GUARD: UM ARCABOUÇO PARA RECOMENDAÇÃO BASEADO EM PROGRAMAÇÃO GENÉTICA

ADOLFO PINTO GUIMARÃES

GUARD: UM ARCABOUÇO PARA RECOMENDAÇÃO BASEADO EM PROGRAMAÇÃO GENÉTICA

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: Gisele Lobo Pappa
Coorientador: Nivio Ziviani

Belo Horizonte
Março de 2013

Guimarães, Adolfo Pinto
G963g GUARD: Um Arcabouço para Recomendação Baseado em Programação Genética / Adolfo Pinto Guimarães. — Belo Horizonte, 2013
xviii, 50 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais.Departamento de Ciência da Computação
Orientador: Gisele Lobo Pappa
Coorientador: Nivio Ziviani

1. Computação - Teses. 2. Programação genética (Computação) - Teses. 3. Recuperação da informação - Teses. I. Orientador. II. Coorientador. III. 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

GUARD: Um arcabouço para recomendação baseado em programação genética

ADOLFO PINTO GUIMARÃES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. GISELE LOBO PAPPA - Orientadora
Departamento de Ciência da Computação - UFMG

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

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

PROF. RICARDO BASTOS CAVALCANTE PRUDENCIO
Centro de Informática - UFPE

Belo Horizonte, 22 de março de 2013.

Agradecimentos

Obrigado a minha família, sem a qual eu não teria o suporte necessário, em especial, aos meus pais (João e Fatima) e aos meus irmãos (Alexandre, Marcelo e Eduardo). Agradeço também às minhas cunhadas, sobrinha, tios e primos que sempre torceram por mim.

À minha namorada Chris, companheira de todas as horas. Só a gente sabe o quão foi difícil esse relacionamento à distância. Obrigado pelo amor, pelo apoio, pelos conselhos e por aguentar minhas reclamações nos momentos de stress.

Aos grandes amigos de Aracaju que mesmo distante, sempre estiveram presentes ao longo desses anos.

À minha família temporária em BH: D. Wanda, Eduardo, João e Flávio por me acolher nestes dois anos.

Ao pessoal do LATIN (Aécio, Cristiano, Thales, Wladmir, Aline, Anisio, Sabir, Itamar e Leonardo) que compartilharam comigo grandes experiências acadêmicas e que estiveram presentes no trabalho do dia-a-dia.

De maneira especial, agradeço também a Gisele e ao Nivio que me orientaram para a construção deste trabalho. Tenho certeza que os ensinamentos foram de grande importância para minha formação e será de grande valia para os próximos trabalhos.

Por fim e não menos importante, obrigado a todos os professores e funcionários do DCC/UFMG que contribuíram com os ensinamentos e suporte para realização deste trabalho.

Resumo

Os sistemas de recomendação fornecem sugestões de itens baseados no interesse do usuário. Esses sistemas estão presentes em diversos contextos, tais como comércio eletrônico, máquinas de busca e guias de programação para TV digital. Esta dissertação propõe o GUARD (*A Genetic Unified Approach for Recommendation*), um arcabouço baseado em programação genética criado para gerar funções de *ranking* de itens para sistemas de recomendação. O arcabouço desenvolvido é flexível, e apesar de ter sido implementado para trabalhar com filtragem colaborativa, pode ser facilmente estendido para recomendação baseada em conteúdo ou híbrida. Na filtragem colaborativa, itens são recomendados ao usuário levando em consideração a preferência de usuários com interesses semelhantes aos seus.

A programação genética (PG) é um método baseado nas teorias de evolução e sobrevivência dos indivíduos mais adaptados. A principal motivação para propor um método baseado em PG para gerar funções de *ranking* está em sua flexibilidade para combinar diferentes evidências, além de sua capacidade de lidar com incerteza e ruído nos dados. O GUARD avalia as funções de ranking geradas com base em quatro medidas: precisão, revocação, novidade e diversidade. Essa avaliação baseada em diferentes critérios de qualidade segue duas abordagens distintas de otimização dos objetivos: uma abordagem baseada em Pareto e uma abordagem lexicográfica.

O arcabouço foi aplicado no cenário de recomendação de filmes com as bases Movielens 100k e 1M, e comparados com o PureSVD, algoritmo estado da arte para recomendação de filtragem colaborativa. Para a Movielens 100k, os resultados de precisão e revocação obtidos foram melhores que os do PureSVD, gerando ganho de aproximadamente 7% em número de itens nas primeiras posições do *ranking*. No caso dos indivíduos com melhor novidade e diversidade, apesar de uma pequena perda na precisão, houve um ganho significativo em relação ao PureSVD nesses objetivos. Para a Movilens 1M, os resultados não superaram os do baseline proposto de acordo com as quatro medidas, ficando em média 3% abaixo em número de itens relevantes nas 20 primeiras posições do ranking. Por outro lado, as funções geradas são muito mais simples e eficientes que aquelas geradas pelo PureSVD.

Abstract

Recommender systems suggest new items to the user based on his/her interest. These systems appear in distinct contexts, including e-commerce, search engines, program guides for digital TV.

This work proposes GUARD (A Genetic Unified Approach for Recommendation), a framework based on genetic programming conceived to generate items ranking functions for recommendation. The framework is flexible, and although developed under a colaborative filtering framework can be easily extended to work with content-based or hybrid recommender systems. When working with collaborative filtering, items are recommended to the user based on preferences of similar users in the system.

Genetic Programming is a method based on the theories of evolution and survival of the fittest. The main motivation behind using genetic programming to generate items raking functions is in their flexibility to combine different data evidences and capability of dealing with data uncertainty and noise.

GUARD evaluates the generated ranking functions using four different measures: precision, recall, diversity and novelty. The evaluation, which can be based any combination of these criteria, follows two different approaches for multicriteria optimisation: a Pareto-based and a lexicographical approach.

The framework was tested in the scenario of movies recommendation, using the Movielens 100k and 1M datasets. The results obtained were compared to those generated by PureSVD, the state-of-the-art algorithm for collaborative filtering. Considering the Movielens 100k, the results of precision and recall were superior to those of SVD. The framework can also generate more diverse and novel recommendations, with a small loss in precision. For Movielens 1M, the results are not better than those of PureSVD. The generated ranking functions lose in accuracy for PureSVD but gain in simplicity and efficiency.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A quantidade de informação publicada e compartilhada na Web vem crescendo rapidamente, e a busca e seleção do que é de interesse do usuário é um desafio. O usuário não quer apenas a informação em si, mas informação útil e personalizada. É nesse cenário que aparecem os sistemas de recomendação [Adomavicius & Tuzhilin, 2005], que fornecem sugestões de itens baseadas no interesse do usuário, e estão presentes em diversos contextos, incluindo comércio eletrônico, máquinas de busca e guias de programação para TV digital.

O primeiro passo para recomendar um item a um usuário é definir seus interesses. O interesse do usuário pode ser representado de diversas maneiras, dependendo do tipo de recomendação utilizada [Ricci et al., 2011]. Os três principais tipos de sistemas de recomendação são: (1) sistemas baseados em filtragem colaborativa, (2) sistemas baseados em conteúdo e (3) sistemas híbridos. Na filtragem colaborativa itens são recomendados ao usuário levando em consideração a preferência de usuários com interesses semelhantes aos seus (i.e., que se interessam pelos mesmos itens). Já a abordagem baseada em conteúdo recomenda itens que possuam conteúdo semelhante àqueles já avaliados pelo usuário. Nos sistemas híbridos são utilizadas técnicas que combinam os dois primeiros tipos [Adomavicius & Tuzhilin, 2005]. Independente do tipo de recomendação utilizada, sistemas de recomendação buscam fornecer recomendações úteis e capazes de atender a necessidade do usuário.

Este trabalho propõe o GUARD (*A Genetic Unified Approach for Recommendation*), um arcabouço para recomendação baseado em programação genética (PG). PG é um método evolutivo disseminado por Koza [1992] e baseado nas teorias de evolução e sobrevivência dos indivíduos mais adaptados. Ela trabalha com uma população de indivíduos que evolui por um número pré-definido de gerações. A evolução acontece através de operações de cruzamento e mutação, e cada indivíduo na população

representa uma solução candidata para o problema em questão.

A principal motivação para propor um método baseado em PG para gerar funções de *ranking* seguindo quaisquer das três abordagens citadas acima está em sua flexibilidade para combinar diferentes evidências, além de sua capacidade de lidar com incerteza e ruído nos dados [Bäck et al., 2000]. A programação genética já foi utilizada com sucesso em tarefas de aprendizado para, por exemplo, obter *rankings* para o conjunto resposta de uma máquina de busca [Fan et al., 2005] ou associar propagandas a páginas Web [Lacerda et al., 2006]. Porém, não encontramos nenhum trabalho que utilizou PG para recomendação de itens. Isso nos permite contribuir para um domínio ainda não explorado na literatura.

Embora o GUARD tenha sido preparado para trabalhar com os três tipos de recomendação supracitados, esta dissertação foca em métodos de filtragem colaborativa. A função de *ranking* gerada pelo GUARD é então uma combinação de (partes de) medidas comumente utilizadas por métodos de filtragem colaborativa, tais como o valor médio do *rating* de um item, sua popularidade, sua distância aos k vizinhos mais próximos, etc. A qualidade da função é avaliada utilizando quatro diferentes tipos de medidas: precisão, revocação, novidade e diversidade.

Embora o principal objetivo de um sistema de recomendação seja maximizar a acurácia das recomendações, ou seja, garantir boa precisão ao prever o interesse do usuário, outras medidas também devem ser consideradas [Herlocker et al., 2004]. Isso acontece porque as necessidades de um usuário podem variar de acordo com seu perfil. Sabe-se que a idade do usuário no sistema, por exemplo, pode ser fator determinante em relação ao que ele espera como resposta. Para novos usuários é importante garantir recomendações precisas (de alta acurácia) para conquistar sua confiança. Dificilmente o usuário retorna ao sistema se as primeiras recomendações não forem de seu interesse. No entanto, com o passar do tempo é interessante aumentar o grau de novidade e diversidade nas recomendações, mesmo que isso implique em uma perda de precisão [McNee et al., 2006]. Essas duas características podem surpreender o usuário, e fazer com que ele não perca o interesse pelo sistema. Maximizar esses três objetivos (acurácia, novidade e diversidade) não é uma tarefa simples e tem sido objeto de pesquisa de diversos trabalhos (por exemplo, [Ribeiro et al., 2012]), incluindo essa dissertação.

1.1 Objetivos

O principal objetivo deste trabalho é propor um arcabouço genérico para sistemas de recomendação baseado em programação genética. O arcabouço será então instanciado

utilizando um método de filtragem colaborativa capaz de gerar funções de *ranking* precisas. Além da precisão, o arcabouço também leva em consideração outras medidas de interesse do usuário nesse tipo de sistema, tais como novidade e diversidade.

Sendo assim, os principais objetivos específicos deste projeto são:

- Realizar um estudo detalhado de todos os métodos utilizados para filtragem colaborativa;
- Estudar a eficácia de algoritmos de PG na tarefa de recomendação;
- Avaliar as funções de *ranking* criadas utilizando não apenas a precisão, mas também a revocação, novidade e diversidade;
- Investigar se a agregação de *rankings* com ênfase nas diferentes medidas acima melhora a qualidade da recomendação.

1.2 Trabalhos Relacionados

A programação genética é conhecida como um método independente de aplicação, uma vez que o algoritmo a ser executado independe do problema, sendo apenas as definições da representação do indivíduo, *fitness* e, em alguns casos, operadores genéticos, diferentes para cada problema. Por esse motivo, PG já foi aplicada a diversos problemas nas mais diferentes áreas [Poli et al., 2008], incluindo recuperação de informação.

Em [Fan et al., 2005], um método de PG foi utilizado na busca por funções de *ranking* de páginas Web. O objetivo do trabalho era combinar evidências tradicionais dos métodos de recuperação de informação para gerar funções de *ranking* capazes de ordenar um conjunto de páginas de acordo com o grau de relevância para a busca do usuário. O trabalho agrupa informações de conteúdo e estruturais dos documentos. O autor destaca a flexibilidade da PG que permite que seu método seja aplicado a diferentes contextos de busca na Web. As funções encontradas também geram resultados melhores que os baselines propostos. [Yeh et al., 2007] também aplica PG para busca na Web, mas amplia o conjunto de evidências utilizadas por [Fan et al., 2005].

Um outro exemplo de trabalho que gera *rankings* em recuperação de informação é [Lacerda et al., 2006]. Os autores utilizaram PG para aprender a associar propagandas a páginas Web. O trabalho propôs um arcabouço baseado em PG capaz de combinar diferentes evidências referentes a propagandas e à página, a fim de gerar funções que associam propagandas a páginas garantindo a relevância das mesmas. A melhor função encontrada garante um ganho de 61% se comparado com os baselines propostos.

PG também tem sido usada na área de recuperação de imagens baseada no conteúdo. Em [Torres et al., 2009], os autores propõe um arcabouço baseando em PG capaz de gerar uma função que agrupa diferentes descritores de imagens de forma eficiente. O trabalho também destaca a flexibilidade do uso de PG e a possibilidade de aplicação em outros contextos. O trabalho gera resultados superiores aos baselines propostos.

Por fim, [Anand & Bharadwaj, 2010] é o trabalho mais próximo ao proposto aqui. Nesse caso, PG é utilizada para encontrar funções que conseguem extrair da melhor maneira a tendência que o usuário tem em avaliar um item. A finalidade é evitar problemas como usuários que sempre avaliam bem ou mal um item. Uma vez que essa tendência é calculada, os valores são utilizados para calcular a similaridade entre usuários para a recomendação por filtragem colaborativa. Observe que essa proposta tem um propósito bem diferente deste trabalho, que propõe PG para encontrar funções que retornem a preferência final do usuário para um item e, consequentemente, um *ranking* de itens relevantes.

1.3 Contribuições

De acordo com os objetivos definidos na seção anterior, a principal contribuição deste trabalho é o GUARD, um arcabouço eficaz e flexível para recomendação em diferentes contextos. O arcabouço poderá ser facilmente estendido para trabalhar com outros tipos de recomendação, e pode ser configurado para trabalhar com quaisquer combinações das quatro medidas consideradas relevantes em sistemas de recomendação: precisão, revocação, novidade e diversidade.

Além disso, a execução do arcabouço gerou um conjunto de funções de *ranking* baseadas na abordagem colaborativa que mostram que, no geral, existe um conjunto simples e pequeno de medidas capazes de gerar recomendações mais precisas que as geradas por algoritmos estado da arte, tais como o PureSVD [Cremonesi et al., 2010].

1.4 Organização

Este trabalho está organizado da seguinte forma: no Capítulo 2 introduzimos os principais conceitos relacionados a sistemas de recomendação e programação genética. O Capítulo 3 descreve o GUARD, um arcabouço para recomendação baseado em PG, detalhando seus principais componentes. No Capítulo 4 são descritas as bases de dados utilizadas, a metodologia experimental e os resultados obtidos. Por fim, o Capítulo 5 traz conclusões e apresenta possíveis direções de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Esta dissertação aborda duas áreas distintas de pesquisa: (1) Sistemas de Recomendação por Filtragem Colaborativa e (2) Programação Genética. Essa seção descreve os principais conceitos e trabalhos relacionados a cada uma delas.

2.1 Sistemas de Recomendação por Filtragem Colaborativa

O problema de recomendação pode ser definido da seguinte forma: seja $U = \{u_1, u_2, u_3, ..., u_m\}$ e $I = \{i_1, i_2, i_3, ..., i_n\}$ o conjunto de todos os usuários e itens de um serviço, respectivamente. Cada usuário u é associado a um subconjunto $I_u \subseteq I$ de itens avaliados por u. Da mesma forma, cada item i é associado a um subconjunto $U_i \subseteq U$ de usuários que avaliaram o item i. As avaliações dos usuários são armazenadas na matriz V onde cada elemento v_{ui} representa o valor da avaliação dado pelo usuário u ao item i. As avaliações são normalmente valores inteiros dentro de um certo intervalo (ex: 1-5, onde 1 é ruim e 5 é ótimo) [Cacheda et al., 2011]. O sistema de recomendação pode ser implementado com dois objetivos diferentes: no primeiro, o objetivo é prever um valor p_{ui} que mede o grau de preferência do usuário $u \in U$ pelo item $i \in I$. No segundo, o objetivo é retornar uma lista ordenada de itens relevantes recomendados [Cacheda et al., 2011], obtida ordenando o conjunto de itens de acordo com o p_{ui} estimado pelo algoritmo.

Como já mencionado, sistemas de recomendação são classificados em três tipos principais: (1) filtragem colaborativa, (2) sistemas baseados em conteúdo e (3) sistemas híbridos [Adomavicius & Tuzhilin, 2005]. A Figura 2.1 mostra uma taxonomia detalhando diferentes abordagens segundo esses tipos de sistemas de recomendação.

Esse trabalho foca em algoritmos de filtragem colaborativa (destacados na Figura 2.1), onde itens são recomendados a um usuário levando em consideração a preferência de usuários com interesses semelhantes. Esse interesse é representado por um vetor de avaliações (*ratings*) atribuídas a um conjunto de itens. O sistema de filtragem colaborativa utiliza essas informações para identificar similaridades entre usuários ou itens, e prover a recomendação baseado nessas similaridades. Dependendo de como a matriz de avaliações V (usuário×item) é processada, os algoritmos são classificados em dois grupos: algoritmos baseados em memória (*memory-based*) ou algoritmos baseados em modelos (*model-based*), como detalhado nas próximas seções.

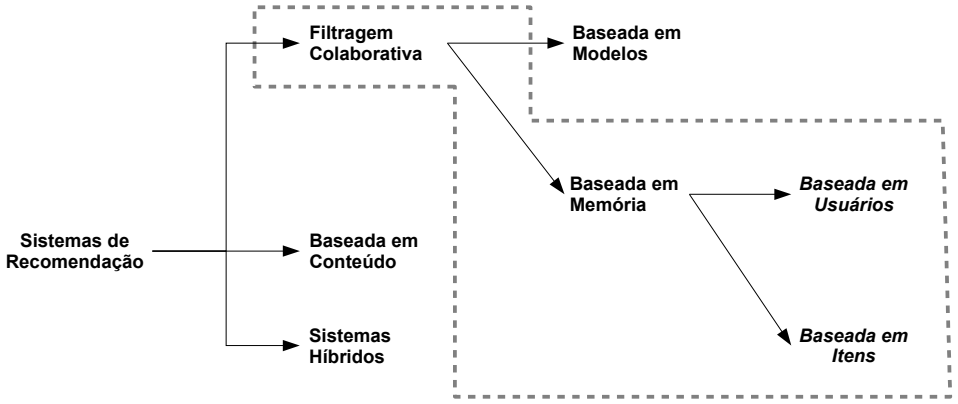


Figura 2.1: Taxonomia dos sistemas de recomendação

2.1.1 Algoritmos Baseados em Memória

Algoritmos baseados em memória utilizam a matriz completa V de usuário×item para gerar recomendações, e uma função de similaridade para encontrar usuários ou itens semelhantes [Cacheda et al., 2011]. Esse conjunto de algoritmos pode ser baseado em usuário (*user-based*) ou baseado em item (*item-based*), dependendo se a função utilizará os usuários mais próximos ou os itens mais similares durante a recomendação.

O uso do modelo baseado em usuários ou itens depende principalmente das características da base em questão. Em sistemas em que o número de itens é muito maior do que o número de usuários, os métodos baseado em itens tendem a ter melhor acurácia do que aqueles baseados em usuários. No entanto, sistemas que utilizam a similaridade de itens normalmente sofrem com o problema da diversidade da recomendação, já que tem maior tendência de recomendar itens parecidos com aqueles que o usuário já viu [Burke, 2002]. Os algoritmos baseado em usuários são menos sensíveis a esse problema.

Independente do tipo de modelo construído, a tarefa de recomendação baseada em memória pode ser dividida em três fases: (1) Cálculo de similaridade; (2) Seleção dos vizinhos mais próximos dos usuários ou itens e (3) Cálculo do *rating* usuário-item.

Cálculo da similaridade : O cálculo da similaridade é feito entre os usuários ou itens, a depender de qual algoritmo estamos usando: baseado em usuários ou baseado em itens. No caso do baseado em usuário a similaridade é calculada entre o usuário para o qual se deseja prover a recomendação e o restante dos usuários da base. Já no baseado em itens a similaridade é calculada entre o item que se deseja recomendar e os demais itens da base.

Dentre as principais técnicas utilizadas para a similaridade de usuários podemos destacar a distância do cosseno [Breese et al., 1998], que mede a distância entre os vetores de avaliações dos usuários e é dada por:

$$s(a,u) = \sum_{j \in I} \frac{v_{aj}}{\sqrt{\sum_{k \in I_a} v_{ak}^2}} \frac{v_{uj}}{\sqrt{\sum_{k \in I_u} v_{uk}^2}} \tag{2.1}$$

onde $a \in U$ é o usuário para qual se deseja prover a recomendação, $u \in U$ é um usuário da base.

Outra medida de similaridade bastante utilizada é o coeficiente de correlação de Pearson [Resnick et al., 1994], que leva em consideração a diferença entre a nota dada pelo usuário e a média de avaliações do item, tal como:

$$s(a,u) = \frac{\sum_{i \in I_a \cap I_u} (v_{ai} - \overline{v_a})(v_{ui} - \overline{v_u})}{\sqrt{\sum_{i \in I_a \cap I_u} (v_{ai} - \overline{v_a})^2 \sum_{i \in I_a \cap I_u} (v_{ui} - \overline{v_u})^2}} \tag{2.2}$$

onde $\overline{v_u}$ e $\overline{v_i}$ representam a média das avaliações do usuário u e do item i, respectivamente. A medida de Pearson com restrições [Shardanand & Maes, 1995], por sua vez, é uma variante da medida anterior, só que ao invés de utilizar a média da avaliação é usado um valor central da escala de valores de avaliações permitida. Por exemplo, em uma escala de avaliação que vai de 1 a 5, o central é 3. A similaridade é calculada por:

$$s(a,u) = \frac{\sum_{i \in I_a \cap I_u} (v_{ai} - 3)(v_{ui} - 3)}{\sqrt{\sum_{i \in I_a \cap I_u} (v_{ai} - 3)^2 \sum_{i \in I_a \cap I_u} (v_{ui} - 3)^2}} \tag{2.3}$$

As três equações anteriores calculam a similaridade de usuários. Para o caso da similaridade de itens, ela pode ser calculada por [Sarwar et al., 2001]:

$$s(i,j) = \frac{\sum_{u \in U} (v_{ui} - \overline{v_u})(v_{uj} - \overline{v_u})}{\sqrt{\sum_{u \in U} (v_{ui} - \overline{v_u})^2 \sum_{u \in U} (v_{uj} - \overline{v_u})^2}} \tag{2.4}$$

Seleção dos vizinhos : O número de vizinhos considerados durante o cálculo de distância pode ser determinado utilizando um limiar, onde são considerados apenas usuários ou itens que estejam acima de um determinado grau de similaridade, ou selecionando os N usuários/itens mais similares. Herlocker et al. [2002] mostra que para sistemas reais valores de N entre 20 e 50 são suficientes para garantir a qualidade da recomendação.

Cálculo do *rating* usuário-item : Nesta fase, os algoritmos utilizam as informações de similaridades de usuários ou itens e as avaliações dos N vizinhos mais semelhantes para estimar o grau de preferência do usuário por um item.

Nos algoritmos baseados em usuários essa preferência está diretamente relacionada a preferência dos N-usuários mais semelhantes. Uma forma de calcular essa preferência é utilizando uma técnica de normalização [Herlocker et al., 2002]. Nesse caso, os valores dos *ratings* são determinados com base na média dos *ratings* dados por um usuário. Essa média é ajustada de acordo com a similaridade deste usuário com seus vizinhos, como descrito em:

$$p_{ai} = \overline{v_a} + \frac{\sum_{u \in Neigh_a} [(v_{ui} - \overline{v_u}) s(a,u)]}{\sum_{u \in Neigh_a} s(a,u)} \tag{2.5}$$

Já para a recomendação baseada em itens, a preferência de usuário por um item está diretamente relacionada aos N-itens mais similares àquele que está sendo recomendado e pode ser calculada por [Sarwar et al., 2001]:

$$p_{aj} = \frac{\sum_i (s(j,i) v_{ai})}{\sum_i |s(j,i)|} \tag{2.6}$$

Se o objetivo final da recomendação for gerar uma lista de itens recomendados, os valores retornados pelo último passo são ordenados, e os n itens no topo do *ranking* mostrados ao usuário.

2.1.2 Algoritmos Baseados em Modelos

Enquanto algoritmos baseados em memória utilizam uma função de similaridade para recomendar itens aos usuários, algoritmos baseados em modelos geram modelos mais sofisticados a partir da matriz de dados disponível. Esses algoritmos vêm ganhando destaque na literatura dados os altos valores de acurácia obtidos por eles [Koren et al., 2009]. Técnicas como pLSA (*Probabilistic Latent Semantic Analysis*) [Hofmann, 2004], redes neurais [Salakhutdinov et al., 2007] e modelos construídos a partir da fatorização da matriz de avaliação usuário×itens (SVD - *Singular Value Decomposition*) são as mais utilizadas.

Dentro dos diversos modelos já utilizados para recomendação [Burke, 2002], esse trabalho foca no SVD, que representa hoje o estado da arte em métodos de recomendação. A aplicação direta do SVD para recomendação não é uma tarefa simples, uma vez que o SVD tende a não gerar bons resultados quando há uma grande esparsidade dos dados [Ricci et al., 2011]. Como solução para tal problema, a literatura apresenta extensões do SVD como o SVD++ [Koren, 2008] e o PureSVD [Cremonesi et al., 2010].

Tanto o SVD quanto o SVD++ tem como finalidade encontrar um valor p_{ui} que mede o grau de preferência do usuário $u \in U$ pelo item $i \in I$. No entanto, como mencionado, esse não é o único objetivo dos sistemas de recomendação. Sistemas que recomendam apenas uma lista de itens sem tentar prever a nota vem se tornando cada vez mais populares e alvo de pesquisa. O PureSVD é considerado o estado da arte para essa tarefa, e portanto será utilizado para comparações nesse trabalho. A seguir descrevemos com mais detalhes o SVD e o PureSVD.

SVD

Os modelos baseados em fatoração de matrizes mapeiam tanto usuários quanto itens para um espaço de fatores latentes de dimensão f. Esse espaço representa as relações entre usuários e itens modelados por meio de um produto interno. O método SVD proposto por [Koren, 2008] fatoriza a matriz de avaliações usuário×item em dois fatores: (1) cada item i é associado com um vetor $q_i \in \Re^f$ e (2) cada usuário u é associado com um vetor $p_u \in \Re^f$. O produto entre esses fatores procura extrair o interesse do usuário em relação aos itens. Sendo assim, é possível determinar a preferência do usuário para um item de acordo com:

$$p_{ui} = b_{ui} + q_i^T p_u \tag{2.7}$$

onde b_{ui} representa a tendência de um usuário em atribuir uma nota alta em relação

aos outros usuários e a tendência que um item tem em receber uma nota alta e é dado por $b_{ui} = \mu + b_u + b_i$ sendo μ a média de todos os *ratings* da base. Os valores de b_u e b_i estimam a tendência do usuário e do item, respectivamente, em relação a média total dos *ratings* em uma determinada base de dados. Os parâmetros do modelo são determinados por meio de uma minimização, tal que:

$$min_{b_*,q_*,p_*} = \sum_{(u,i)\in V} (p_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{2.8}$$

onde λ é um fator de regularização que pode ser determinado por validação cruzada, μ é a média de todas as avaliações da base de dados e κ é o conjunto de avaliações (u, i) conhecidas.

Para cada *rating* v_{ui} da base de treino é calculado um valor p_{ui} que corresponde a uma nota que o algoritmo estima que o usuário u daria ao item i. Dessa forma, é associado a esses valores um erro que corresponde a diferença entre o valor real e o valor encontrado p_{ui}. Esse valor é conhecido como erro de previsão, e é representado por e_{ui}. Os parâmetros são então atualizados a medida que procura-se reduzir o erro. A sequência abaixo mostra como atualizar os parâmetros a cada iteração:

- $b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)$
- $b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)$
- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

onde γ determina a taxa de aprendizagem. Neste trabalho, vamos utilizar as métricas de b_{ui}, b_u e b_i no arcabouço proposto. No entanto, esses valores não serão calculados por meio da minimização e sim por um método proposto por [Koren, 2010] que calcula b_u e b_i de forma independente de acordo com:

$$b_i = \frac{\sum_{u:(u,i)\in V} (v_{ui} - \mu)}{\lambda + |u|(u,i) \in V|} \tag{2.9}$$

$$b_u = \frac{\sum_{i:(u,i)\in V} (v_{ui} - \mu - b_i)}{\lambda + |i|(u,i) \in V|} \tag{2.10}$$

Justamente por esse método ser mais simples que a forma de minimização descrita anteriormente, ele faz com que o SVD gere resultados de acurácia piores que aqueles obtidos no processo de minimização inicialmente apresentado.

PureSVD

Diferente do SVD, que é utilizado na tarefa de predizer o *rating* do usuário para um item, o PureSVD visa encontrar um valor que mede a preferência do usuário. Esse valor é utilizado para gerar um ranking de itens ordenados pela relevância para o usuário. Essa diferença para o método anterior permite que o SVD seja aplicado sem sofrer com os efeitos da esparsidade da matriz já que podemos preencher os valores desconhecidos por um valor padrão, como o zero, por exemplo. Isso é possível porque os desvios nos valores causados por esse preenchimento não alteram a posição dos itens no ranking, o que permite aplicar SVD sem a necessidade de minimização de parâmetros mostrados na seção anterior. Sendo assim podemos definir a preferência do usuário por um item como:

$$p_{ui} = r_u \cdot Q \cdot q_i^T \tag{2.11}$$

onde r_u é a linha u da matriz de avaliação e Q os valores singulares extraídos da aplicação do SVD. Dessa forma, a principal vantagem do PureSVD é a facilidade de implementação já que não se tem a necessidade de calibrar uma série de parâmetros como nos métodos anteriores.

2.2 Programação Genética

PG [Koza, 1992] é um método evolucionário baseado nas teorias de Darwin e sobrevivência do indivíduo mais adaptado. Ele é um método poderoso devido a sua flexibilidade de representação de soluções, mecanismos de busca global e tolerância a ruído [Bäck et al., 2000]. Os algoritmos de programação genética são compostos por uma população inicial de indivíduos, onde cada indivíduo representa uma solução potencial para o problema atacado. A Figura 2.2 ilustra o funcionamento de um algoritmo de programação genética, e ressalta seus principais componentes, discutidos a seguir.

Inicialmente, um conjunto de indivíduos é criado a partir de dois conjuntos de dados, denominados terminais e operadores. Os terminais são compostos por variáveis e constantes relevantes ao problema. Os operadores são conjunto de funções específicas que, juntamente com os terminais, formam uma solução para o problema. A população inicial é composta por um conjunto de indivíduos criados aleatoriamente a partir do

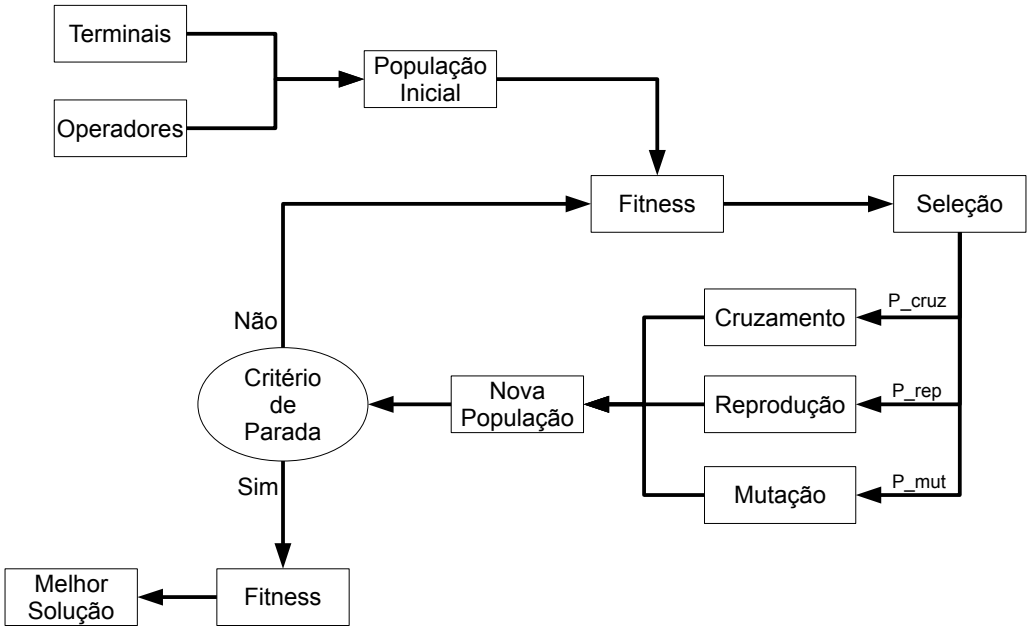


Figura 2.2: Diagrama de um algoritmo de PG

conjunto de terminais e operadores. Normalmente, esses indivíduos são representados por meio de árvores limitadas por uma altura máxima pré-definida.

A população inicial é inicializada de forma aleatória seguindo três métodos tradicionais de inicialização: (1) *Grow*, (2) *Full* e (3) *Ramped half-and-half* [Poli et al., 2008], que diferem entre si pela forma como os componentes (terminais e operadores) são selecionados para a formação do indivíduo.

Quando o método *grow* é utilizado, os indivíduos são formados escolhendo componentes tanto do conjunto de terminais quanto de operadores. Nessa abordagem são geradas árvores irregulares, mas limitadas por uma altura máxima. Já no segundo método (*full*), os indivíduos são formados selecionando componentes apenas do conjunto de operadores. Ao se atingir a altura máxima, escolhem-se os componentes do conjuntos de terminais. Essa abordagem gera indivíduos representados por árvores mais balanceadas. O último método, *Ramped half-and-half*, é uma combinação das duas primeiras técnicas, onde metade da população é inicializada usando *Grow* e a outra metade *Full*. Desta forma, procura-se atingir uma maior diversidade da população inicial.

Em um processo evolucionário os melhores indivíduos tem uma probabilidade

maior de serem mantidos para as próximas gerações, enquanto os demais são descartados ao longo das gerações. Essa seleção é feita de acordo com a *fitness* do indivíduo, que mede o quanto um indivíduo é bom ou ruim para solução do problema em questão. Essa avaliação pode ser feita de acordo com uma ou mais medidas de qualidade.

Problemas que tentam melhorar a qualidade de várias medidas de qualidade simultaneamente, onde essas medidas são normalmente conflitantes entre si (e.g., melhorar qualidade enquanto se diminui o custo), são denominados problemas de otimização multiobjetivo. A literatura define uma série de abordagens para lidar com problemas de otimização multiobjetiva [Deb, 2001] dos quais destacamos três:

Abordagem baseada em pesos: A qualidade de um indivíduo é representada por um único valor escalar. Esse valor é calculado por meio de uma combinação linear que atribui pesos a cada um dos objetivos [Bäck et al., 2000].

Abordagem lexicográfica: Nessa abordagem é definida uma ordem de prioridade para cada um dos objetivos de acordo com sua importância para o problema. Ao comparar os indivíduos, leva-se inicialmente em consideração o objetivo de maior prioridade. Em caso de um empate, o objetivo com prioridade seguinte é avaliado, e assim por diante até o menor nível de prioridade [Fourman, 1985].

Abordagem usando dominância de Pareto: Os indivíduos são comparados utilizando o conceito de dominância de Pareto. De acordo com esse conceito, dado um conjunto de objetivos, uma solução (indivíduo) domina outra se a primeira não é pior que a segunda em todos os objetivos, e, existe pelo menos um objetivo no qual ela é melhor [Poli et al., 2008].

Existe uma série de algoritmos evolucionários propostos para resolver o problema de otimização multiobjetivo. Dentre os principais se destacam o SPEA2 [Zitzler et al., 2001] e NSGA2 [Deb, 2001]. Neste trabalho esses algoritmos não foram considerados, uma vez que modificamos apenas a forma de seleção dos indivíduos para considerar mais de um objetivo. Porém, nada impede que as principais características dessas técnicas (tais como métodos para manter a diversidade de indivíduos na população) sejam futuramente incorporadas ao arcabouço proposto.

Calculada a *fitness* dos indivíduos, inicia-se a etapa de seleção. Dentre as técnicas de seleção existentes, a mais utilizada é a seleção por torneio [Banzhaf et al., 1997]. Neste tipo de seleção, um conjunto K de indivíduos é selecionado aleatoriamente da população. Quando a avaliação do indivíduo é feita utilizando apenas uma medida (um objetivo ou abordagem multiobjetiva baseada em pesos), o indivíduo que possui

melhor *fitness* é considerado vencedor do torneio. No entanto, caso os indivíduos sejam avaliados considerando mais de um objetivo, outras estratégias devem ser utilizadas. No caso da abordagem lexicográfica, os objetivos da lista de prioridades devem ser avaliados um a um. No caso da abordagem baseada em dominância de Pareto, as relações de dominância entre os indivíduos selecionados deve ser considerada. Detalhes de como essas abordagens estão presentes no arcabouço proposto estão no Capítulo 3.

Os indivíduos escolhidos durante o processo de seleção são modificados por meio de três operações principais: (1) cruzamento, (2) mutação e (3) reprodução. O cruzamento envolve dois indivíduos, e cria dois novos descendentes, compostos por sub-expressões (subárvores) de seus pais. A mutação envolve um indivíduo, e substitui uma sub-expressão escolhida aleatoriamente no indivíduo por uma nova, gerando um novo indivíduo. A reprodução seleciona um indivíduo para ser replicado para as gerações seguintes. Ao final das operações obtém-se uma nova população.

O processo descrito acima é repetido até que um critério de parada seja alcançado (normalmente um número pré-definido de gerações). Caso o critério de parada seja satisfeito, o algoritmo calcula a *fitness* nos indivíduos da nova população e retorna como solução do problema aquele de melhor *fitness*.

Capítulo 3

GUARD: Um Sistema de Recomendação Baseado em Programação Genética

Esta seção descreve o GUARD (*A Genetic Unified Approach for Recommendation*), um arcabouço que modela um algoritmo de programação genética para gerar funções para recomendação de itens a usuários. O arcabouço abrange a abordagem de filtragem colaborativa baseada em memória, embora tenha sido implementado para que outros tipos de recomendação, tais como baseado em conteúdo e híbrido, possam ser abordados. O arcabouço recebe como entrada uma matriz M_{ui}, onde u representa o número de usuários e i o número de itens disponíveis. A partir dessa matriz, o algoritmo gera uma função de *ranking*, capaz de ordenar itens a serem recomendados de acordo com a sua relevância para um usuário. As próximas seções discutem em detalhe o arcabouço proposto, iniciando com uma descrição geral de sua arquitetura e dos principais componentes modelados.

3.1 Descrição Geral

Para facilitar a inclusão de novos componentes ao arcabouço, o GUARD foi dividido em seis módulos principais, responsáveis pelas seguintes tarefas: (1) Extração de características; (2) Definição dos terminais e funções da PG; (3) Cálculo dos valores dos terminais da PG; (4) Evolução da população; (5) Escolha da melhor solução e (6) Recomendação. O diagrama da Figura 1 mostra como cada uma das etapas se relacionam e interagem entre si. Note que as duas etapas mostradas em azul requerem um esforço

do usuário, como será descrito a seguir.

A primeira etapa, extração de características, consiste em selecionar na base de dados as informações que serão utilizadas na recomendação. Essa etapa é especialmente útil porque a recomendação colaborativa, tratada neste trabalho, utiliza como entrada informações diferentes daquelas necessárias na recomendação baseada em conteúdo, por exemplo. No caso da filtragem colaborativa, extrai-se da base os itens avaliados por cada usuário.

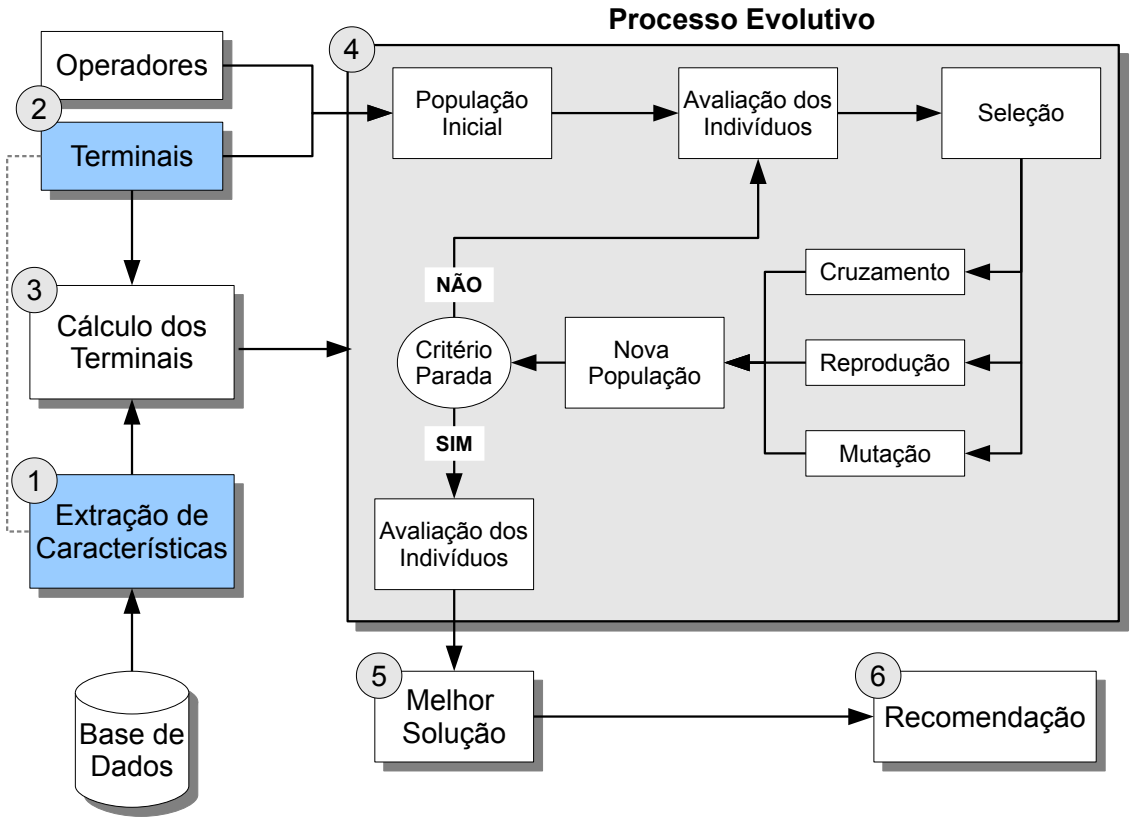


Figura 3.1: GUARD: Framework baseado em Programação Genética para Sistemas de Recomendação

A segunda define os terminais e funções que vão gerar a população inicial. Identificar esses componentes é uma das primeiras tarefas dentro do arcabouço, uma vez que eles são componentes essenciais para execução da fase 4, que é a evolução do algoritmo de programação genética. A fase 2 está diretamente ligada a extração de características, uma vez que os terminais dependem do que foi extraído da base de dados. Após a definição dos terminais, a etapa seguinte pré-calcula seus valores utilizando as informações extraídas da base de dados. O pré-cálculo é importante, já que evita que os terminais sejam recalculados durante a evolução, reduzindo o tempo de execução do processo evolutivo.

A quarta etapa consiste no algoritmo de programação genética em si. Dada uma população inicial, esta é avaliada e seus indivíduos são selecionados e modificados de acordo com as operações de mutação, cruzamento e reprodução. Essas operações geram uma nova população, que é avaliada e modificada até atingir um critério de parada. Uma vez atingido o critério, deve-se selecionar a melhor solução. Essa solução é então usada para recomendar itens ao usuário.

As próximas seções descrevem em detalhes os componentes essenciais para o sucesso da fase quatro: a representação do indivíduo, função de avaliação, seleção, operadores e como selecionamos a função que será utilizada para o usuário durante a fase de recomendação.

3.2 Representação do Indivíduo

Uma das representações mais comuns do indivíduo em programação genética é a representação por árvores. Nesse tipo de representação, cada indivíduo é formado por um conjunto de nós que representam os operadores e os terminais. Os operadores aparecem nos nós internos da árvore, e os terminais nas folhas. A Figura 3.2 exemplifica o indivíduo $T1/T2 \times T3$ com os operadores ($\times$) e ($/$) e os terminais T1, T2 e T3.

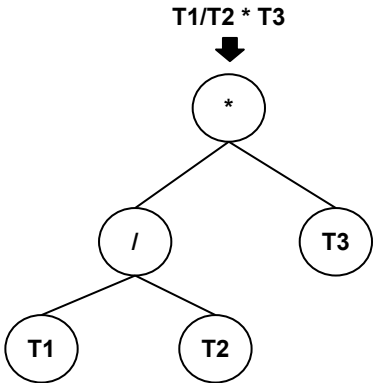


Figura 3.2: Representação em árvore de um indivíduo

Para este trabalho, os operadores poderão ser representados por sete operações: soma ($+$), subtração (-), multiplicação ($\times$), divisão ($/$), potência (*pow*), raiz quadrada (*sqrt*) e logaritmo (*log*). Essas funções foram escolhidas por estarem presentes em grande parte das funções de *ranking* para recomendação. Já os terminais são identificados de acordo com a abordagem de recomendação utilizada.

Como mencionado anteriormente, este trabalho utiliza modelos baseados em usuários e itens da filtragem colaborativa. Nesse caso, as funções utilizam apenas

a informação de avaliação dos usuários. Por exemplo, a base *MovieLens* disponibiliza para cada usuário a lista de itens que esse avaliou e um *rating* entre 1 e 5, que representa a preferência do usuário pelo filme. No capítulo 2 foram estudados algoritmos que usam essa informação para calcular a similaridade entre os usuários ou itens e para determinar a preferência do usuário pelo item. Esses algoritmos serviram de base para definir o conjunto de terminais.

A Tabela 3.1 lista os terminais utilizados na recomendação por filtragem colaborativa. Os terminais foram definidos de acordo com métricas já presentes na literatura, detalhadas no Capítulo 2. Novamente, considere que $I = \{i_1, i_2, i_3, ..., i_n\}$ é o conjunto de itens, $U = \{u_1, u_2, u_3, ..., u_m\}$ é o conjunto de usuários, u_a é o usuário para qual será feito a recomendação, v_{ai} é o *rating* do usuário a para o item i, v_{ui} é o *rating* do usuário u para o item i, $\overline{v_u}$ e $\overline{v_i}$ são a média das avaliações do usuário u e do item i, respectivamente.

Tabela 3.1: Lista de terminais para a abordagem colaborativa

T1	Valor médio das avaliações do usuário
T2	Valor médio das avaliações do item
T3	b_u
T4	b_i
T5	b_{ui}
T6	Desvio padrão das avaliações do usuário
T7	Número de usuários
T8	Número de itens
T9	Número de itens avaliados pelo usuário
T10	Número de usuários que avaliaram o item
T11_1	$\sum_{u \in Neigh(a)} (v_{ui} - \overline{v_u}) \times s(a,u)$, s = cosseno
T11_2	$\sum_{u \in Neigh(a)} (v_{ui} - \overline{v_u}) \times s(a,u)$, s = coeficiente de correlação de Pearson
T11_3	$\sum_{u \in Neigh(a)} (v_{ui} - \overline{v_u}) \times s(a,u)$, s = Pearson com restrições
T12_1	$\sum_{u \in Neigh(a)} s(a,u)$, s = cosseno
T12_2	$\sum_{u \in Neigh(a)} s(a,u)$, s = coeficiente de correlação de Pearson
T12_3	$\sum_{u \in Neigh(a)} s(a,u)$, s = Pearson com restrições
T13_1	$\sum_{j \in Neigh(i)} (v_{aj} - \overline{v_j}) \times s(i,j)$, s = cosseno
T14_1	$\sum_{j \in Neigh(i)} s(i,j)$, s = cosseno
TNUM	Número entre 1 e 5

Os terminais $T3$, $T4$ e $T5$ foram extraídos da proposta do Koren [2008]. Os terminais $T1$, $T2$ e $T6$ foram definidos a partir das propostas de normalização apresentadas por Cacheda et al. [2011]. Os terminais $T11$ e $T12$ são derivados das métricas de normalização que utilizam a vizinhança dos usuários mais próximos definidas em Koren [2008] e Cacheda et al. [2011]. Os terminais $T13$ e $T14$ usam os itens mais próximos

e foram também definidos por Cacheda et al. [2011]. Os terminais $T7$, $T8$, $T9$ e $T10$ são sugestões deste trabalho.

3.3 Avaliação dos Indivíduos

Após a criação do indivíduo, o próximo passo dentro do algoritmo de programação genética é a avaliação dos indivíduos. Em nossa proposta, cada indivíduo é uma função que, dado um usuário e um item, retorna um valor que mede a preferência do usuário para com aquele item. Tendo os valores para todos os itens da base é possível ordená-los de forma a obter um *ranking*. Como cada indivíduo representa uma função a partir da qual será gerado um *ranking*, eles devem ser avaliados utilizando métricas de avaliação de *rankings*. Para cada *ranking*, quatro métricas distintas serão consideradas: (1) precisão, (2) revocação, (3) novidade e (4) diversidade. Assim, esses quatro valores são considerados como métricas de *fitness*, mas utilizados de formas distintas, como detalhado na fase de seleção.

É importante ressaltar que, no caso da tarefa de recomendação, a avaliação do *ranking* pode ser complicada porque, em geral, o número de itens avaliados pelo usuário é muito menor que o número de itens presentes na base. Embora esquemas de avaliação tradicional baseados no *ranking* em relação aos itens relevantes sejam muito utilizados [Herlocker et al., 2004], esse trabalho segue a metodologia definida por Cremonesi et al. [2010]. Cremonesi et al. [2010] modifica a forma como a precisão e revocação são medidas de forma que itens não avaliados possam ser considerados. Como a metodologia de avaliação dos indivíduos deve ser a mesma utilizada para recomendação final, vamos antecipar para este capítulo alguns detalhes sobre a metodologia experimental.

Metodologia Experimental para Avaliação de *Rankings*

A metodologia proposta em Cremonesi et al. [2010] funciona da seguinte forma. Da base completa, 1.4% dos *ratings* são selecionados e retirados para compor o que é denominado *probe base* P. O restante dos dados é adicionado ao conjunto de treino Tr. De P, retiram-se todos os *ratings* considerados relevantes para compor o conjunto de teste Te (aqueles com o maior valor de *score* possível). O modelo é treinado com os dados de Tr, e a precisão e revocação medidas em cima da base de teste. Para cada item i da base de teste Te, 1000 itens que o usuário não avaliou são selecionados aleatoriamente, e considerados como não-relevantes. Esses 1001 itens são então ordenados, e os N itens do topo do *ranking* utilizados para avaliações de precisão, revocação, novidade e diversidade.

Precisão e Revocação

A precisão e revocação são calculadas com base na posição que o item i de Te fica em relação aos 1000 considerados não-relevantes. As funções tem que ser capazes de posicionar esses itens na lista dos top-N selecionados. Sendo r a posição do item i no ranking, se $r <= N$ consideramos que este item é recomendado para o usuário u, e temos um *hit*. Caso não esteja nos top-N, temos um $miss$. Baseado nesses conceitos, precisão e revocação são definidas como [Cremonesi et al., 2010]:

$$recall(N) = \frac{\#hits}{|Te|} \tag{3.1}$$

$$precision(N) = \frac{\#hits}{N \times |Te|} = \frac{recall(N)}{N} \tag{3.2}$$

Novidade e Diversidade

Já para o cálculo da novidade e da diversidade foram utilizados os modelos apresentados por Vargas & Castells [2011]. A grande vantagem destes modelos em relação a outras abordagens presentes na literatura é que eles consideram a posição do item no *ranking* e sua relevância para o usuário. Como a metodologia experimental apresentada anteriormente considera 1000 itens não-relevantes e um relevante, as métricas abaixo descritas não levam em consideração a relevância dos itens.

O cálculo da novidade de um item no *ranking* leva em consideração sua popularidade na base de dados. Entende-se por popularidade a probabilidade de um item ser visto, que é calculada por:

$$P(seen|i_k) = \frac{|u \in U|v(u,i) \neq \emptyset|}{|U|} \tag{3.3}$$

onde U é o conjunto de usuários. Desta forma, a novidade de uma lista top-N de uma recomendação R para um usuário u é dada por:

$$nov(R(N)) = EPC(N) = C \sum_{i_k \in R}^{i_N} disc(k)(1 - p(seen|i_k)) \tag{3.4}$$

onde $disc(k)$ é um fator que varia de acordo com a posição do item no *ranking* e é calculado por por $disc(k) = 0.85^{k-1}$, C é uma constante de normalização dada por $1/\sum_{i_k \in R}^{i_N} disc(k)$ e k é a posição do item no *ranking*.

Para medir a diversidade é usado um modelo baseado em distância entre itens,

que é calculado por:

$$div(R(N)) = EILD(N) = \sum_{\substack{i_k \in R \\ i_l \in R \\ l \neq k}}^{i_N, l_N} C_k disc(k) disc(l|k) d(i_k, i_l) \tag{3.5}$$

onde $disc(l|k) = disc(max(1, l-k))$ também é um fator que varia de acordo com a posição no *ranking* de l e k, e $d(i_k, i_l)$ é a similaridade do cosseno entre dois itens que é calculada por:

$$d(i,j) = 1 - \frac{|U_i \cap U_j|}{\sqrt{|U_i|}\sqrt{|U_j|}} \tag{3.6}$$

sendo U_i os usuários que gostam do item i e U_j os que gostam do item j.

Observe que para cada linha do teste será gerado um valor para cada métrica avaliada. A média de todas as linhas é calculada e a *fitness* definida como a média de cada um dos objetivos pretendidos.

Sendo assim, nossa abordagem procura encontrar soluções que levem em consideração três objetivos: (1) o número de *hits* (através das métricas de precisão e revocação), (2) novidade e (3) diversidade.

3.4 Seleção dos Indivíduos

Após a avaliação dos indivíduos, ocorre a etapa de seleção, que escolhe aqueles que passarão por operações de mutação e cruzamento. Nossa seleção será baseada nos três critérios definidos na seção anterior: número de *hits* (com a precisão e a revocação), novidade e diversidade. Para isto, nosso arcabouço apresenta uma abordagem de seleção multiobjetivo [Deb, 2001]. Na Seção 2.2 foram apresentadas três abordagens para o problema da seleção em problemas multiobjetivo. O primeiro falava em dar pesos aos objetivos de acordo com sus importância. Essa ideia não foi considerada por ser difícil atribuir valores a cada um dos objetivos. Ao invés disso, implementamos dois outros tipos: (1) uma abordagem lexicográfica e (2) uma abordagem utilizando dominância de Pareto. Independente da abordagem implementada, o processo de seleção se inicia com um torneio entre os participantes. Durante o torneio, as comparações entre indivíduos são feitas de formas diferentes nas duas abordagens.

No caso da abordagem lexicográfica, que ordena os objetivos de acordo com sua prioridade, a precisão foi definida como a métrica mais importante, seguida da revocação, diversidade e novidade. Assim, na abordagem lexicográfica a comparação do torneio é feita utilizando os objetivos pretendidos de acordo com a ordem de prioridade

definida. Ao se comparar dois indivíduos selecionados aleatoriamente, seleciona-se primeiro aquele que possui maior precisão. Em caso de empate é selecionado o de melhor revocação, e assim por diante com a diversidade e novidade. Ao longo da evolução são passados para gerações seguintes tanto elementos de indivíduos com bom grau de precisão quanto elementos que garantem bons resultados nos outros objetivos, sem perder na precisão. A intuição por trás dessa abordagem é de que nada adianta ter *rankings* diversos se a precisão deles for baixa.

Já na abordagem por Pareto, a comparação dos indivíduos selecionados é feita utilizando o critério de dominância de Pareto. Dado um conjunto de objetivos, uma solução domina outra se a primeira não é pior que a segunda em todos os objetivos, e existe pelo menos um objetivo no qual ela é melhor [Poli et al., 2008].

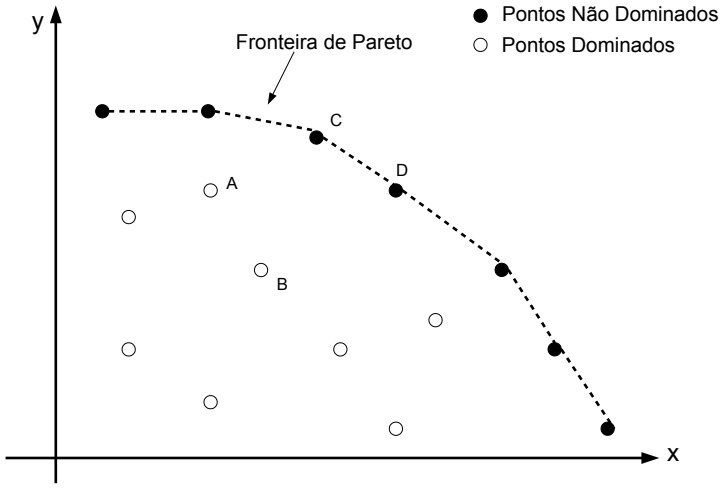


Figura 3.3: Exemplo de dominância

A Figura 3.3 ilustra esse conceito em um problema com dois objetivos. Se analisarmos os pontos A e B, percebe-se que A é melhor que B no objetivo representado pelo eixo y e B é melhor que A no objetivo representado no eixo x. Nesse caso, não é possível dizer que a solução A ou a solução B seja melhor, uma vez que cada uma delas é melhor em um dos objetivos. O mesmo acontece com C e D. No entanto, se compararmos os pontos B e C, nota-se que C é melhor que B tanto no objetivo representado no eixo x quanto no objetivo representado pelo eixo y. Logo, podemos afirmar que a solução C domina a solução B. Ainda em relação as soluções C e D, ambas não são dominadas por nenhuma outra solução no espaço de soluções estudado. Chama-se de fronteira de Pareto o conjunto de soluções que não é dominada por nenhuma outra, conforme ilustrado pela linha pontilhada na Figura 3.3.

A seleção baseada no critério de dominância de Pareto utiliza a abordagem apre-

sentada por [Ritzel et al., 1994]. Nessa abordagem, dois indivíduos são selecionados aleatoriamente da população, e comparados com os K indivíduos selecionados durante o torneio utilizando o critério de dominância de Pareto. O vencedor do torneio é aquele que domina as outras K soluções [Deb, 2001]. Se as duas soluções são dominadas pelo conjunto de K indivíduos, vence aquela que foi dominada por um menor número de indivíduos.

3.5 Operações

Durante a evolução do GP, três operações são utilizadas: cruzamento, mutação e reprodução. Essas operações são aplicadas nos indivíduos selecionados utilizando o método de torneio descrito na seção anterior.

No cruzamento (Figura 3.4), os dois indivíduos selecionados, denominados de indivíduos-pai trocam material genético. Para isso, um nó de cada indivíduo é selecionado aleatoriamente, e as subárvores a partir deste nó são trocadas entre si. Dois novos indivíduos, denominados indivíduos-filho, são gerados.

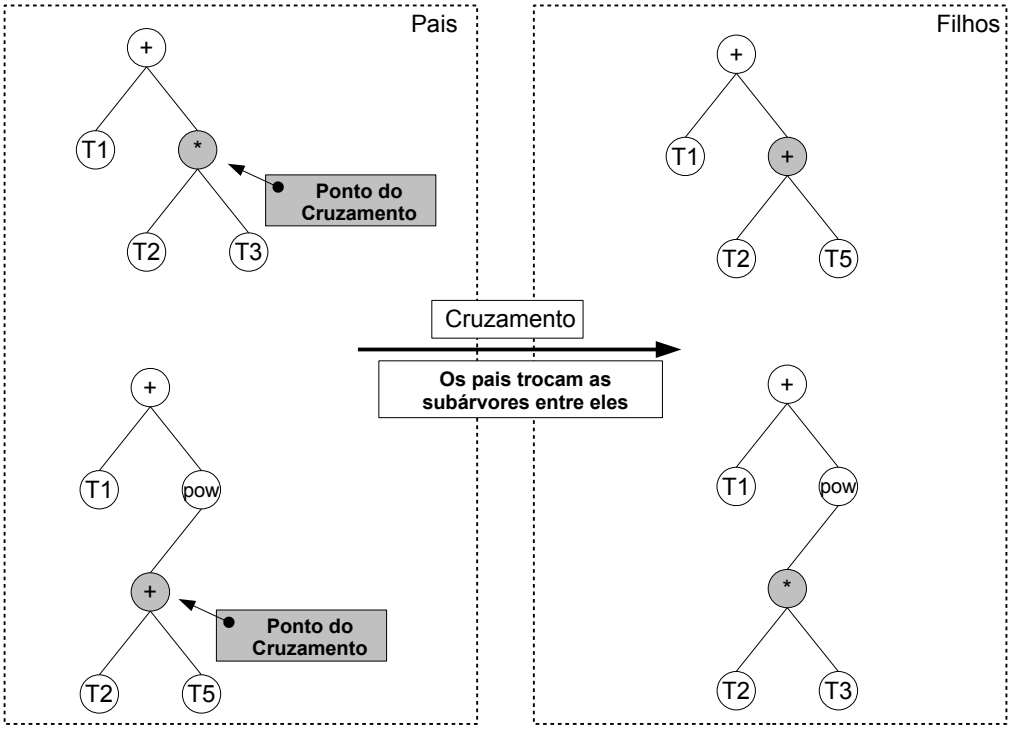


Figura 3.4: Exemplo de cruzamento.

Na mutação uma sub-expressão de um indivíduo selecionado é escolhida e modificada aleatoriamente, gerando um novo indivíduo. O arcabouço implementa dois tipos

de mutação [Poli et al., 2008]: (1) mutação em um ponto[1] e (2) mutação de redução[2]. A mutação de um ponto troca um ponto escolhido na árvore seguindo algumas restrições. A Figura 3.5 mostra um exemplos destas restrições. Na Figura 3.5(a) o ponto escolhido aleatoriamente é um terminal. Neste caso, esse ponto só pode ser trocado por outro terminal. Na Figura 3.5(b) o ponto selecionado é uma função. Uma função só pode ser trocada por uma outra função com o mesmo número de argumentos. No exemplo mostrado, o ponto escolhido (a função de multiplicação) foi trocado pela função de divisão. Ambas possuem dois argumentos.

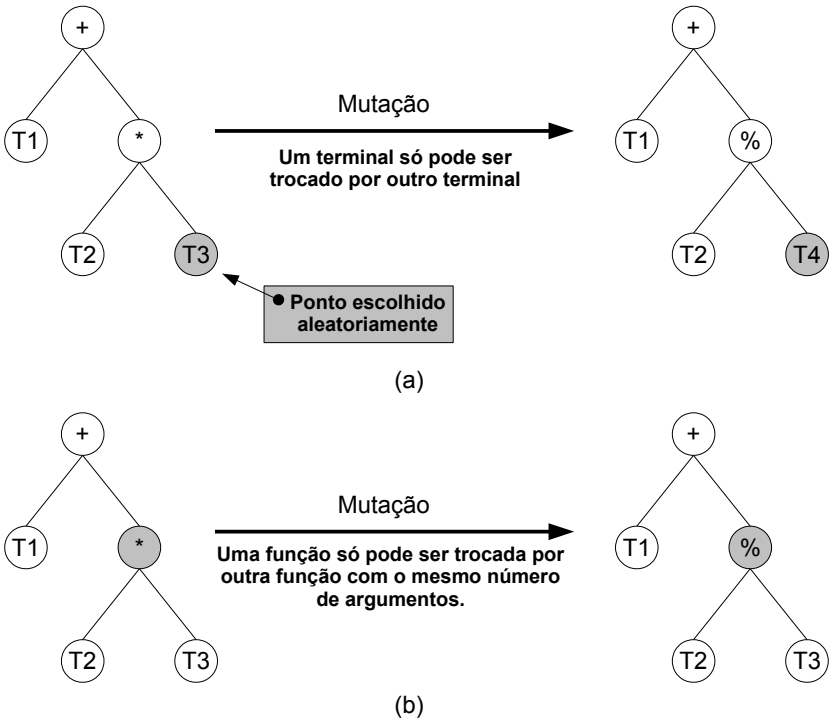


Figura 3.5: Exemplo de mutação em um ponto. (a) Mutação trocando terminais. (b) Mutação trocando funções.

Na mutação de redução (figura 3.6) a subárvore do nó selecionado é substituída por um terminal. Essa operação encolhe a árvore, e ajuda nos casos em que se deseja criar indivíduos menores. Essa redução pode ser interessante quando se leva em consideração o tamanho do indivíduo na função de *fitness* e também quando deseja reduzir o tamanho de indivíduos grandes, que podem comprometer o desempenho do GP.

Por fim, a reprodução consiste em simplesmente selecionar um indivíduo e passá-lo para a geração seguinte sem alterações.

[1]No arcabouço essa mutação é denominada *swap*
[2]No arcabouço essa mutação é denominada *shrink*

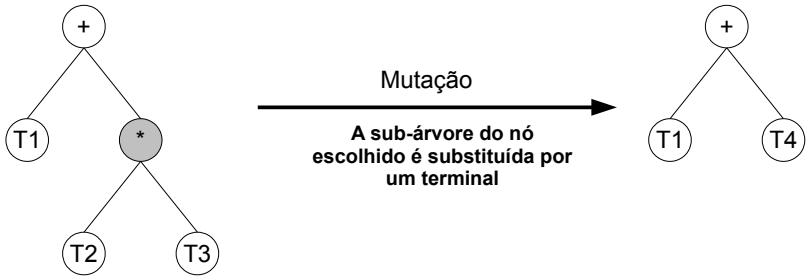


Figura 3.6: Exemplo de mutação de redução.

3.6 Elitismo e Escolha da Melhor Solução

Ao final de cada geração, os melhores indivíduos da geração atual são passados para a geração seguinte sem nenhuma alteração. Essa técnica é denominada de elitismo. O arcabouço implementa duas formas de elitismo. A primeira consiste simplesmente em passar o indivíduo com melhor precisão. Essa abordagem é usada juntamente com a seleção lexicográfica. A segunda passa para a próxima geração os melhores indivíduos não dominados segundo cada um dos três objetivos de interesse. Essa abordagem é utilizada juntamente com a seleção de dominância de Pareto.

Ao final da execução do GUARD, o arcabouço retorna três indivíduos que representam os melhores para cada um dos objetivos analisados: número de *hits* (precisão, revocação), novidade e diversidade. Ao final temos três possíveis *rankings* que podem ser retornados aos usuários, que podem ser inclusive combinados. Essa é uma forma simples de retornar as soluções de acordo com a fronteira de Pareto, uma vez que o problema da tomada de decisão automática sem auxílio de um usuário é ainda um problema em aberto nessa área [Pedro & Takahashi, 2011].

Capítulo 4

Resultados Experimentais

Esse capítulo descreve a avaliação experimental executada para testar o GUARD (*A Genetic Unified Approach for Recommendation*).

4.1 Coleções Utilizadas

O arcabouço proposto foi testado no cenário de recomendação de filmes em duas bases: (1) *Movielens 100k* e (2) *Movielens* 1M. A base 100k consiste em 100.000 avaliações usuário-item com 943 usuários e 1.682 filmes. Já a base 1M possui 1 milhão de avaliações com um total de 6.040 usuários e 3.952 itens. Ambas possuem apenas informações explícitas dos usuários, ou seja, possuem apenas informações que foram fornecidas diretamente pelos usuários.

Embora todas as análises experimentais tenham sido feitas nessas duas bases, utilizamos também, para avaliar a generalização do arcabouço proposto, a base da *LastFM*[1]. Nesse caso, as funções geradas na *Movielens* puderam ser avaliadas em um contexto de recomendação diferente, já que a *LastFM* contém informações implícitas dos usuários, ou seja, não é armazenado a preferência do usuário por um item. A base armazena o histórico de músicas que cada usuário escutou. Para uso no trabalho é necessário converter esse histórico em informações explícitas.

A preferência do usuário é calculada de acordo com a frequência que ele escuta um artista [Vargas & Castells, 2011]. A tarefa de recomendação passa a ser recomendar artistas para um determinado usuário. A base original da *LastFM* possui 19.150.868 registros de músicas escutadas pelos usuários. Esses registros abrangem um total de 176.948 artistas e 992 usuários. Fazendo o mapeamento do *feedback* implícito para o *feedback* explícito foram extraídos um total de 889.558 avaliações usuário-artista.

[1] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/

4.2 Metodologia Experimental

Para realizar os experimentos cada base foi dividida em três partes: treino, validação e teste, de acordo com a metodologia que foi apresentada no Capítulo 3. É importante relembrar que 1.4% dos *ratings* são selecionados e retirados para compor o que é denominado *probe base* P. O restante dos dados é adicionado ao conjunto de treino Tr. De P, retiram-se todos os *ratings* considerados relevantes para compor o conjunto de teste Te (aqueles com o maior valor de *score* possível). O modelo é treinado com os dados de Tr, e a precisão e revocação medidas em cima da base de teste. Para cada item i da base de teste Te, 1000 itens que o usuário não avaliou são selecionados aleatoriamente, e considerados como não-relevantes. Esses 1001 itens são então ordenados, e os N itens do topo do *ranking* utilizados para avaliações de precisão, revocação, novidade e diversidade

A precisão e revocação são calculadas com base na posição que o item i de Te fica em relação aos 1000 considerados não-relevantes. As funções tem que ser capazes de posicionar esses itens na lista dos top-N selecionados. Utilizando a mesma metodologia anterior dividimos nosso Treino Tr em outras duas partes: novo treino Tr' e na base de validação Va que contém assim como o teste apenas itens relevantes. Os terminais da PG são calculados utilizando as informações da base de treino Tr' e, ao longo da evolução, as funções são avaliadas utilizando a base de validação Va. O melhor indivíduo ao final de todas as gerações são avaliados utilizando a base de teste Te. Cada experimento foi executado 5 vezes, cada um com uma semente aleatória diferente para criação dos indivíduos do GUARD, já que ele é um método não-determinístico.

A Tabela 4.1 lista os principais parâmetros de PG que foram utilizados no arcabouço. Todos os experimentos foram executados com esses parâmetros. O tamanho da população determina o número de indivíduos que serão evoluídos a cada geração. A probabilidade de criação determina a porcentagem de indivíduos que serão criados aleatoriamente a cada nova geração. Já a probabilidade em que as operações acontecem são determinadas pelos parâmetros de probabilidade de cruzamento e mutação. Os valores do número de gerações e tamanho do torneio são específicos de cada experimento, e serão definidos ao decorrer do capítulo. Os valores de precisão, revocação, novidade e diversidade foram calculados utilizando os top-20 itens no *ranking*. Esses parâmetros foram encontrados através de testes preliminares, avaliando a convergência do algoritmo de programação genética.

Todos os resultados foram comparados com uma abordagem baseada em memória, descrita na Seção 2.1.1, que utiliza o cosseno como distância de similaridade e número de vizinhos (N) igual a 30. Essa abordagem foi definida na Equação 2.5, que é

Tabela 4.1: Parâmetros utilizados

Parâmetro	**Valor**
Tamanho da População	100
Probabilidade de Cruzamento	0.9
Probabilidade de Criação	0.02
Probabilidade de Mutação	0.1
Criação dos indivíduos	Ramped Half and Half
Seleção	Torneio (com Multiobjetivo)

utilizada quando o objetivo do sistema de recomendação é retornar uma nota do usuário para o item. Nesse caso, o denominador da Equação 2.5 força os valores a ficarem em um intervalo de *ratings* pré-definido (por exemplo, 1 a 5 no caso da *Movielens*). Para a abordagem de recomendação de *rankings*, utilizada nesse trabalho, podemos simplificar a equação eliminando o denominador [Cremonesi et al., 2010]. Além da abordagem baseada em memória, os resultados foram comparados com o PureSVD, descrito na Seção 2.1.2, que hoje representa o estado da arte em recomendação por filtragem colaborativa. O PureSVD foi executado considerando 50 fatores latentes.

Os resultados do arcabouço foram comparados estatisticamente com os do PureSVD utilizando o *test-t* com 95% de intervalo de confiança. Nas tabelas de resultados, os valores marcados com ▲ são estatisticamente melhores do que o PureSVD. Já os valores marcados com ▼ são estatisticamente inferiores aos do PureSVD.

4.3 Movielens 100k

Nessa seção apresentamos os resultado para a base *Movielens 100k*. Os experimentos foram realizados em três fases: na primeira fase as funções foram avaliadas considerando apenas o número de *hits* com as métricas de precisão e revocação juntamente com a seleção por dominância de Pareto, que consistia na primeira proposta desta dissertação (a partir de agora denominada GUARD-P2). Dados os resultados positivos, em uma segunda fase foram adicionados a seleção por dominância de Pareto as medidas de novidade e diversidade, totalizando três objetivos calculados pelas 4 métricas de precisão, revocação, novidade e diversidade (designada GUARD-P4). Na terceira fase, as funções foram avaliadas novamente considerando os três objetivos, mas a abordagem de Pareto foi substituída pela abordagem lexicográfica, originando o GUARD-Lex4, conforme detalhado nas seções a seguir.

Em relação aos parâmetros do arcabouço utilizamos para o GUARD-P2 e GUARD-P4 o número de gerações igual a 100 e o tamanho do torneio igual a 2. Para o

GUARD-Lex4 foram utilizados 50 gerações e um tamanho de torneio igual a 20, para acelerar a convergência.

4.3.1 Resultados Obtidos

A Tabela 4.2 mostra o resultado da média de cinco execuções para cada um dos três experimentos descritos acima. Para o experimento com o GUARD-P2 exibimos a média dos melhores indivíduos em precisão e revocação. No caso do GUARD-P4 e GUARD-Lex4, além dos resultados dos melhores indivíduos em precisão e revocação, exibimos também os resultados para os melhores em novidade e diversidade.

As funções encontradas pelas três versões do GUARD conseguem gerar resultados melhores, em todos os casos, quando comparados com a abordagem baseada em memória. No entanto, nosso objetivo é comparar as funções com algoritmos com características distintas, no caso, as abordagem baseada em modelos. Os resultados detalhados a seguir vão focar na comparação com o PureSVD, algoritmo baseado em modelo e estado da arte na área de recomendação por filtragem colaborativa.

O melhor resultado geral para precisão e revocação foi dado pelo GUARD-P4, que obteve uma precisão de 0,0297 contra 0,0266 em relação ao PureSVD. Apenas para darmos uma ideia para o leitor que não está acostumado com a metodologia proposta por Cremonesi et al. [2010], onde os resultados se diferenciam nas terceira ou quarta casa decimal, mostramos aqui as diferenças de número de *hits* para cada método. Como explicado anteriormente, um *hit* acontece quando o item relevante do conjunto Te aparece nos top-20 itens quando ordenado juntamente com os outros 1000 itens não relevantes. Em número de *hits* esse resultado mostra que acertamos, em um total de 297 recomendações, 20 a mais do que o *PureSVD*, o que representa um ganho de 6,7%. Observe também que, para os resultados do GUARD-P4, ao escolhermos os indivíduos com melhor novidade e diversidade, o resultado para precisão e revocação foram nulos ou muito próximos de zero. Selecionar tais indivíduos não faz sentido. Embora outros indivíduos pudessem ter sido selecionados, como já citado, o problema de decidir quais indivíduos retornar de uma fronteira de Pareto sem auxílio do usuário continua em aberto.

Assim, como solução para esse problema, propomos o GUARD-Lex4. Como mencionado, a seleção lexicográfica leva em consideração os 3 objetivos assim como a seleção por dominância de Pareto. No entanto, nesse caso é definida uma ordem de prioridade para cada um dos objetivos. Neste trabalho, o objetivo de maior prioridade é o número de *hits* seguida da diversidade e novidade. Para esses resultados, observe que há uma perda na precisão e revocação mas um ganho quando olhamos a novidade e a

diversidade, principalmente ao considerar os melhores indivíduos nesses dois objetivos.

Tabela 4.2: Resultado comparativo da média dos melhores indivíduos obtidos pelas três versões do GUARD (*A Genetic Unified Approach for Recommendation*) para a base *Movielens 100k*. Utilizamos a seguinte representação: **P/R** para os melhores indivíduos em precisão e revocação, **N** para os melhores indivíduos em novidade e **D** para os melhores indivíduos em diversidade

Algoritmo		**P@20**	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória		0,0197	0,3939	0,7705	0,7944
PureSVD		0,0266	0,5320	0,7999	0,8807
GUARD-P2	P/R	0,0293 ± 0,0006 ▲	0,5859 ± 0,012 ▲	0,7251 ± 0,008 ▼	0,7962 ± 0,019 ▼
GUARD-P4	P/R	0,0297 ± 0,0004 ▲	0,5933 ± 0,010 ▲	0,7254 ± 0,009 ▼	0,8005 ± 0,018 ▼
	N	-	-	0,9989 ± 0,0002 ▲	1,0000 ▲
	D	-	-	0,9989 ± 0,0002 ▲	1,0000 ▲
GUARD-Lex4	P/R	0,0289 ± 0,001 ▲	0,5785 ± 0,021 ▲	0,7363 ± 0,018 ▼	0,8076 ± 0,023 ▼
	N	0,0242 ± 0,004 ▼	0,4835 ± 0,097 ▼	0,8182 ± 0,071 ▲	0,8769 ± 0,062 ▼
	D	0,0242 ± 0,004 ▼	0,4848 ± 0,099 ▼	0,8176 ± 0,072 ▲	0,8810 ± 0,055 ▲

Tabela 4.3: Resultado comparativo dos melhores indivíduos obtidos pelo GUARD (*A Genetic Unified Approach for Recommendation*) com o *PureSVD* para a base *Movielens 100k*. Em negrito destacamos os melhores resultados. Utilizamos a seguinte representação: **P/R** para os melhores indivíduos em precisão e revocação, **N** para os melhores indivíduos em novidade e **D** para os melhores indivíduos em diversidade

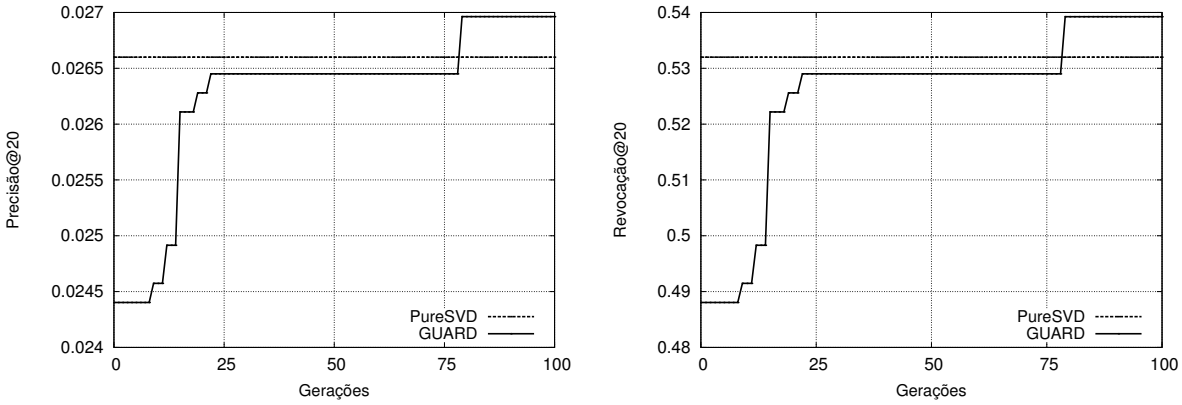
Algoritmo		**P@20**	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória		0,0197	0,3939	0,7705	0,7944
PureSVD		0,0266	0,5320	0,7999	0,8807
GUARD-P2	P/R	0,0300	0,5993	0,7203	0,8017
GUARD-P4	P/R	**0,0303**	**0,6061**	0,7235	0,8061
GUARD-Lex4	P/R	0,0298	0,5960	0,7211	0,8043
	N	0,0249	0,4983	**0,8421**	0,9103
	D	0,0249	0,4983	0,8421	**0,9103**

Porém, note que os resultados apresentados até o momento são uma média. Para que a recomendação seja feita por um modelo baseado em memória, temos que escolher uma das cinco funções evoluídas. Assim, a Tabela 4.3 mostra o resultado do melhor indivíduo no conjunto de validação para cada um dos experimentos citamos acima. O melhor resultado de precisão e revocação continua sendo quando usamos o GUARD-P4.

No caso do GUARD-P4, o melhor indivíduo em precisão e revocação gera recomendações com resultados bem próximos aos mostrados anteriormente. Já aquele melhor em novidade e diversidade gera, para esses objetivos, resultados melhores do que o baseline com valores próximos de acurácia. Em números, o melhor indivíduo em novidade/diversidade apresenta uma diferença de 32 *hits* no universo de 297 recomendações realizadas em relação ao PureSVD. Já a novidade e diversidade sobem de 0.7999

e 0.8807 com o PureSVD para 0.8421 e 0.9103, respectivamente. Nesse caso, podemos dizer que a função gerada pelo GUARD-P4 é ideal para usuários novos no sistema, que privam por precisão. Já os resultados gerados pelo GUARD-Lex4 seriam ideais para usuários experientes, que preferem sacrificar a precisão em nome da novidade e diversidade.

Para entender o comportamento do GUARD durante a evolução, a Figura 4.1 mostra a evolução para o indivíduo com melhor precisão e revocação (GUARD-P4). A Figura 4.1(a) mostra os valores da precisão ao longo de 100 gerações e a Figura 4.1(b) mostra os valores da revocação. Afim de comparação, a linha pontilhada representa os valores para o *PureSVD*. A linha contínua representa o processo de evolução do melhor indivíduo a cada geração. O valor do melhor indivíduo supera os valores do baseline no treino na geração 75.



(a) Precisão

(b) Revocação

Figura 4.1: Gráfico da evolução da precisão (a) e revocação (b) do melhor indivíduo na base Movielens 100k para o GUARD

4.3.2 Agregação de *Rankings*

Até agora vimos que ao final da evolução selecionamos três indivíduos, que representam os melhores em cada um dos objetivos sendo otimizados. Isso significa que no final temos três possíveis *rankings* que podem ser apresentados aos usuários. No entanto, uma outra abordagem seria combinar tais listas com a finalidade de retornar somente um ranking que possua características de acurácia, novidade e diversidade.

Para isso propomos combinar as listas obtidas de duas formas. A primeira, denominada de *borda count* [Dwork et al., 2001], leva em consideração a posição em que o item aparece no *ranking*. Dado um conjunto de *rankings* $R = r_1, r_2, ..., r_k$ cada item

i de r_k recebe um peso de $P_k(i)$ = posição de i no ranking r_k mais 1. O valor total desse item na agregação dos k *rankings* é calculado pelo somatório $P(i) = \sum_{j=1}^{k} P_k(i)$.

A segunda leva em consideração a frequência do item nos *rankings*. Aqueles itens que aparecem em todos os *rankings* tem maior relevância do que os que aparecem em apenas um. Como o objetivo é manter a precisão e revocação altas, se um item aparece no *ranking* de melhor precisão/revocação ele tem um peso maior do que aquele que aparece apenas nos *rankings* de novidade e diversidade.

A Tabela 4.4 mostra o resultado da agregação de *rankings* comparados ao baseline *PureSVD* e a função com melhor precisão/revocação encontrada pelo GUARD-P4.

Tabela 4.4: Agregação do ranking dos melhores indivíduos em precisão, revocação, novidade e diversidade obtidos utilizando GUARD-P4. Os valores representam a média de 5 execuções.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
	P@20	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória	0,0197	0,3939	0,7705	0,7944
PureSVD	0,0266	0,5320	0,7999	0,8807
GUARD-P4	0,0297 ± 0,0004 ▲	0,5933 ± 0,010 ▲	0,7254 ± 0,009 ▼	0,8005 ± 0,018 ▼
BordaCount	0,0233 ± 0,0004 ▼	0,4660 ± 0,008 ▼	0,8524 ± 0,006 ▲	0,9504 ± 0.004 ▲
Votação	0,0297 ± 0.0004 ▲	0,5933 ± 0.010 ▲	0,7594 ± 0.005 ▼	0,8306 ± 0.013 ▼

Observe que na agregação de *ranking* com o *borda count* perde-se na precisão e revocação no entanto os valores de novidade e diversidade melhoram em relação ao baseline e a função encontrada pelo GUARD-P4. Para a segunda abordagem de agregação, os resultados de precisão e revocação se mantém e a novidade e diversidade melhoram em relação ao GUARD-P4. No entanto, esses ganhos nos dois últimos atributos ainda não possibilitam que eles sejam melhores que o PureSVD nesses dois critérios, embora a precisão e revocação continuem sendo melhores.

Aplicamos também a agregação para o GUARD-Lex4. Os resultados são listados na Tabela 4.5. Para essa combinação, os valores variam pouco em relação ao GUARD-Lex4, embora os resultados de diversidade e novidade melhorem novamente. Assim, se quer melhores resultados para diversidade e novidade, a agregação de *rankings* pode ser uma abordagem interessante, e no futuro métodos mais sofisticados para essa tarefa devem ser testados.

4.3.3 Análise dos Indivíduos

As seções anteriores analisaram os resultados obtidos pelas diferentes versões do GUARD, mas sem analisar as funções de recomendação geradas. Nesta seção será feita

Tabela 4.5: Agregação do ranking dos melhores indivíduos em precisão, revocação, novidade e diversidade obtidos utilizando o GUARD-Lex4. Os valores representam a média das 5 execuções.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
	P@20	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória	0,0197	0,3939	0,7705	0,7944
PureSVD	0,0266	0,5320	0,7999	0,8807
GUARD-Lex4	0,0289 ± 0,001 ▲	0,5785 ± 0,021 ▲	0,7363 ± 0,018 ▼	0,8076 ± 0,023 ▼
BordaCount	0,0272 ± 0,002 ▲	0,5448 ± 0,043 ▲	0,7633 ± 0,032 ▼	0,8450 ± 0,039 ▼
Votação	0,0289 ± 0,001 ▲	0,5784 ± 0,021 ▲	0,7590 ± 0,016 ▼	0,8303 ± 0,018 ▼

uma análise dos melhores indivíduos e dos terminais mais representativos ao longo da evolução. Para o GUARD-P2, o melhor individuo foi a função:

$$r_{ai} = ((\sqrt{T2_i} \times ((T12\_1_a)^2)^2 + (T12\_1_a)^2))^2 \tag{4.1}$$

onde $T2_i$ é a média dos *ratings* do item i e $T12\_1_a$ o somatório da similaridade de usuário a que é dado por:

$$\sum_{u \in Neigh(a)} s(a,u), \tag{4.2}$$

sendo s a similaridade de cosseno entre usuários e $Neigh(a)$ o conjunto de 30 vizinhos mais próximos do usuário a.

Para o experimento com o GUARD-P4, a função que representa o melhor indivíduo é dada por:

$$r_{ai} = \log(T2_i \times T2_i + (T12\_1_a)^2 + \log((T12\_1_a)^2) + 2.08 \times T2_i) \tag{4.3}$$

onde $T2_i$ e $T12\_1_a$ é definido como no exemplo anterior. E finalmente, no experimento utilizando GUARD-Lex4, o melhor indivíduo em precisão e revocação é dado por:

$$\begin{aligned} r_{ai} = {} & \log(((T12\_1_a)^2)) \times (T9_a + T4_i) + \log(T12\_1) \times \log((\log(T12\_1 \\ & \times (T9_a + T4_i)^3)) \times (T9_a + T4_i) + \log(T12\_1_a) \times (T9_a + T4_i)^2 \end{aligned} \tag{4.4}$$

e o melhor em novidade e diversidade por:

$$\begin{aligned} r_{ai} = {} & \log(((T12\_1_a)^2)) \times (T9_a + T4_i) + \log(T12\_1_a) \times \log((\log(T12\_1_a) \\ & \times (T9_a + T4_i)^3)) \times (T9_a + T4_i) + (T12\_1_a)^2 \end{aligned} \tag{4.5}$$

onde $T9$ é o número de itens avaliados pelo usuário a e $T4_i$ é o fator b_i da métrica *baseline estimates* definida no Capítulo 2.

Analisando os indivíduos percebemos que o terminal $T12\_1$ aparece em todos os melhores indivíduos, e em indivíduos com altos valores de acurácia quando analisamos a população como um todo. Esse terminal representa a vizinhança do usuário, ou seja, o quão ele é similar aos usuários vizinhos. Observe que utilizando apenas informações como a média de avaliações dos itens ($T2$) e a similaridade dos usuários (T12_1) obtemos resultados próximos ou melhores aos do PureSVD. Isso mostra a importância da vizinhança na qual o usuário está inserido para obter recomendações de qualidade. Essa é certamente uma das grandes vantagens da utilização do arcabouço: a obtenção de resultados consistentes através de informações simples.

Além do $T12\_1$, um subconjunto de terminais aparece constantemente ao longo da evolução. O gráfico da Figura 4.2 mostra a frequência de terminais que mais aparecem ao longo das gerações.

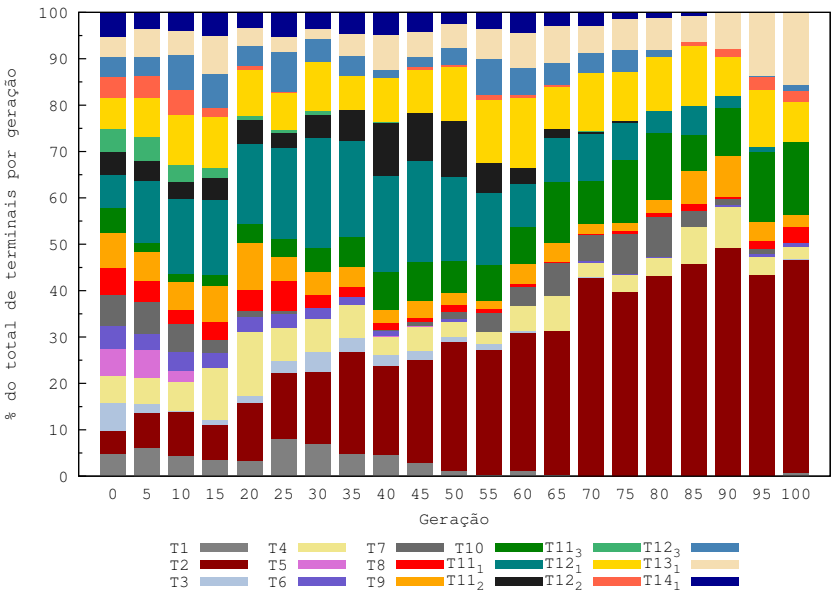


Figura 4.2: Porcentagem do total de terminais por geração na base Movielens 100k

Observe que alguns terminais são eliminados rapidamente como por exemplo o $T12\_2$, $T1$ e $T14\_1$. Já outros se mantém ao longo de toda evolução, como os terminais $T10$, $T13\_1$, assim como o $T2$ e o $T12\_1$ que aparecem nos melhores indivíduos.

4.3.4 Generalização das Funções Evoluídas

Para testar a capacidade de generalização das funções produzidas pelo GUARD-P4, a função representada pelo melhor indivíduo foi testada com outras base de dados como

a *MovieLens 1M* e a base da *LastFM*.

A Tabela 4.6 lista os resultados obtidos ao aplicarmos essa função evoluída na base *Movielens 100k* na base *Movielens 1M*. Observe que mesmo sem ter usado a base em questão para a evolução no arcabouço, os resultados são próximos do baseline.

Tabela 4.6: Resultado do melhor indivíduo encontrado pelo GUARD-P4 na base *Movielens 100k* quando testado na base *Movielens 1M*

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
Baseado em Memória	0,0186	0,3712	0,8232	0,8315
PureSVD	**0,0324**	**0,6477**	**0,8192**	**0,8880**
GUARD-P4	0,0293	0,5862	0,7768	0,8400

Tabela 4.7: Resultado do melhor indivíduo encontrado pelo GUARD-P4 na base *Movielens 100k* quando testado na base *LastFM*

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
Baseado em Memória	0,0338	0,6753	0,8822	0,9410
PureSVD	**0,0408**	**0,8168**	**0,9113**	**0,9570**
GUARD-P4	0,0376	0,7519	0,8791	0,9403

Em número de *hits*, o *PureSVD* consegue colocar 195 itens relevantes a mais no TOP-20 do que o melhor indivíduo encontrado pelo GUARD-P4. Esse valor representa uma diferença de 6,1% no universo de 3168 recomendações realizadas. O mesmo acontece com a base *LastFM*. Mesmo sendo de um domínio e de características completamente diferentes da recomendação de filmes. O melhor indivíduo também gera resultados próximos ao do *PureSVD*, como mostrado na Tabela 4.7.

Em número de *hits*, o ganho do *PureSVD* é de 6,5% que representa 156 *hits* a mais do que o melhor indivíduo encontrado pelo nosso arcabouço em 2402 recomendações realizadas. Isso mostra que os terminais que o arcabouço selecionou para os indivíduos levando em consideração a base *Movielens 100k* conseguem manter bons resultados quando testados em outras bases. Além da vantagem da simplicidade da função e de sua generalização, sabemos que podemos gerar funções competitivas para bases maiores usando o arcabouço em uma base bem menor (ou uma amostrada da mesma), e com menor custo computacional.

4.3.5 Análise de Desempenho

A Tabela 4.8 mostra a análise de desempenho do arcabouço para a base *Movielens 100k*. O tempo de processamento foi medido em três etapas: (1) Cálculo dos Terminais, (2)

Evolução e (3) Recomendação. As duas primeiras etapas, que consomem mais tempo, são executadas *off-line*. Um vez que a função foi encontrada, sua aplicação para geração do *ranking* é direta. O tempo descrito na tabela corresponde a aplicação da função na base de testes conforme a metodologia discutida no Capítulo 3. Para cada um dos 297 registros na base de testes, foram calculadas as preferências de 1.001 usuário-item.

Tabela 4.8: Análise de desempenho do GUARD na base *Movielens 100k*

Experimento	**Cálculo dos Terminais**	**Evolução**	**Recomendação**	**Total**
SEED 0	00:05:35	00:26:15	00:00:05	00:31:55
SEED 1	00:05:27	00:25:34	00:00:06	00:31:07
SEED 2	00:05:22	00:32:30	00:00:05	00:37:57
SEED 3	00:05:26	00:38:51	00:00:05	00:44:22
SEED 4	00:05:27	00:29:36	00:00:05	00:35:08
Média	00:05:27	00:30:33	00:00:05	00:36:05

4.4 Movielens 1M

Da mesma forma que a seção anterior, aqui mostramos os resultado para a base *Movielens 1M* seguindo a mesma metodologia: serão sempre apresentados resultados para o GUARD-P2 (apenas precisão e revocação com Pareto), GUARD-P4 (precisão, revocação, novidade e diversidade com Pareto) e GUARD-Lex4 (precisão, revocação, novidade e diversidade seguindo a abordagem lexicográfica).

Em relação aos parâmetros do arcabouço utilizamos para o GUARD-P2 e para o GUARD-P4 o número de gerações igual a 100 e o tamanho do torneio igual a 20. Para o GUARD-Lex4 foram utilizados 50 gerações e um tamanho de torneiro igual a 3, dado que esse último converge mais rápido.

4.4.1 Resultados Obtidos

Os resultados da base *Movielens 100k* mostram que, ao selecionarmos o indivíduo com melhor precisão, conseguimos obter uma função que supera o resultado dos baselines propostos e mantém valores próximos de novidade e diversidade. No entanto, para a base *Movielens 1M*, os resultados do melhor indivíduo na precisão e revocação são melhores que aqueles obtidos por outras abordagens baseadas em memória, mas não são superiores aos do PureSVD, apesar dos valores manterem-se próximo em todos os objetivos. Na Tabela 4.9 detalhamos o resultado da média das cinco execuções para cada um dos experimentos.

Tabela 4.9: Resultado comparativo da média dos melhores indivíduos obtidos pelo GUARD com o PureSVD para a base *Movielens 1M*. Utilizamos a seguinte representação: **P/R** para os melhores indivíduos em precisão e revocação, **N** para os melhores indivíduos em novidade e **D** para os melhores indivíduos em diversidade

Algoritmo		**P@20**	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória		0,0186	0,3712	0,8232	0,8315
PureSVD		**0,0321**	**0,6427**	**0,8298**	**0,8881**
GUARD-P2	P/R	0,0302 ± 0,0002 ▼	0,6043 ± 0,004 ▼	0,7717 ± 0,001 ▼	0,8412 ± 0,001 ▼
GUARD-P4	P/R	0,0303 ± 0,0002 ▼	0,6056 ± 0,003 ▼	0,7762 ± 0,017 ▼	0,8422 ± 0,009 ▼
	N	-	-	0,9995 ± 0,0007 ▲	0,9993 ± 0,001 ▲
	D	-	-	0,9995 ± 0,0007 ▲	1,0000 ▲
GUARD-Lex4	P/R	0,0303 ± 0,00005 ▼	0,6069 ± 0,001 ▼	0,7686 ± 0,001 ▼	0,8374 ± 0,002 ▼
	N	0,0293 ± 0,0005 ▼	0,5855 ± 0,011 ▼	0,7834 ± 0,008 ▼	0,8601 ± 0,012 ▼
	D	0,0293 ± 0,0005 ▼	0,5855 ± 0,011 ▼	0,7834 ± 0,008 ▼	0,8601 ± 0,012 ▼

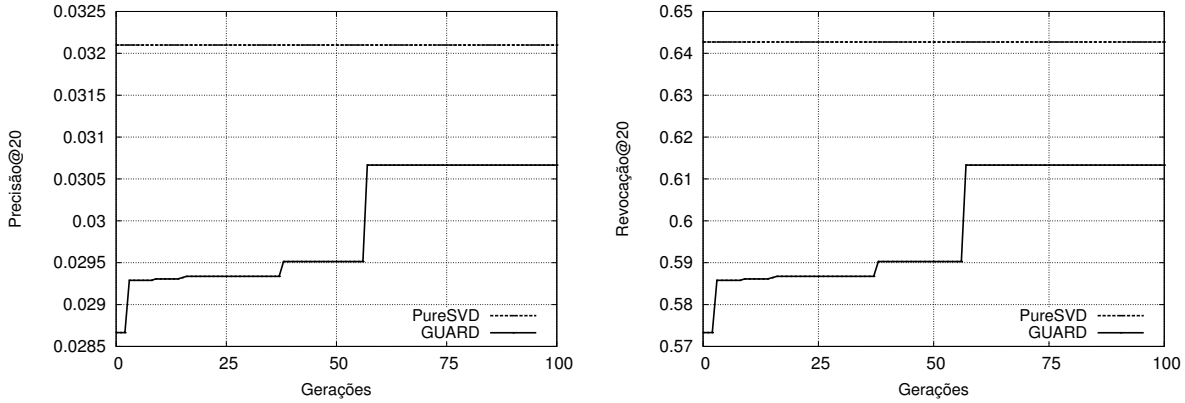
Na Tabela 4.10 destacamos os resultados dos melhores indivíduos. O melhor resultado foi obtido pelo GUARD-P2, com precisão de 0,0306 contra 0,0321 do *PureSVD*. Em número de *hits* esse resultado mostra que erramos, em um total de 3.168 recomendações, 90 a mais que o *PureSVD*, o que representa uma diferença de 3%. Já no caso do melhor indivíduo em novidade e diversidade do GUARD-Lex4, o PureSVD obteve 206 *hits* a mais, o que corresponde a 6,5%.

Tabela 4.10: Resultado comparativo dos melhores indivíduos obtidos pelo GUARD para a base *Movielens 1M*. Em negrito destacamos os melhores resultados. Utilizamos a seguinte representação: **P/R** para os melhores indivíduos em precisão e revocação, **N** para os melhores indivíduos em novidade e **D** para os melhores indivíduos em diversidade

Algoritmo		**P@20**	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória		0,0186	0,3712	0,8232	0,8315
PureSVD		**0,0321**	**0,6427**	**0,8298**	**0,8881**
GUARD-P2	P/R	0,0306	0,6121	0,7739	0,8391
GUARD-P4	P/R	0,0304	0,6086	0,7695	0,8378
GUARD-Lex4	P/R	0,0304	0,6086	0,7688	0,8372
	N	0,0289	0,5777	0,7925	0,8708
	D	0,0289	0,5777	0,7925	0,8708

A Figura 4.3 mostra a evolução para o GUARD-P2. A Figura 4.3(a) mostra os valores da precisão ao longo de 100 gerações e a Figura 4.3(b) mostra os valores da revocação. A linha pontilhada representa os valores obtidos pelo *PureSVD*. A linha contínua representa o processo de evolução do melhor indivíduo a cada geração. O melhor indivíduo evolui até a geração 50. Depois disso, o processo de evolução estagna. Uma mudança na configuração dos parâmetros do GP pode mudar essa situação, e esse

estudo será realizado no futuro.



(a) Precisão

(b) Revocação

Figura 4.3: Gráfico da evolução da precisão (a) e revocação (b) do melhor indivíduo na base Movielens 1M apara o GUARD (*A Genetic Unified Approach for Recommendation*)

4.4.2 Agregação de *Rankings*

O comportamento das técnicas de agregação de *ranking* na base *Movielens 1M* são similares aos obtidos com a *Movielens 100k*(veja Tabela 4.11). Assim como no experimento passado, os valores de novidade e diversidade melhoram em relação ao SVD quando o *borda count* foi utilizado para combinar os resultados do GUARD-P4, as custas de uma diminuição nos valores de precisão e revocação. Para a abordagem baseada em votação, os valores de precisão e revocação não se alteram e há um ganho de novidade e diversidade em relação ao GUARD-P4. Ao combinar os indivíduos do GUARD-Lex4 (Tabela 4.12) o comportamento é semelhante, e os *rankings* obtidos combinando o GUARD-Lex4 com votação apresentam maior novidade e diversidade que o ranking produzido pelo indivíduo retornado pelo algoritmo.

4.4.3 Análise dos Indivíduos

Nessa seção apresentamos uma análise dos melhores indivíduos e dos terminais mais representativos ao longo da evolução. Para o GUARD-P2, o melhor indivíduo foi a

Tabela 4.11: Resultado comparando o baseline e o GUARD-P4 com a agregação de *rankings* na base Movielens 1M.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
	P@20	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória	0,0186	0,3712	0,8232	0,8315
PureSVD	0,0321	0,6427	0,8298	0,8881
GUARD-P4	0,0303 ± 0,0002 ▼	0,6056 ± 0,003 ▼	0,7762 ± 0,017 ▼	0,8422 ± 0,009 ▼
BordaCount	0,0243 ± 0,0008 ▼	0,4859 ± 0,016 ▼	0,8601 ± 0,040 ▲	0,9420 ± 0,037 ▲
Votação	0,0303 ± 0,0002 ▼	0,6056 ± 0,003 ▼	0,8056 ± 0,010 ▼	0,8687 ± 0,006 ▼

Tabela 4.12: Resultado comparando o baseline e o GUARD-Lex4 com a agregação de *rankings* na base Movielens 1M.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
	P@20	**R@20**	**EPC@20**	**EILD@20**
Baseado em Memória	0,0186	0,3712	0,8232	0,8315
PureSVD	0,0321	0,6427	0,8298	0,8881
GUARD-Lex4	0,0303 ± 0,00005 ▼	0,6069 ± 0,001 ▼	0,7686 ± 0,001 ▼	0,8374 ± 0,002 ▼
BordaCount	0,0298 ± 0,0002 ▼	0,5968 ± 0,004 ▼	0,7756 ± 0,002 ▼	0,8510 ± 0,004 ▼
Votação	0,0303 ± 0,00005 ▼	0,6068 ± 0,001 ▼	0,7997 ± 0,0008 ▼	0,8665 ± 0,002 ▼

função:

$$r_{ai} = \log((T5_{ai}/T9_a)) - \sqrt{3.24} - \sqrt[8]{T14\_1} - \sqrt{(1.4 - T4_i + (T12\_1_a)^2)} \tag{4.6}$$

onde $T5_{ai}$ é o *baseline estimates* definido no Capítulo 2, $T9_a$ é o número de itens avaliados pelo usuário a, $T14\_1$ é o somatório da similaridade cosseno do item i com os itens mais semelhantes, $T4_i$ é o fator b_i do *baseline estimates* e por fim o $T12\_1$, o somatório da similaridade cosseno entre o usuário a e os usuários mais semelhantes.

Para o experimento com o GUARD-P4, o melhor indivíduo é dado por:

$$r_{ai} = \log(2.35)/\sqrt[8]{(log(T10_i)/T2_i)}/T2_i \times (3.97 \times T12\_1_a)^2/T2_i \tag{4.7}$$

onde $T10\_1_i$ é o número de usuários que avaliaram um item, $T12\_1_a$ e $T2_i$ são o somatório da similaridade cosseno e a média de avaliações do item i, respectivamente, como já foi definido na equação anterior.

No experimento GUARD-Lex4, o melhor indivíduo em precisão e revocação é dado pela equação:

$$r_{ai} = (T12\_1_a)^2 + (\log(T5_{ai}) \times (T12\_1_a)^2)/\sqrt{T12\_1_a} \tag{4.8}$$

e o de melhor novidade e diversidade por:

$$r_{ai} = \sqrt{T12\_1_a} \times (T12\_1_a)^2 + (\log(T5_{ai}) \times ((T12\_1_a)^2 + (\log(T5_{ai}) \times (T12\_1_a)^2)/((T12\_1_a)^2 + (T12\_1_a)^2)))/(\log(T12\_1_a) \times \sqrt{T12\_1_a}) \quad (4.9)$$

Nesses dois últimos casos, a diferença dos resultados entre eles é bem pequena, e observa-se que os terminais utilizados são os mesmos, mudando apenas a construção da equação. Em termos gerais, os terminais nas melhores funções não mudaram muito em relação aos melhores indivíduos da *Movielens 100k*. Os terminais $T12\_1$ e $T2$ continuam sendo terminais frequentes nos melhores indivíduos. No caso da base *Movielens 1M*, o terminal $T5$ aparece com mais frequência nos melhores indivíduos.

Para se ter uma ideia dos terminais presentes nos demais indivíduos, a Figura 4.4 mostra o gráfico da porcentagem de terminais ao longo da evolução. Assim como na base *Movielens 100k*, um grupo de terminais são rapidamente descartados pelo arcabouço.

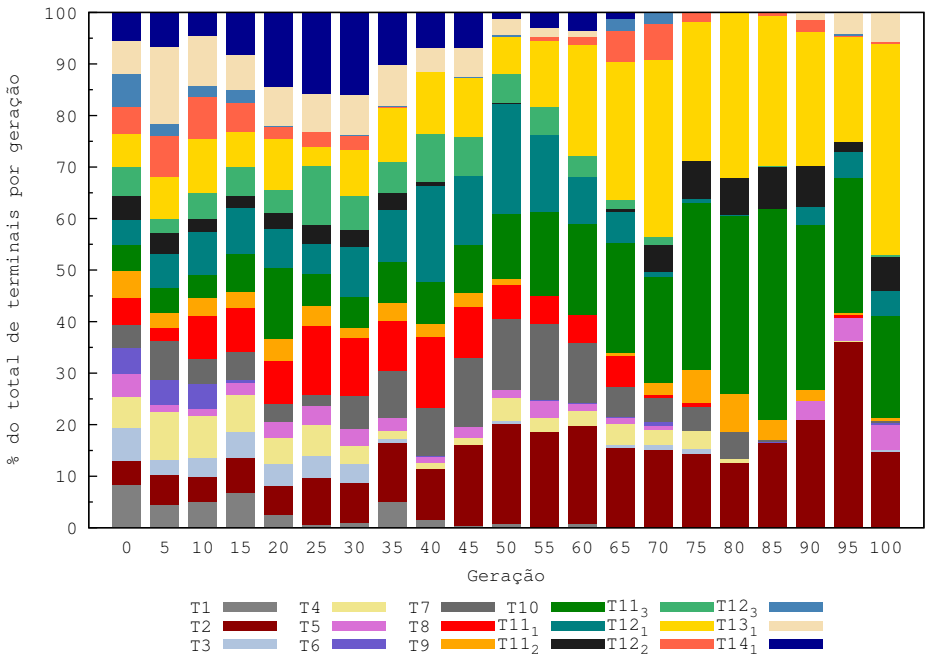


Figura 4.4: Número de terminais ao longo das gerações na base Movielens 1M

Observe que alguns terminais são eliminados rapidamente, como por exemplo o $T12\_2$, $T1$ e $T12\_3$. Já outros se mantém ao longo de toda evolução, como os terminais $T10$, $T11\_1$, $T11\_2$ assim como o $T5$, $T2$ e o $T12\_1$ que aparecem nos melhores indivíduos.

4.4.4 Generalização das Funções Evoluídas

Para testar a capacidade de generalização das funções produzidas pelo GUARD-P4, a função do melhor indivíduo foi testada com outras base de dados como a *MovieLens 100k* e a base da *LastFM*. A Tabela 4.13 apresenta o resultado da generalização para a base *Movielens 100k* do melhor indivíduo encontrado quando usada a base *Movielens 1M*. Observe que mesmo sem ter usado a base em questão na evolução no arcabouço, os resultados foram próximos ao baseline e semelhantes aos resultados obtidos quando utilizamos as funções encontradas para esta base.

Tabela 4.13: Resultado do melhor indivíduo encontrado pelo GUARD-P4 na base *Movielens 1M* quando testado na base *Movielens 100k*.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
Baseado em Memória	0,0197	0,3939	0,7705	0,7944
PureSVD	0,0266	0,5320	**0,7999**	**0,8807**
GUARD-P4	**0.0281**	**0.5622**	0.7212	0.8226

Em número de *hits*, o GUARD-P4 consegue colocar 9 itens relevantes a mais no top-20 do que o *PureSVD*. Esse valor representa um ganho de 3%. Para o caso da *LastFM*, o melhor indivíduo gera resultados próximos ao baseline, como listado na Tabela 4.14. Em número de *hits* a diferença é de 130 itens a mais no TOP-20 para o *PureSVD*, o que representa 5,4% no universo de 2.402 recomendações.

Tabela 4.14: Resultado do melhor indivíduo encontrado pelo GUARD-P4 na base *Movielens 1M* quando testado na base *LastFM*.

Algoritmo	**Precisão**	**Revocação**	**Novidade**	**Diversidade**
Baseado em Memória	0,0338	0,6753	0,8822	0,9410
PureSVD	**0.0408**	**0.8168**	**0.9113**	**0.9570**
GUARD-P4	0.0381	0.7627	0.8496	0.9222

4.4.5 Análise de Desempenho

A Tabela 4.15 mostra a análise de desempenho do arcabouço para a base *Movielens 1M*. O tempo de processamento também foi medido em três etapas: (1) Cálculo dos Terminais, (2) Evolução e (3) Recomendação. As duas primeiras etapas, que consomem mais tempo, são executadas *off-line*.

Um vez que a função quando encontrada tem sua aplicação direta. O tempo descrito na tabela corresponde a aplicação da função na base de testes conforme a

Tabela 4.15: Análise de desempenho do GUARD na base *Movielens 1M*

Experimento	**Cálculo dos Terminais**	**Evolução**	**Recomendação**	**Total**
SEED 0	09:31:32	09:19:58	00:02:26	18:53:56
SEED 1	09:11:19	10:24:13	00:02:32	19:38:04
SEED 2	09:36:25	10:24:51	00:02:27	20:03:43
SEED 3	09:30:56	11:42:37	00:02:30	21:16:03
Média	09:27:33	10:27:54	00:02:28	19:57:56

metodologia discutida no Capítulo 3. Para cada um dos 3.168 registros na base de testes, foram calculadas as preferências de 1001 usuário-item.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho propomos e testamos o GUARD (*A Genetic Unified Approach for Recommendation*), um arcabouço baseado em programação genética que gera funções de *ranking* de itens, resultando em recomendações de qualidade. O arcabouço desenvolvido é flexível ao tipo de recomendação, e avalia os indivíduos gerados com base em três métricas: número de *hits* (precisão e revocação), novidade e diversidade, seguindo duas abordagens diferentes de otimização dos objetivos: uma abordagem baseada em Pareto e uma abordagem lexicográfica. Como mencionado, os sistemas de recomendação são avaliados principalmente em relação a acurácia (precisão e revocação). No entanto, aspectos como novidade e diversidade devem ser considerados quando se pretende gerar recomendação de qualidade.

Considerar esses aspectos parte do princípio de que para usuários que estão começando a usar o sistema é muito mais interessante recomendar uma lista de itens com alto grau de acurácia, garantindo assim sua fidelidade ao sistema. Já para usuários mais antigos deve-se recomendar itens com um grau de novidade e diversidade maiores, mas mantendo sempre um grau de acurácia alto para manter o interesse do usuário em usar o sistema.

O arcabouço foi testado para filtragem colaborativa e aplicado no cenário de recomendação de filmes com as bases Movielens 100k e 1M. Na primeira base, os resultados de acurácia foram maiores do que os do baseline proposto gerando ganho de aproximadamente 7% em número itens nas primeiras posições do ranking. No caso dos indivíduos com melhor novidade e diversidade, apesar de uma pequena perda na acurácia, houve um ganho significativo em relação ao PureSVD nesses objetivos. Na segunda base, os resultados não superaram os do baseline proposto, mas ficaram em média 3% abaixo em número de itens relevantes nas 20 primeiras posições do *ranking*.

Para testar a generalização das funções encontradas, os melhores indivíduos foram

testados no contexto de recomendação de música com a base de dados da *LastFM*. Os resultados se mantiveram próximos ao do PureSVD, como um perda média também de 3%. Isso permite mostrar a generalização das funções geradas em base bem menores como a *Movielens 100k* se comparadas com a *LastFM*. Desta forma, podemos aplicar as funções em outros contextos onde o uso da evolução do GP é inviável por conta do tamanho da base de dados.

A grande vantagem de aplicar as funções encontradas está em sua simplicidade. A sua aplicação é direta e garante resultados melhores (no caso da *Movielens 100k*) e próximos (no caso da *Movielens 1M* e *LastFM*) se comparados com o PureSVD, estado da arte na tarefa de recomendar lista de itens. As funções geradas pelo GUARD-P4 serão muito interessantes para recomendação de itens a usuários novos no sistema, enquanto aquelas geradas pelo GUARD-Lex4 poderiam ser mais apropriadas para usuários experientes.

Direções futuras baseadas nos resultados obtidos até o momento sugerem que os resultados para a $Movielens1M$ ainda podem ser melhorados, principalmente com um melhor estudo do comportamento do algoritmo de programação genética e seus parâmetros. No entanto, técnicas de amostragem inteligentes devem ser aplicadas para que o custo computacional do algoritmo permita testar diversas configurações de parâmetros. Uma outra linha que pode ser promissora é como escolher as melhores funções de *ranking* da população final do GUARD e combiná-las através de técnicas de agregação de *rankings*.

Além disso, pretende-se preparar o arcabouço para realizar recomendações baseadas em conteúdo e/ou híbridas. Experimentos preliminares já foram realizados com terminais que utilizam informação de conteúdo, mas os resultados não foram melhores que os obtidos com a abordagem de filtragem colaborativa. Uma outra proposta de continuação deste trabalho é aplicar o arcabouço para encontrar funções que, ao invés de gerarem um lista de itens, determinam a nota que usuário daria a um item. Essa tarefa, conhecida como predição de *ratings*, requer uma nova definição de terminais e de outras métricas de avaliação dos indivíduos. Também seria interessante investigar a aplicação do arcabouço para evoluir funções em outros contextos como a própria *LastFM*. Por fim, pretende-se também utilizar no arcabouço outros algoritmos de aprendizado multiobjetivo como o SPEA2 e NSGA2, e analisar o impacto desses algoritmos nos resultados quando comparados com as técnicas utilizadas nesta versão.

Referências Bibliográficas

Adomavicius, G. & 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.

Anand, D. & Bharadwaj, K. K. (2010). Adaptive user similarity measures for recommender systems: A genetic programming approach. *3rd IEEE International Conference on Computer Science and Information Technology*, 8:121–125.

Bäck, T.; Fogel, D. & Michalewicz, Z., editores (2000). *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing.

Banzhaf, W.; Nordin, P.; Keller, R. E. & Francone, F. D. (1997). *Genetic Programming: An Introduction (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann Publishers.

Breese, J. S.; Heckerman, D. & Kadie, C. M. (1998). Empirical analysis of predictive algorithms for collaborative filtering. Em *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52.

Burke, R. D. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

Cacheda, F.; Carneiro, V.; Fernández, D. & Formoso, V. (2011). Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5:2:1–2:33.

Cremonesi, P.; Koren, Y. & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. Em *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 39–46.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester.

Dwork, C.; Kumar, R.; Naor, M. & Sivakumar, D. (2001). Rank aggregation methods for the web. Em *Proceedings of the 10th International World Wide Web Conferences*, pp. 613–622.

Fan, W.; Gordon, M. D. & Pathak, P. (2005). Genetic programming-based discovery of ranking functions for effective web search. *Journal of Management Information Systems*, 21:37–56.

Fourman, M. P. (1985). Compaction of symbolic layout using genetic algorithms. Em *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 141–153.

Herlocker, J. L.; Konstan, J. A. & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *IEEE Jornal Information Retrieval*, 5(4):287–310.

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

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. Em *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426–434.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data*, 4(1).

Koren, Y.; Bell, R. M. & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Jornal Computer*, 42(8):30–37.

Koza, J. R. (1992). *Genetic Programming - On the Programming of Computers by Means of Natural Selection*. Complex adaptive systems. MIT Press.

Lacerda, A.; Cristo, M.; Gonçalves, M. A.; Fan, W.; Ziviani, N. & Ribeiro-Neto, B. (2006). Learning to advertise. Em *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 549–556.

McNee, S. M.; Riedl, J. & Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems. Em *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems*, pp. 1097–1101.

Pedro, L. R. & Takahashi, R. H. C. (2011). Modeling decision-maker preferences through utility function level sets. Em *Proceedings of the 6th International Conference On Evolutionary Multi-Criterion Optimization*, pp. 550–563.

Poli, R.; Langdon, W. B. & McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P. & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. Em *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 175–186.

Ribeiro, M. T.; Lacerda, A.; Veloso, A. & Ziviani, N. (2012). Pareto-efficient hybridization for multi-objective recommender systems. Em *Proceedings of the 6th ACM conference on Recommender systems*, pp. 19–26.

Ricci, F.; Rokach, L.; Shapira, B. & Kantor, P. B., editores (2011). *Recommender Systems Handbook*. Springer.

Ritzel, B. J.; Eheart, J. W. & Ranjithan, S. (1994). Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30(5):1589–1603.

Salakhutdinov, R.; Mnih, A. & Hinton, G. E. (2007). Restricted boltzmann machines for collaborative filtering. Em *Proceedings of the 24th International Conference on Machine Learning*, pp. 791–798.

Sarwar, B. M.; Karypis, G.; Konstan, J. A. & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Em *Proceedings of the 10th International World Wide Web Conferences*, pp. 285–295.

Shardanand, U. & Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth. Em *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 210–217.

Torres, R. d. S.; Falcão, A. X.; Gonçalves, M. A.; Papa, J. P.; Zhang, B.; Fan, W. & Fox, E. A. (2009). A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):283–292.

Vargas, S. & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. Em *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 109–116.

Yeh, J. Y.; Lin, J. Y.; Ke, H. R. & Yang, W. P. (2007). Learning to Rank for Information Retrieval Using Genetic Programming. Em *SIGIR 2007 workshop: Learning to Rank for Information Retrieval*.

Zitzler, E.; Laumanns, M. & Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. Relatório técnico, Technical Report 103, ETH.