed algorithms have obtained results that are competitive with the best algorithms according to each objective and almost always better on the other objectives.

As for future work, the most interesting experiments and extensions to this work would need an online setting. In such a setting, we could measure how a difference in each objective correlates with user satisfaction metrics. Another interesting pursuit would be figuring out how often the models need to be retrained, or proposing online versions of the algorithms listed in this work.

# Bibliography


Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734--749.

Agarwal, D., Chen, B.-C., Elango, P., and Wang, X. (2011). Click shaping to optimize multiple objectives. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 132--140.

Baeza-Yates, R. and Ribeiro-Neto, B. (2011 (second edition)). *Modern information retrieval–The Concepts and Technology behind Search*. Addison-Wesley.

Bao, X., Bergman, L., and Thompson, R. (2009). Stacking recommendation engines with additional meta-features. In *ACM International Conference on Recommender Systems*, pages 109--116.

Basilico, J. and Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *21st International Conference on Machine learning*.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: using social and content-based information in recommendation. In *15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 714--720.

Bell, R., Koren, Y., and Volinsky, C. (2007). Chasing $1,000,000: How we won the netflix progress prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2):4--12.

Bennett, J., Lanning, S., and Netflix, N. (2007). The netflix prize. In *KDD Cup and Workshop in Conjunction with KDD*.

Billsus, D. and Pazzani, M. (2000). User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2):147--180.

Börzsönyi, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In *IEEE International Conference on Data Engineering*, pages 421–430.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331--370.

Cecchini, R., Lorenzetti, C., Maguitman, A., and Brignole, N. (2010). Multiobjective evolutionary algorithms for context-based search. *Journal of the American Society for Information Science and Technology*, 61(6):1258--1274.

Celma, O. and Herrera, P. (2008). A new approach to evaluating novel recommendations. In *ACM International Conference on Recommender Systems*, pages 179--186.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems*, pages 40--48.

Corne, D., Knowles, J., and Oates, M. (2000). The pareto envelope-based selection algorithm for multi-objective optimisation. In *Parallel Problem Solving from Nature*, pages 839–848.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *ACM International Conference on Recommender Systems*, pages 39--46. ACM.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230.

Desrosiers, C. and Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 107–144. Springer.

Dokoohaki, N., Kaleli, C., Polat, H., and Matskin, M. (2010). Achieving optimal privacy in trust-aware social recommender systems. In *International Conference on Social Informatics*, pages 62–79.

Eiben, A. and Smith, J. (2003). *Introduction to evolutionary computing*. Springer Verlag.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2011). Mymedialite: a free recommender system library. In *ACM International Conference on Recommender Systems*, pages 305–308.

Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *ACM International Conference on Recommender Systems*, pages 257--260.

Goldberg, E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning.* Springer Series in Statistics.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5--53.

Holland, J. (1975). *Adaptation in natural and artificial systems.* Number 53. University of Michigan Press.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*, pages 263–272.

Hwang, C. (2010). Genetic algorithms for feature weighting in multi-criteria recommender systems. *Journal of Convergence Information Technology*, 5(8).

Jambor, T. and Wang, J. (2010). Optimizing multiple objectives in collaborative filtering. In *ACM International Conference on Recommender Systems*, pages 55--62.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133--142.

Joachims, T. (2006). Training linear svms in linear time. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226.

Lekakos, G. and Caravelas, P. (2008). A hybrid approach for movie recommendation. *Multimedia tools and applications*, 36(1):55--70.

Lin, X., Yuan, Y., Zhang, Q., and Zhang, Y. (2007). Selecting stars: The k most representative skyline operator. In *IEEE International Conference on Data Engineering*, pages 86–95.

McNee, S., Riedl, J., and Konstan, J. (2006). Accurate is not always good : How accuracy metrics have hurt recommender systems. *Search*, pages 1097--1101.

Michalewicz, Z. (1996). *Genetic algorithms+ data structures*. Springer.

Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. (2003). Movielens unplugged: experiences with an occasionally connected recommender system. In *International Conference on Intelligent User Interfaces*, pages 263--266.

Minchul Jung, M., Jehwan Oh, J., and Eunseok Lee, E. (2008). Genetic recommend generating method with real-time fitness function adaption*. *International Journal of u-and e-Service, Science and Technology*, 1(1):9--16.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc.

Naruchitparames, J., Gunes, M., and Louis, S. (2011). Friend recommendations in social networks using genetic algorithms and network topology. In *IEEE Congress on Evolutionary Computation*, pages 2207–2214.

Pagonis, J. and Clark, A. (2010). Engene: A genetic algorithm classifier for content-based recommender systems that does not require continuous user feedback. In *UK Workshop on Computational Intelligence*, pages 1--6.

Pan, R., Zhou, Y., Cao, B., Liu, N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *IEEE International Conference on Data Mining*, pages 502–511.

Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2003). An optimal and progressive algorithm for skyline queries. In *ACM SIGMOD International Conference on Management of Data*, pages 467–478.

Pazzani, M. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393--408.

Polikar, R. (2006). Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45.

Ribeiro, M. T., Lacerda, A., Moura, E., Hata, I., Veloso, A., and Ziviani, N. (2013). Multi-objective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology*.

Ribeiro, M. T., Lacerda, A., Veloso, A., and Ziviani, N. (2012). Pareto-efficient hybridization for multi-objective recommender systems. In *ACM International Conference on Recommender Systems*, pages 19--26.

Rodriguez, M., Posse, C., and Zhang, E. (2012). Multiple objective optimization in recommender systems. In *ACM International Conference on Recommender Systems*, pages 11--18.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *ACM International Conference on Recommender Systems*, pages 109--116.

Wang, Y. and Witten, I. H. (1997). Inducing model trees for continuous classes. In *9th European Conference on Machine Learning Poster Papers*, pages 128--137.

Zhang, M. and Hurley, N. (2008). Avoiding monotony: improving the diversity of recommendation lists. In *ACM Conference on Recommender Systems*, pages 123--130. ACM.

Ziegler, C., McNee, S., Konstan, J., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22--32. ACM.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. Technical report 103.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257--271.