LUCAS NUNES BARBOSA

DISTRIBUTED RECOMMENDER SYSTEMS ON AN OPPORTUNISTIC NETWORK ENVIRONMENT

LAVRAS – MG

2020

LUCAS NUNES BARBOSA

DISTRIBUTED RECOMMENDER SYSTEMS ON AN OPPORTUNISTIC NETWORK ENVIRONMENT

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

Prof. Dr. Tales Heimfarth
Orientador

Prof. PhD Jonathan Gemmell
Coorientador

LAVRAS – MG

2020

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Nunes Barbosa, Lucas.
Distributed recommender systems on an opportunistic network environment / Lucas Nunes Barbosa. - 2020.
59 p.

Orientador(a): Tales Heimfarth.
Coorientador(a): Jonathan Gemmell.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2020.
Bibliografia.

1. Opportunistic Networks. 2. Recommender Systems. 3. Mobile ad hoc Networks. I. Heimfarth, Tales. II. Gemmell, Jonathan. III. Título.

LUCAS NUNES BARBOSA

DISTRIBUTED RECOMMENDER SYSTEMS ON AN OPPORTUNISTIC NETWORK ENVIRONMENT
SISTEMAS DE RECOMENDAÇÃO DISTRIBUÍDOS EM UM AMBIENTE DE REDE OPORTUNISTA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

APROVADA em 29 de Fevereiro de 2020.

Prof. Dr. Tales Heimfarth	UFLA
Prof. PhD Jonathan Gemmell	DePaul University
Prof. Dr. João Carlos Giacomin	UFLA
Prof. Dr. Edison Pignaton de Freitas	UFRGS

Prof. Dr. Tales Heimfarth
Orientador

Prof. PhD Jonathan Gemmell
Co-Orientador

LAVRAS – MG
2020

*Dedico esta dissertação de mestrado à minha amada esposa **Flávia**, meu pilar de apoio em todos os desafios, a mulher que me desafia diariamente a me tornar uma pessoa melhor e minha companheira de vida.*

*Dedico aos meus pais **Inês e Marcos**, pessoas que ajudaram a construir meu caráter, a abraçar o mundo e não temer o desconhecido.*

*Dedico à minha querida avó **Aparecida**, que construiu esta família com muito suor e ventre molhado diariamente, uma lavadeira que merece seu nome gravado na dissertação de seu neto que se tornou mestre ao seguir seu exemplo.*

AGRADECIMENTOS

À **Vida**, o bem mais precioso que tenho, o qual tento aproveita-lo aos segundos, onde busco multiplica-lo impactando as pessoas ao meu redor e o eternizando através do meu legado.

À **Universidade Federal de Lavras**, instituição que possibilitou que eu conhecesse pessoas maravilhosas nacionalmente e internacionalmente, mudou a minha vida para sempre!

Ao **Prof. Dr. Tales Heimfarth** que me acolheu como seu orientado desde a primeira semana de faculdade. Foram cinco anos de aprendizado, onde iniciei como um garoto maravilhado com um novo mundo e agora encerro como um homem de família, trilhando o próprio caminho.

Ao **Prof. PhD Jonathan Gemmell**, meu mentor na nova jornada dos Estados Unidos e na mudança de área de Redes de Computadores para Inteligência Artificial. O **Gemmell** foi um marco na minha trajetória, me guiou do meu sonho de desenvolvimento de jogos para realidade de me tornar um pesquisador de verdade. Não olhou apenas para seus interesses e sim para o potencial do aluno, direcionando para voos sempre mais altos.

Aos membros da banca examinadora, **Prof. Dr. João Carlos Giacomin** e **Prof. Dr. Edison Pignaton de Freitas**, que tão gentilmente aceitaram participar e colaborar com esta dissertação. Ao **Giacomin** em especial por ter me acompanhado por alguns anos na jornada de pesquisa científica e por seus conselhos sempre pertinentes.

Ao amigo **Miller Horvath**, às disciplinas, trabalhos e pesquisas feitas em conjunto. Principalmente, ao tempo dedicado ao de tênis de mesa, onde tinhamos conversas constantes sobre a vida e nosso futuro. Seu conhecimento e companherismo foram fundamentais para aceitar à pesquisa em uma nova área e ganhar um amigo pra vida.

À minha mãe **Inês** e ao meu pai **Marcos** deixo um agradecimento especial, por todas as lições de vida ensinadas de forma diária. Tive uma infância sem presentes e festas em datas comemorativas. O que recebi, foi a vida e os ensinamentos de como vivê-la aproveitando o seu melhor. Sinto-me orgulhoso e privilegiado por ter pais tão especiais.

Ao meu irmão **Matheus**, apesar de nossas diferenças, foi o responsável por me mostrar que é possível ir além, de certa forma, foi um exemplo. Mostrou o caminho, mas não como trilha-lo, e sim como passar por cada obstáculo sem precisar cair.

À minha amada esposa **Flávia**, por todo amor, carinho, compreensão e apoio constante na nossa caminhada. Obrigado por permanecer ao meu lado, mesmo nas constantes mudanças sem roteiros e irmos juntos em direção ao novo, mesmo com medo. Obrigado por me fazer feliz diariamente e andar sempre ao meu lado. Com você eu me torno uma pessoa melhor a cada dia.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação, o meu sincero agradecimento.

Happiness is only real when shared
(Christopher McCandless)

RESUMO

Dispositivos móveis são comuns em todo o mundo, mesmo em países com acesso limitado à internet, inclusive quando desastres naturais intenrrompem o acesso a uma infraestrutura centralizada. Este acesso permite a troca de informações em um ritmo incrível e através de grandes distâncias. No entanto, essa riqueza de informações pode frustrar os usuários a medida que são inundados com dados irrelevantes ou indesejados. Os sistemas de recomendação ajudam a aliviar o peso para computar cada perfil. Este projeto apresenta uma nova abordagem de sistema de recomendação de filtragem colaborativa baseada em uma rede distribuída oportunista. Algoritmos de filtragem colaborativa são amplamente utilizados em muitos sistemas online. Geralmente, o computação desses sistemas de recomendação é realizado em um servidor central, controlado pelo provedor, exigindo conexão constante a internet para coletar e computar dados. Entretanto, em muitos cenários, essas restrições não podem ser garantidas ou mesmo desejadas. No sistema de recomendação proposto, os usuários compartilham informações por meio de uma rede oportunista independente de conexão a internet dedicada. Utilizando tecnicas de recomendação centralizada de filtragem colaborativa como base, avaliamos dois cenários simulados compostos por diferentes velocidades de movimento e parâmetros de troca de dados. Nossos resultados demonstram que, em um período de tempo relativamente curto, o sistema de recomendação distribuído em redes oportunistas podem obter resultados semelhantes a um sistema centralizado tradicional. Além disso, notamos que a velocidade com que o sistema de recomendação oportunista se estabiliza dependendo de vários fatores, incluindo densidade dos usuários, velocidade de movimento e padrões dos usuários, e estratégias de transmissão. Em trabalhos futuros analisaremos novas estratégias e conjuntos de dados, da mesma forma, aumentaremos o número de usuários adicionando diferentes cenários.

Palavras-chave: Redes Oportunistas. Sistema de Recomendação. Redes Móveis *ad hoc*.

ABSTRACT

Mobile devices are common throughout the world, even in counties with limited internet access and even when natural disasters disrupt access to a centralized infrastructure. This access allows for the exchange of information at an incredible pace and across vast distances. However, this wealth of information can frustrate users as they become inundated with irrelevant or unwanted data. Recommender systems help alleviate this burden. The project presents a novel collaborative filtering recommender system based on an opportunistic distributed network. Collaborative filtering algorithms are widely used in many online systems. Often, the computation of these recommender systems is performed on a central server, controlled by the provider, requiring constant internet connection for gathering and computing data. However, in many scenarios, such constraints cannot be guaranteed or may not even be desired. On the proposed recommendation engine, users share information via an opportunistic network independent of a dedicated internet connection. Each node is responsible for gathering information from nearby nodes and calculating its own recommendations. Using a centralized collaborative filtering recommender as a baseline, we evaluate three simulated scenarios composed by different movement speeds and data exchange parameters. Our results show that in a relatively short time, an opportunistic distributed recommender systems can achieve results similar to a traditional centralized system. Furthermore, we noticed that the speed at which the opportunistic recommender system stabilizes depends on several factors including density of the users, movement speed and patterns of the users, and transmission strategies. On future works we will analyze new strategies and datasets, likewise, we will increase the number of users on different scenarios.

Keywords: Opportunistic Networks. Recommender Systems. Mobile *ad hoc* Networks.

LIST OF FIGURES

Figure 3.1 – "The ONE"simulator downtown dense simulation. 28
Figure 3.2 – A portion view of three of the cities simulated on our second experiment. The city in the center is Lavras; the west side is Ribeirão Vermelho; the north portion is Perdões. 29
Figure 3.3 – Modules communication. 29
Figure 3.4 – The figure illustrates the organization of the framework. An important observation is that the four components will be constantly working independent of each other. User representation will collect information and model user profile. The communication module exchanges modeled profiles as handshake messages and selects the top n profiles to be send. The user profile ranking selects which profiles will be stored and discarded. The recommendation component employs profiles stored as the neighborhood and computes recommendations. 31
Figure 4.1 – Results for MovieLens Downtown Sparce Scenario. The plot presents the *nDCG* for the `MDowS` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 45
Figure 4.2 – Results for MovieLens Downtown Dense Scenario. The plot presents the *nDCG* for the `MDowD` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 46
Figure 4.3 – Results for MovieLens Campus Sparce Scenario. The plot presents the *nDCG* for the `MCamS` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 47
Figure 4.6 – Results for LastFM Downtown Sparse Scenario. The plot presents the *nDCG* for the `LDowS` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 47
Figure 4.4 – Results for MovieLens Campus Dense Scenario. The plot presents the *nDCG* for the `MCamD` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 48
Figure 4.5 – Results for LastFM Campus Dense Scenario. The plot presents the *nDCG* for the `LCamD` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted. 49

Figure 4.7 – Secondary buffer result across five different cities, considering three hours of constant movement. 51
Figure 4.8 – Scatter plot for the technique employing 5 profiles for the primary buffer and 5 profiles for the secondary buffer. 51
Figure 4.9 – Scatter plot for the technique employing 10 profiles for the primary buffer and no profiles for the secondary buffer. 52
Figure 4.10 – Scatter plot for the technique employing 10 profiles for the primary buffer and 5 profiles for the secondary buffer. 52

LIST OF TABLES

Table 4.1 – The ONE simulator parameters for communication interface, world map, routes, and vehicles. Values employing for the world was provided as default for simulating Helsinki city. Parameters used for created vehicles was an approximation of maximum and minimum speeds in general. 40

Table 4.2 – The ONE scenarios specific configurations for each scenario. Those configurations were employed as an attempt to emulate real-world scenarios as a campus and a downtown of a city. We increase the population on both scenarios in order to determine the behaviour of the framework with the increase of data. 41

Table 4.3 – Coverage results in different metrics employed on a data centralized item-based collaborative approach. 43

Table 4.4 – Coverage results in different metrics employed on a data centralized user-based collaborative approach. 43

CONTENTS

1 Introduction 13
2 Related Work 16
2.1 Opportunistic Networks 16
2.2 Recommender Systems 18
2.3 Hybrid Recommender Systems 20
2.4 Recommendations in Distributed Environments 23
3 Opportunistic Distributed Recommender System 26
3.1 Scenarios 27
3.2 System Overview 28
3.3 Users Representation 30
3.4 Communication 32
3.5 User Profile Ranking 34
3.6 Recommendation 36
4 Experimental Results 38
4.1 Evaluation Metric 38
4.2 Simulation Configuration 38
4.3 Dataset 42
4.4 Assessment of the Centralized Approach 42
4.5 Assessment of the Distributed Approach 44
4.5.1 Primary Buffer 44
4.5.2 Secondary Buffer 50
5 Conclusions and Future Work 53
REFERÊNCIAS 55

1 INTRODUCTION

Mobile devices enable consumers to access vast amounts of data unfettered from land-line connections. Recent data demonstrate that smartphones are more common in developed countries, with 72% ownership in the United States, and 41% in Brazil, compared with the global average of 43% (POUSHTER et al., 2016). Moreover, Internet of things (IoT) applications are becoming more common and are expected to experience growth with the support of fifth generation (5G) networks (LI; XU; ZHAO, 2018; MAVROMOUSTAKIS; MASTORAKIS; BATALLA, 2016; RAHIMI; ZIBAEENEJAD; SAFAVI, 2018). While this access to data across many platforms is a great boon, it also presents a challenge. A consumer is likely interested in only a fraction of the available content, and finding that content can be a time consuming and arduous task.

Recommender systems can help users find relevant content. Popular applications, such as Netflix, Spotify, and YouTube, employ recommenders to assist their customers find movies, music and even user-generated content. Applications such as these often clean the data, build user profiles and store relevant information for later use (HUANG; CHUNG; CHEN, 2004; RICCI et al., 2015; YANG; HWANG, 2013).

Many strategies have been proposed to construct recommendations. In academia there are three popular approaches, which are content-based filtering, collaborative filtering, and knowledge-based filtering (FELFERNIG et al., 2014). Content-based filtering (BALABANOVIĆ; SHOHAM, 1997; PAZZANI; BILLSUS, 2007) leverages the description of the items, such as genre or director, to find relevant items for a user. Collaborative filtering (EKSTRAND et al., 2011; SCHAFER et al., 2007), on the other hand, focuses on the user's interactions with the system, such as their ratings. This strategy relies on the notion that users which expressed similar tastes in the past will likely consume similar items in the future. Knowledge-based filtering (TREWIN, 2000) employs explicit domain knowledge to filter items, it does not build upon users' record. Therefore, it will not present a cold-start problem (FELFERNIG et al., 2011). This technique is usually employed to recommend items which are not regularly consumed (RICCI et al., 2015).

Generally, recommender systems are implemented on a centralized server. Users consume items provided by the application and the recommender system collects their information to generate a profile. When queried, the application presents the most relevant information to the user. As a consequence of this architecture, the application owner controls access to both

the user and content information. This centralized architecture has many advantages for the content provider, most notably control over the data.

However, adopting a centralized approach also has many drawbacks such as privacy, reliability, and scalability. One remarkable problem of having a central server is how to ensure customers that their recommendations are not manipulated. Likewise, there is no guarantee that the information collected is secure or would not be exploited without the user's consent. The reliability and scalability of the recommender system is also an important concern when the central architecture can be targeted by hackers or taken offline by natural disasters.

Consider customers of a recommender systems applications in one of these three scenarios: in communities where internet access is expensive or intermittent, in a city after an earthquake or tsunami, or during a political rally when users seek anonymity or the government is limiting internet access. In these cases, the user may not wish or may not be able to use a service based on a central server.

We propose a distributed strategy independent of internet connection, employing an opportunistic network architecture to distribute content among users. The system was designed having in mind the fundamental needs of a recommender system. Relevant information from the host user is collected, such as their rating for a song or a whether or not they read a news article. This information is aggregated to create a user model. Multiple user models are aggregated to create neighborhoods. From these neighborhoods, recommendations are produced.

However, the goal of our proposed opportunistic distributed recommender system is to distribute the burden of each step among the users rather than relying on a central server. Its framework is organized into four components with different steps 1) User Representation 2) Communication 3) User Profile Ranking and 4) Recommendation. The user representation will collect information from activities of users on their smartphones, then compresses it into modeled profiles, shielding private information. The second component is responsible for detecting nearby users (or their devices) and exchanging their profiles. The user profile ranking will select which profiles should be kept or discarded. Finally, the recommender will compute the metadata stored on the device's buffer to compute predictions and recommendations.

The framework has several benefits when compared to a traditional approach. First, it eliminates the need for infrastructural internet connection to communicate with a centralized server. Second, in a distributed approach, much of the data would be distributed throughout the

network and would, therefore, be resilient to loss, eliminating a point of failure. Each user models their own profiles, stores relevant information, and computes their own recommendations.

Our experimental experiments include simulations over a variety of scenarios. Two datasets were evaluated. Movielens includes rating applied by users to movies. LastFM includes the playlists and artist preference for users in the music domain. Users were simulated in a variety of ways including walking and riding in a car. The density of users in the simulation was set to model both rural and urban settings. The amount of information transmitted between users was manipulated to evaluate the trade-off between the size of the messages and the performance of the recommender. Our results showed that in certain scenarios the proposed framework can achieve results similar to a centralized architecture while operating solely on an opportunistic framework.

The paper is organized in five sections. First we introduced the research problem and the motivations of our framework. In the next section, we introduce related on different areas: opportunistic networks, recommender systems, and recommender systems in distributed environment. Section 3 discusses how the system will operate, from data modeling to generating recommendations. In section 4 we introduce the evaluation metrics, datasets, and simulations parameters. We then discuss the experimental results. Finally, in section 5 we discuss our findings and consider future work.

2 RELATED WORK

In this section we present related work. We begin with a discussion of opportunistic networks and then present a background of recommender systems. We then discuss the distributed recommender systems and opportunistic distributed recommender systems.

2.1 Opportunistic Networks

Device-to-Device (D2D) communication allows devices to exchange information directly without a network infrastructure (LEI et al., 2012). The communication between mobile devices are made either through Bluetooth, WiFi Direct or other WiFi interfaces. D2D communication was first proposed as a medium access control (MAC) protocol, to be applied into a mobile ad hoc network (MANET) with multi-channel access and multi-hop communication environment (LIN; HSU, 2000). Later, peer-to-peer (P2P) communication techniques were exploited into D2D architectures (LEI et al., 2012). The D2D paradigm improves network performance by enhancing spectral efficiency, and improving throughput, energy efficiency, delay, and fairness (ASADI; WANG; MANCUSO, 2014).

Opportunistic Network (OppNet) (PELUSI; PASSARELLA; CONTI, 2006) studies were motivated by advances on delay-tolerant networking (DTN) architecture (FALL, 2003) and routing (JAIN; FALL; PATRA, 2004). An OppNet deals with computer network scenarios with limited or no internet connection, where the connection between nodes are intermittent, or a path between nodes willing to communicate might not directly exist.

Recent works exploiting OppNets have been developed. In (ANASTASIADES; BRAUN; SIRIS, 2014), Anastasiades et al. explored Information-Centric and Content Centric Networking solutions into OppNet scenarios for content discovery and transfer. The PRoWait routing protocol, based on Spray and Wait and Prophet routing protocols, was proposed in (DHURANDHER et al., 2015). It improves the packet delivery ratio, hop count and latency in OppNets compared to the well-known Spray and Wait, Prophet and Epidemic protocols.

In (MARTÍN-CAMPILLO et al., 2013), several routing protocols are tested in realistic disaster scenarios in order to verify effectiveness on delivery and energy performances. In the experimental results, The MaxProp forwarding method presented the best delivery performance while TTR forwarding presented the lowest energy cost in almost all disaster scenarios simulated.

The focus in (IPPISCH; KÜPER; GRAFFI, 2018) is to analyze time syncing and buffer management of mobile devices in OppNets. Using *opptain*, a OppNet framework for mobile devices, data is exchanged in the network as *bundles* (IPPISCH; KÜPER; GRAFFI, 2018). *Bundles* keep essential time related information like creation time and Time-To-Live (TLL) (IPPISCH; KÜPER; GRAFFI, 2018). It is important for devices in the OppNet to be synced in order to keep time contextuality of data (IPPISCH; KÜPER; GRAFFI, 2018). Furthermore, personal mobile devices usually have a general purpose buffer shared by applications, personal data and the OppNet data (IPPISCH; KÜPER; GRAFFI, 2018). Therefore, an important concern in OppNets is to not demand too much of users' device storage (IPPISCH; KÜPER; GRAFFI, 2018). The authors proposed a method to bypass the time syncing problem using a method that keeps the difference of time between devices from the sender to the destination (IPPISCH; KÜPER; GRAFFI, 2018). *Bundles* drop policies were also proposed to deal with limited storage problem (IPPISCH; KÜPER; GRAFFI, 2018).

Smart-phones with high computational power and several wireless network interfaces are commonplace today. This adoption allows the applications OppNets into complex distributed scenarios. Distributed computing is proposed in (CONTI et al., 2010), which is defined as a step forward in OppNets where users would be able to opportunistically exploit any resource available in network, including other users' devices, in a trustable and secure way. Moreover, a survey in opportunistic mobile social networks is presented in (JEDARI; XIA, 2013), where the main discussion topics were: mobile social networks characteristics, human mobility models, dynamic community detection methods, and routing and data dissemination protocols.

A novel approach to building a context representation is proposed in (UNGER et al., 2016). These contexts are based on users' mobile phone sensors data (UNGER et al., 2016). Unsupervised deep learning techniques and PCA are applied in the users' data to learn latent contexts, represented by numeric vectors (UNGER et al., 2016). An Android application was developed to evaluate the proposed approach, which recommends points of interest, e.g., restaurants and entertainment centers, retrieved from Foursquare API (FOURSQUARE, 2018). The authors in (UNGER et al., 2016) compared 4 state-of-art recommender models, which are: (1) matrix factorization model without contextual information, (2) Explicit content model, (3) The proposed latent context model, (4) Hybrid context model, which considers both explicit and latent context information; using RMSE, Hit@K, average ranking, and nDCG as evaluation

metrics. The results show that the proposed approach provides up to 20% accuracy improvement (UNGER et al., 2016).

2.2 Recommender Systems

The recommender systems, first proposed as collaborative filtering in (GOLDBERG et al., 1992), was born from the necessity of filtering relevant content from the increasing amount of information available on internet (RICCI; ROKACH; SHAPIRA, 2011). Recommenders aim to deliver personalized list of interesting suggestions to its users (BOBADILLA et al., 2013). Those items might be content from many domains, such as songs, movies or books.

Recommender systems received a great deal of attention after the Netflix Prize (BENNETT; LANNING et al., 2007) was announced. This challenge, proposed by Netflix in 2006, made their dataset of anonymous movie ratings available and awarded a grand prize to the solution that provide at least 10% accuracy improvement compared to their own system (BENNETT; LANNING et al., 2007).

According to Burke's taxonomy (BURKE, 2002), recommender techniques are divided into four classes: Collaborative Filtering, Content-Based, Demographics, and Knowledge-Based; Collaborative Filtering being one of the most popular recommender techniques. More recently, another technique class has risen in recommender system studies and applications, the Context-Based (or Context-Aware) Recommender Systems (ADOMAVICIUS; TUZHILIN, 2015). Two or more of those recommender techniques are often combined into Hybrid Recommenders (BURKE, 2007) in order to achieve greater performances on either accuracy or computational cost, or even to deal with classic recommender challenges such as sparsity or the cold-start problem.

Collaborative filtering recommenders leverage the similarity between items or the similarity between users. They often rely on ratings data, focusing on the interaction of the users with the items rather than the content of the items.

In user-based collaborative filtering, ratings are predicted for items users have not consumed based on how similar users have rated those items (DESROSIERS; KARYPIS, 2011). Following the Tapestry (GOLDBERG et al., 1992) proposal, some early works adopted and evolved user-based collaborative filtering techniques by proposing new collaborative systems (RES-

NICK et al., 1994) and by evaluating and comparing different user-based recommender algorithms (BREESE; HECKERMAN; KADIE, 1998).

A collaborative recommender based on Facebook data was combined with content-based and expert-based approaches into an hybrid recommender, called TastedWeights (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012). This approach recommends music through an interactive user interface that discloses recommendation process details and allows users to control it.

In (ZANOTTI et al., 2016), A model-based collaborative filtering is proposed, based on Continuous Bag of Words (CBOW) and Skip-gram neural network topologies were applied to build distributed representations and extract semantic relationships between data from movie domain. The distributed representation was trained using three types of data, which are: users rating data (ratings users assigned to movies), user assigned tags data (tags users assigned to movies), and movie specific data (e.g., actors, directors, genre) (ZANOTTI et al., 2016). The users ratings and assigned tags data was retrieved from MovieLens 10M dataset (RIEDL; KONSTAN, 1998) and the movie specific data was collected from Internet Movie Database (IMDb) though its interface (IMDB, 2018).

Finally, a weighted hybrid RecSys was built combining all the proposed models and traditional collaborative filtering models (ZANOTTI et al., 2016). The evaluation results show a substantial accuracy improvement of the model-based approaches compared with the memory-based ones, using RMSE as the accuracy metric (ZANOTTI et al., 2016). Furthermore, the hybrid approach outperformed all standalone models (ZANOTTI et al., 2016).

Recommenders are often designed with the aim at tackling specific challenges, such as scalability, sparsity and the cold-start problems. A recommender usually holds data from several users and items. All this data must be processed in order to generate proper recommendations, which may cause scalability problems since the model demands computational resources and, depending on the model, it may periodically need additional computation to update the model. In addition, even the most active users might rate only a fraction of the available items on the platform. This sparsity may cause the system to fail the recommendation task due to a lack of information. The utmost sparsity problem occurs when there is no information at all about a user or an item, usually caused by the addition of new users or items in the system. This scenario is often called the cold-start problem.

Collaborative Filtering and Fuzzy C-Means Clustering are combined in (VERMA; MITTAL; AGARWAL, 2013) to deal with sparsity and scalability in a movie domain, which MovieLens dataset (RIEDL; KONSTAN, 1998) was adopted. The C-Means is applied to cluster movies by their genre (VERMA; MITTAL; AGARWAL, 2013). So, the scalability is increased since the system applies Collaborative Filtering separately in each movie cluster, reducing the amount of data to be matched with each other (VERMA; MITTAL; AGARWAL, 2013). Also, the sparsity, and even the cold-start problem, is eased since it uses the movie cluster to estimate ratings when it is unknown (VERMA; MITTAL; AGARWAL, 2013). The evaluation of the proposed recommender, using MAE, showed that C-Means overcome the classic K-Means in this context and that it is able to handle with sparsity and cold-start problem (VERMA; MITTAL; AGARWAL, 2013).

2.3 Hybrid Recommender Systems

The main focus of Hybrid RecSys is to provide recommendations with higher accuracy. Hybrid recommender techniques are adopted to build the so-called TastedWeights (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012), a music RecSys with an interactive user interface that discloses recommendation process details and allows users to control it. This approach first builds recommenders from three data sources: Facebook (collaborative and social), Wikipedia (content-based and semantics), and Twitter (expert-based) (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012). The systems hybridization combines these three models using three different hybrid methods: weighted, mixed, and cross-source (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012). The accuracy of the approach proposed in (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012) was evaluated using a utility metric based on Breeze's R-Score (BREESE; HECKERMAN; KADIE, 1998). The results show that all the hybrid approaches overcome the standalone ones, being Cross-Source the best among the hybrid strategies (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012). Moreover, the interaction methods overcome hybrid approaches (BOSTANDJIEV; O'DONOVAN; HÖLLERER, 2012).

A novel approach to build a context representation is proposed in (UNGER et al., 2016). These contexts are based on users' mobile phone sensors data (UNGER et al., 2016). Unsupervised deep learning techniques and PCA are applied in the users' data to learn latent contexts, represented by numeric vectors (UNGER et al., 2016). An Android application was developed

to evaluate the proposed approach, which recommends points of interest, e.g., restaurants and entertainment centers, retrieved from Foursquare API (FOURSQUARE, 2018). The authors in (UNGER et al., 2016) compared 4 state-of-art recommender models, which are: (1) matrix factorization model without contextual information, (2) Explicit content model, (3) The proposed latent context model, (4) Hybrid context model, which considers both explicit and latent context information; using RMSE, Hit@K, average ranking, and nDCG as evaluation metrics. The results show that the proposed approach provides up to 20% accuracy improvement (UNGER et al., 2016).

In (ZANOTTI et al., 2016), Continuous Bag of Words (CBOW) and Skip-gram neural network topologies were applied to build distributed representations in order to extract semantic relationships between data from movie domain. The distributed representation was trained using three types of data, which are: users rating data (ratings users assigned to movies), user assigned tags data (tags users assigned to movies), and movie specific data (e.g., actors, directors, genre) (ZANOTTI et al., 2016). The user rating and assigned tags data was retrieved from MovieLens 10M dataset (RIEDL; KONSTAN, 1998) and the movie specific data was collected from Internet Movie Database (IMDb) though its interface (IMDB, 2018). Then, user-based and item-based collaborative filtering models were built based on both CBOW and Skip-gram distributed representations (ZANOTTI et al., 2016). Finally, a weighted hybrid RecSys was built combining all the proposed models and traditional collaborative filtering models (ZANOTTI et al., 2016). The evaluation results show a substantial accuracy improvement of the hybrid approach compared with all standalone models, using RMSE as the accuracy metric (ZANOTTI et al., 2016).

Hybrid approaches are often applied to tackle specific challenges, such as scalability, sparsity and the cold-start problems. A RecSys usually holds data from several users and items. All this data must be processed in order to generate proper recommendations, which may cause scalability problems since the model demands a lot of computational resources to be build and, depending on the model, it may need additional high-priced computation to update the model from time to time. In addition, even the most active users rates only few of the available items in the RecSys. So, this sparsity aspect sometimes causes the system to fail the recommendation due to lack of information. The utmost sparsity problem is when there is no information at all about a user or an item, usually caused by the add of new users or items in the system, which is the so-called cold-start problem.

In (NIU et al., 2016), a Feature Combination Hybrid RecSys is proposed aiming at dealing with sparsity problem. A novel rating prediction strategy is proposed, which train a classifier, adopting Random Forest in this case, to forecast ratings based on three information, which are: items side information features, such as movie data from IMDb (IMDB, 2018) or tags extracted from online platforms using Natural Language Processing (NLP); users profiles made by their favorite items; the ratings users have already assigned to items (NIU et al., 2016). The proposed prediction strategy is then applied to predict all the missing ratings in the data (NIU et al., 2016). The new user vector is used to compute users similarity, using traditional similarity metrics, as cosine, adjusted cosine, and correlation-based, and also a novel similarity metric, the so called Customer Relative Interest (CTRI) (NIU et al., 2016). Finally, the hybrid system, called FUIR, combines item and user-based collaborative filtering to compute the top-N predicted items (NIU et al., 2016). The experimental evaluation in (NIU et al., 2016) was conducted using three datasets, which are: MoveiLens (RIEDL; KONSTAN, 1998), Book-Crossing (ZIEGLER et al., 2005), and LastFM (LASTFM, 2018). The results suggests that FUIR algorithm effectively deals with sparsity of data and provides high quality recommendations according to MAE evaluation (NIU et al., 2016).

In (YIN et al., 2013), The LCARS, a location-content-aware RecSys, is proposed to generate recommendation, with venues and events as the items, to people that are traveling. In this context, the user-item matrix is very sparse since a user can visit only few places in a limited space range (YIN et al., 2013). Furthermore, the system has no local historic information about users who are traveling to new cities (YIN et al., 2013). To deal with this issues, the LCARS exploits location and content information about the spatial items though the proposed LCA-LDA offline model, which learns user interests and city local interest (YIN et al., 2013). The online step computes the top-k recommendations, optimized by an extension of Threshold Algorithm (YIN et al., 2013). In the experimental evaluation, Foursquare and DoubanEvent datasets were adopted, using Google Maps API to partition spatial items into cities and Natural Language Processing toolkits to extract content words from the events summary and description in DoubanEvent dataset only (YIN et al., 2013). The proposed model was compared with several traditional and state-of-art models, such as User interest, social and geographical influences, Category-based k-Nearest Neighbors Algorithm, Item-based k-Nearest Neighbors Algorithm, LDA, Location-Aware LDA, and Content-Aware LDA (CA-LDA) (YIN et al., 2013). Accor-

ding to the experimental evaluation, using Recall@k as the effectiveness measure, the proposed LCARS significantly overcomes the other models (YIN et al., 2013).

Considering the huge amount of data in the Big Data era, some effort have been made to use Hybrid RecSys to ease processing time and computational resource allocation to compute all this data.

In (VERMA; PATEL; PATEL, 2015), the Hadoop framework (WHITE, 2012) was exploited to provide scalability, distributed and fault-tolerance RecSys to run in low-cost hardware by using MapReduce (DEAN; GHEMAWAT, 2008) programming model. The RecSys model in (VERMA; PATEL; PATEL, 2015) was built using Mahout architecture (OWEN; OWEN, 2012). The evaluation, made using different sizes of MovieLens dataset (RIEDL; KONSTAN, 1998), show that the execution time do not increase at the same ratio as the data size.

2.4 Recommendations in Distributed Environments

Taking advantage of smart-phones with high computational capabilities, decentralized approaches in recommender systems have recently attracted attention. In (TAO, 2015), a mobile advertising recommender is proposed to perform distributed large scale recommendations in real-time. The users are grouped by their location and there is one recommender to meet each group separately (TAO, 2015). The communication between all recommenders is made through high speed networks (TAO, 2015). The distributed system is built using Hadoop framework (WHITE, 2012), which provides tools for performance and low latency applications (TAO, 2015).

A distributed cloud-based service recommendation system is presented in (GANCHEV; JI; O'DROMA, 2015). Based on the "always best connected and best served" (ABC&S) paradigm, arising from the emerging ubiquitous consumer wireless world (UCWW), this systems aims at providing a personalized list of preferred mobile services considering contexts related to users, services and network (GANCHEV; JI; O'DROMA, 2015).

In (KERMARREC et al., 2010), both user-based and item-based collaborative filtering algorithms are compared in a decentralized environment. Moreover, a user-based random walk approach for decentralized systems, which is designed to deal which sparse data, has been proposed (KERMARREC et al., 2010). The approach is evaluated with different sparsity, similarity measures and neighbourhood configurations (KERMARREC et al., 2010). The authors analyze

the effects of the most active users in the random walk approach, which tend to intensify the influence of those user over users with less ratings (KERMARREC et al., 2010). Adopting the MovieLens 10M dataset (RIEDL; KONSTAN, 1998), the experimental evaluation suggests that the proposed algorithm outperforms other collaborative filtering techniques in precision considering distributed P2P environments, where user-based techniques perform better than the item-based ones (KERMARREC et al., 2010).

A decentralized collaborative filtering based news dissemination system, called WhatsUp, is presented in (BOUTET et al., 2013). This system is central authority independent (BOUTET et al., 2013). WhatsUp builds an implicit social network, where users are clustered based on the ratings (like or dislike) they assigned to news they have received (BOUTET et al., 2013). The authors also proposed a novel heterogeneous gossip protocol, which biases the information dissemination based on users' taste and amplifies the dissemination of popular news items (BOUTET et al., 2013). The proposed system was compared against social cascades and distributed collaborative filtering schemes, outperforming them in terms of precision, recall and harmonic mean using three different datasets: a synthetic dataset, a real dataset crawled from Digg, and a news survey dataset conducted by the authors (BOUTET et al., 2013).

In (YANG; HWANG, 2013), the iTravel system is proposed. iTravel is a cost-effective travel recommender system that provides on-tour attraction recommendations to tourists (YANG; HWANG, 2013). The users' rating information is exchanged through P2P communication, achieving infrastructure-free and cost-free convenience in communication tasks (YANG; HWANG, 2013). The authors proposed three approaches of data exchange: (1) Unconditional — all the ratings lists is exchanged between close users; (2) Preference-based — only ratings lists that are similar to the recipient's rating list is sent; and (3) Hybrid — combines the other two approaches aiming at users with atypical taste who can be prevented from acquire enough rating data for recommendations in the preference-based approach (YANG; HWANG, 2013). The experimental results showed that the data propagation methods improved recommendation accuracy over the systems without data propagation (YANG; HWANG, 2013). Also, a user evaluation was conducted, showing that iTravel can provide useful and proper recommendations to support travel decisions (YANG; HWANG, 2013).

Based on collaborative filtering and the decentralized matrix completion algorithm, a decentralized recommender is proposed in (WANG et al., 2015). In this context, users could only exchange limited data with users that are nearby and data is computed locally, providing

privacy, scalability, and robustness benefits (WANG et al., 2015). Using average percentile score, which is based on the average ranked order, as the evaluation measure, the experimental results showed that the proposed recommender achieve similar performances compared with other state-of-art recommender techniques (WANG et al., 2015).

In (ARNABOLDI et al., 2013), opportunistic sensing services are exploited to perceive context and social information in order to support a mobile social network, called *DroidOppPathFinder*, for people who enjoy outdoor physical activities. In *DroidOppPathFinder*, people share information about paths for physical activities (ARNABOLDI et al., 2013). Combining users' interest and environmental data collected by the sensor, the proposed solution recommends the best path in a specific area (ARNABOLDI et al., 2013).

In (ARNABOLDI et al., 2017), a tag-based recommender, called *PLIERS*, is exploited into a decentralized environment, where data exchange is made through D2D communications only. This framework is called *Pervasive PLIERS* (*p-PLIERS*) (ARNABOLDI et al., 2017). The proposed framework was evaluated using synthetic and real data in three different scenarios: (1) users attending to Expo2015; (2) users attending a scientific conference; and (3) users during a working day in the city of Helsinki (ARNABOLDI et al., 2017). The experiments showed that, in an opportunistic environment, *p-PLIERS* can generate recommendations that are as effective as the ones achieved in a centralized approach (ARNABOLDI et al., 2017).

A location-privacy recommender using opportunistic networks is proposed in (ZHAO; YE; HENDERSON, 2016) aiming to mitigate privacy issues related to centralized recommender systems. In this context, data is also exchanged through short-range communications and the recommendations are locally generated (ARNABOLDI et al., 2017). The experimental evaluation showed that the decentralized recommender performance is close to the centralized one after proper data are acquired (ARNABOLDI et al., 2017). Also, a reputation scheme is proposed that is able to decrease the success ratio of sampling attacks (ARNABOLDI et al., 2017).

3 OPPORTUNISTIC DISTRIBUTED RECOMMENDER SYSTEM

In this section, we present the framework for a distributed collaborative filtering recommender on mobile devices employing only D2D communications to exchange messages among users. The proposed framework builds users profiles by collecting information from their applications activities. Then, those profiles may be exchanged among users using an opportunistic network. Later, the application will process the collected data to compute recommendations, using collaborative filtering techniques. The opportunistic distributed recommender system suggests several benefits over a centralized approach. The system would be trustworthy, resilient, scalable, independent of internet connection, and free of a single controlling entity.

A trustworthy recommendation from a centralized system is not guaranteed. Since the server is controlled by a private company, there is no assurance recommendations are not manipulated. Moreover, private companies can exploit the collected information to recommend certain items. Further, the collected data are stored on their servers. The knowledge of users activities can be applied on different areas, without the user's permission. Our distributed system retains critical information on the owners device. The device will restrict the data to be exchanged among users. Therefore, users will be in full control of their personal information.

The resilience of the system is remarkable on distributed systems; the data is distributed among users and will not present a single point of failure. The service can only be suspended when all users are removed from the network. In contrast, a centralized approach presents a dependency of servers, under the control of a single entity. Therefore, the stored data of both users and items can be withheld at any time.

Centralized recommender systems can present scalability and maintenance problems. In order to achieve optimal recommendations, a high number of profiles is necessary and computing it for each user has a high computation cost. Moreover, providing this service for millions of users can be expensive. In the proposed framework, the burden of computing each profile would be distributed among users. The greatest burden for a user in this framework is the necessity of exchanging messages regularly in order to renew the database. However, this burden can be prioritized or suspended based on the user's needs. Thus, the distributed framework can provide a dynamic and low cost alternative for the computation and storage of data.

Considering that the proposed framework adopts an opportunist network as its exclusive method to exchange messages, it is expected that the system would have similar advantages common among other opportunist networks. The framework is independent of internet connec-

tion and may occasionally if at all connect to a centralized server. Consequently, this approach is particularly desirable in regions where the internet connection is scarce and unreliable. However opportunist networks have some drawbacks. A network based on message passing between devices must include enough interactions between users to allow for the dissemination of information. As more people join the network it becomes more reliable. However, if the framework is deployed in areas where people are isolated, users may not have enough information to produce useful recommendations.

3.1 Scenarios

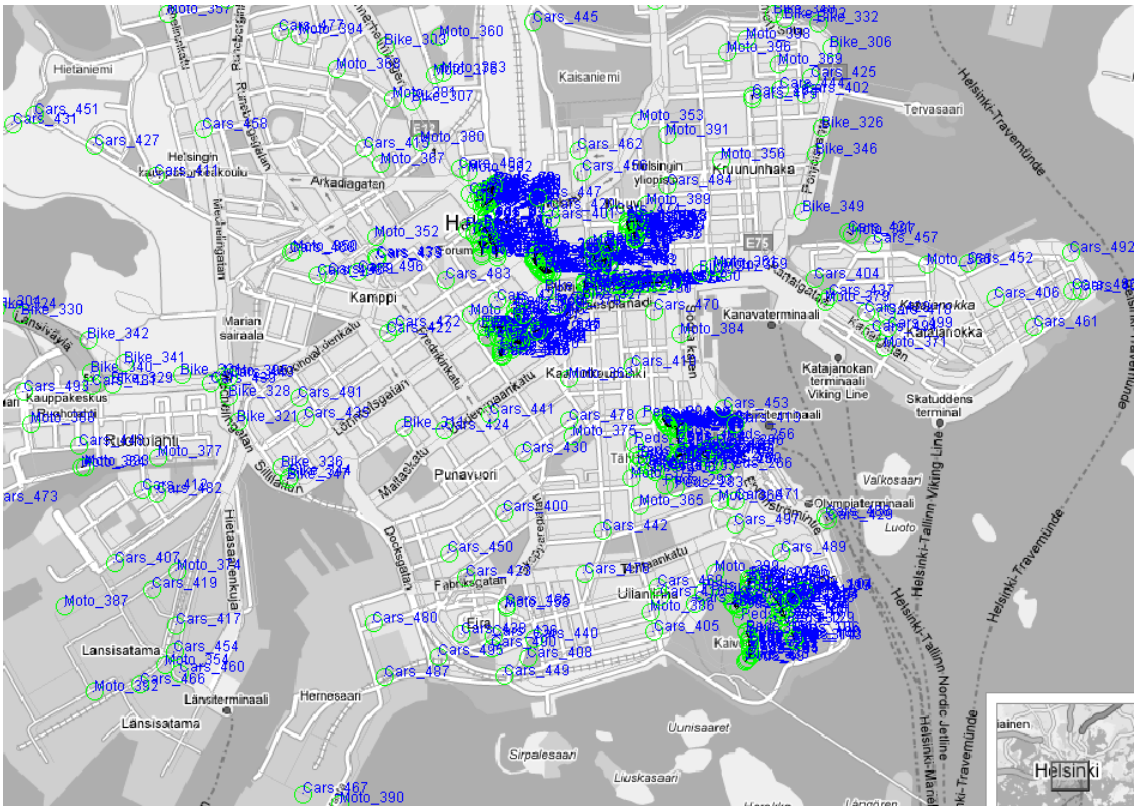
Generally opportunistic networks applications are designed for emergency scenarios, since they are infrastructure-less. Analogous to this condition, we have examples of underdeveloped communities could have limited internet access due to high prices or intermittent providers. Despite of the obvious advantage of use on those scenarios, a recommender system independent of internet connection could be suitable for daily proposes, especially in less developed countries. Our solution aims on building a local knowledge of the population and recommend the most suitable content for each individual.

The proposed framework was designed as a community building a shared database, where the data is stored locally on devices and maybe shared among members. The data collection will occur locally on each member device, selecting exactly which information would be shared among the network. When those users are close in distance, the framework would automatically connect their devices, sharing selected data during the connection time. After a period of time, the members belonging to this community could have shared enough information to have construct a reliable recommendation system. As new users are included in the system, the recommendations will become more accurate and reliable.

Alternatively, consider people during a political rally seeking for information and priming for the anonymity. The system would be capable of selecting relevant information and distribute among participants. Therefore, the gathered data would be temporary and related to that event. Furthermore, the recommendations processed on the circumstance would only be stored on users related to it.

Figure **??** represents an agglomeration of people on the downtown of a city. This scenario might portray pedestrians at a concert in the center of a city. The vehicles outside the radius

Figure 3.1 – "The ONE"simulator downtown dense simulation.



Source: Author

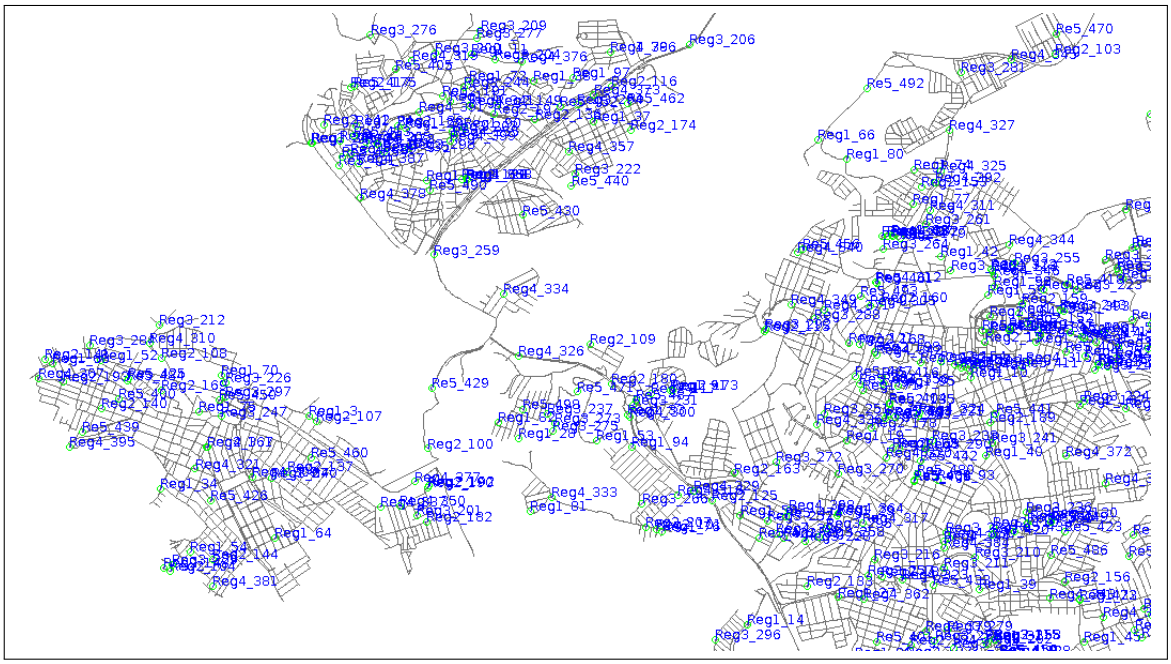
of the concert would be traveling around the city. Therefore, general information collected on crowded regions can be distributed by high speed nodes.

A remarkable detail on the first scenario presented was the agglomeration on the central area, clearly benefiting our proposed framework. In Figure 3.2 represents a realistic scenario with different cities with the population distributed randomly. The population contains people on bikes, motorcycles, cars, or even on foot. Having the population with sparse distribution and lower speed, individual users would not perform enough trades to acquire suitable profiles. A specialized approach was designed to answer those needs. The framework will be presented in details in Section 4.2 with the specifics configurations for each designed scenario.

3.2 System Overview

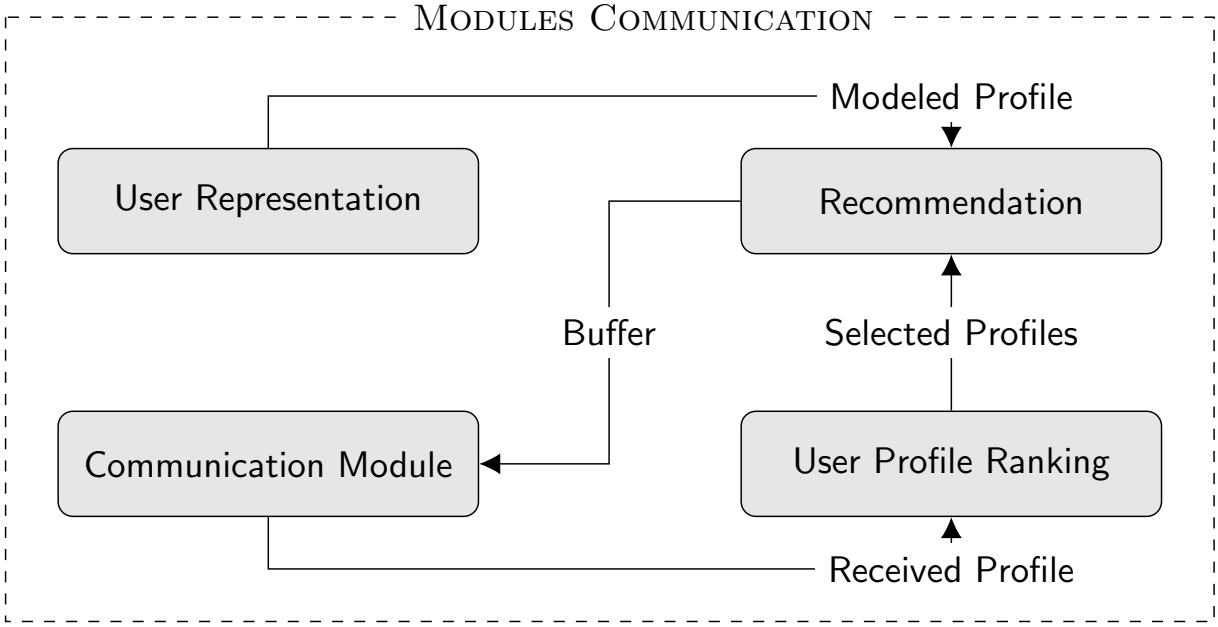
In this section we illustrate the key modules of the proposed framework. The opportunistic distributed recommender system is organized into four components: 1) Users Representation, 2) Communication, 3) User Profile Ranking, 4) Recommendation.

Figure 3.2 – A portion view of three of the cities simulated on our second experiment. The city in the center is Lavras; the west side is Ribeirão Vermelho; the north portion is Perdões.



Source: Author

Figure 3.3 – Modules communication.



Source: Author

Figure 3.3 illustrates how each component communicates with one another. In order to produce recommendations the system will select the most relevant profiles to be stored on each

device's buffer. The buffer is the storage for users profiles, that will separated on a primary buffer, containing host's neighbors profiles and the secondary buffer, containing relevant information for the network. Therefore, user representation module computes the host metadata to build its profile. The modeled profile is stored on the buffers, the host and stored profiles will be exchanged by the communication module. Later, the user profile ranking will select the most relevant profiles to be stored on the primary buffer and the remaining data will selected according to the secondary buffer strategy. The function of each module operates is highlighted in Figure 3.4. Those modules are described in depth in the following sections.

3.3 Users Representation

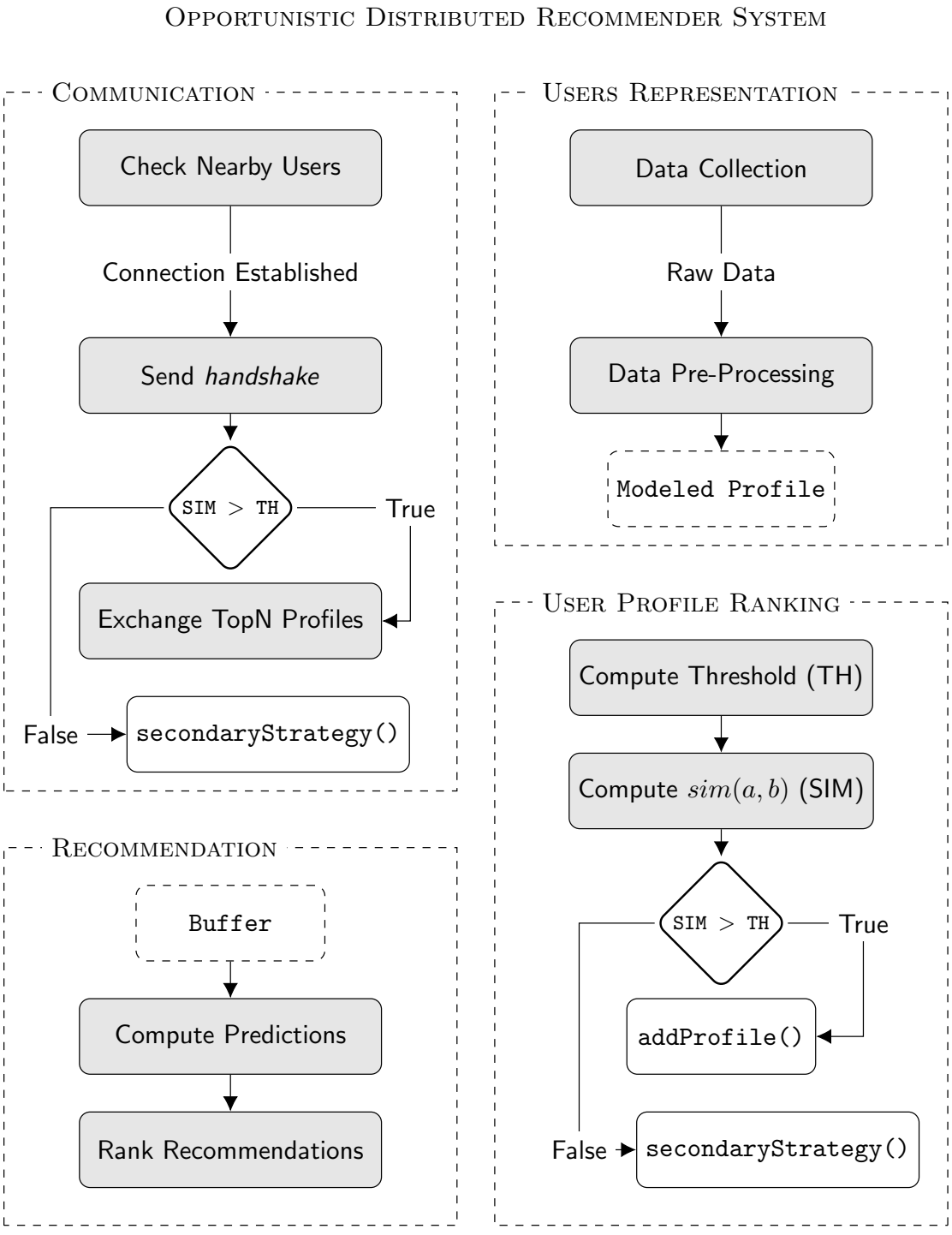
Algorithm 1: User Representation

```
Function UserRepresentation(Buffer hostBuffer)
    while onItemConsumption() do
        storeOnDevice(metadata);
    end
    if onScheduler() then
        hostProfile = dataPreprocessing(storedData());
        updateBuffer(hostProfile);
    end
Function dataPreprocessing(Profile hp)
    if hp contains r_<u,i> then
        userModel = [hp.id_u, hp.id_i, hp.r_<u,i>];
        return userModel;
    end
    else
        ratings = computeRating(hp);
        userModel = [hp.id_u, hp.id_i, ratings];
        return userModel;
    end
```

The user representation component is responsible for collecting and modeling the users' information. Therefore, the system will oversee the user's activities and store metadata for the application. From this data, the application will construct the user profiles.

Gathering certain information could cause privacy concerns due to the distribution of user data across the network. Therefore, the framework collects that information and creates an anonymous profile to be shared. A random user ID is generated and employed as the key to access the user's profile.

Figure 3.4 – The figure illustrates the organization of the framework. An important observation is that the four components will be constantly working independent of each other. User representation will collect information and model user profile. The communication module exchanges modeled profiles as handshake messages and selects the top n profiles to be send. The user profile ranking selects which profiles will be stored and discarded. The recommendation component employs profiles stored as the neighborhood and computes recommendations.



Source: Author

The system will store data from all users entertainment activities. The component is aware of their actions and will store only necessary data. For example, when a user is listening to a song, the system verifies its history and updates or adds the item to the user profile. The application may store its title, artist, plays, and timestamp. At this stage the data can be cleaned and pre-processed. The application may clean noisy data, resolve redundancy, or remove outliers. Later, these processed profiles can be used to create a suitable user model for the recommender system.

Data pre-processing is any process performed on raw data to transform into another format. Generally, real world datasets are noisy, incomplete and inconsistent. Moreover, they contain errors, missing values, outliers, and discrepancies in values. Major tasks in data preprocessing are data cleaning, integration, transformation, and reduction.

Algorithm 1 describes how the users representation module will capture the metadata and generates a user model. Anytime a user consumes an item, the framework will store metadata related to that item. Periodically, the system will build the user model, later employed the user modeled profile. If stored items had a given rating, it will store the information directly. Otherwise, the information will be processed to be suitable for the model.

A user model is defined by an user id and a vector representation of users over the item space. Therefore, a user u is represented by a tuple (id_u, U), being id_u a randomly generated id and U a set of items that the users u has rated. The set U is composed by item tuples $(id_i, r_{<u,i>})$, where id_i stands for the item id and $r_(u,i)$ stands for the rating user u assigned to item i.

3.4 Communication

An opportunistic networks is a mobile *ad hoc* network on which the communication between nodes is temporary and within a small radius, communicating not further than walking distance. In *ad hoc* communications, nodes can communicate without the aid of an access point. Generally, nodes are mobile, communication paths are not predetermined, and the messages can be exchanged among the nodes towards its destination.

For example, imagine a user who wants to send an important message, but the internet connection is not available. The network was designed to take advantage of mobile nodes. Therefore, when a node want to send a message to a destination, it will search for the most likely neighbor to deliver the message to its destination. Each node receiving that message will

Algorithm 2: Communication module.

```
Function Communication(pBuffer hostPBuffer, sBuffer hostSBuffer, Message
 message)
    pBuffer = getHostProfile(hostPBuffer);
    listProfiles = idsProfiles(hostPBuffer);
    if onSender() then
        if onContact() then
            sendHandshake(pBuffer, listProfiles);
        end
        if onReceivedRequest() then
            if requestedNeighbors then
                sendNRequestedProfile(pBuffer);
            end
            else
                sendAllProfiles(sBuffer);
            end
        end
    end
    if onReceiver() then
        receivedProfile = message.senderProfile;
        if RankProfiles(receivedProfile) is valid then
            requestTopNProfiles();
        end
        else
            requestSBufferProfiles();
        end
        if onReceivedProfiles() then
            for p in message.profiles do
                RankProfiles(p);
            end
        end
    end
```

verify which neighbor is the most appropriate to be the next hop. The message will be stored on the device during a period of time, if there is no encounter, the packet will be deleted eventually. Opportunistic networks are usually implemented on scenarios of emergency, where the internet connection is not a possibility and information have to be sent to a person, company or to a base station.

In our proposal, an opportunistic network is employed to create a distributed recommender system. As presented on Section 3.3, the framework will build a modeled profile for each user. Then, the communication module will be responsible for detecting users to exchange profiles. In order to establish the communication without a wireless router, we adopt the Wi-Fi

Direct technology. The closest interface to Wi-Fi Direct is the Bluetooth. Comparing these two technologies, there is a large differences in range and transmission speed. The Wi-Fi Direct offers speeds of up to 250Mbps and has a stated maximum distance of more than 200 metes, compared to Bluetooth that offers up to 25Mbps in speed and has a maximum distance of 100 meters (ALLICENCE, 2018a; ALLICENCE, 2018b; DIDELES, 2003). Range and speed are components that provide a most efficient implementation of a distributed recommender system. Alternatively, the framework could employ 5G technology to improve its performance.

The communication component sends periodic broadcast messages to establish communication between nearby mobile devices (Algorithm 2). When an encounter occurs, a handshake message is exchanged, containing the host profiles and a ID list of profiles stored on the primary buffer. As sender, on the first contact the handshake message will be sent and the device will wait for a request of profiles. When the request is received, the module will sent the N most similar profiles stored that are not in the ID list of profiles received. As receiver, the module will check if the received profile is a potential neighbor, which the Ranking Module will compute, it will be explained how to compute the similarity on Section 3.5. If the Ranking Module stored the received profile, it will request the top N profiles that are not on the buffer. Later, it will receive those profiles, adding most relevant on the buffer and selecting from the remaining profiles which should be stored on the secondary buffer accordingly to the strategy, otherwise, it will be discarded. The ID list of profiles is used to remove items that are already in possession of the receiver, avoiding the exchange of redundant information. Profiles will be resent if the stored profile is outdated. Scenarios was taken from (OpenStreetMap contributors, 2017).

3.5 User Profile Ranking

Employing collaborative filtering technique on a memory-constrained device presents many unique challenges. The amount of leveraged information and the quality of the recommendation are often directly proportional. Therefore, limiting the number of stored profiles could be a significant drawback on the accuracy of the system. The ranking module was designed to reduce this handicap by selecting the most relevant information for each device.

User-based collaborative filtering computes the similarity between users, then selects the best neighbors for each user. It then employs the knowledge shared from its neighbors to make a recommendation. The user ranking module reproduces the collaborative filtering algorithm on demand, storing only the profiles of the k most relevant neighbors for that user. Therefore, the

Algorithm 3: Rank module.

```
Function RankProfiles(Profile receivedProfile, Buffer pBuffer)
    h = getHostProfile(pBuffer);
    th = similarity(h, lastProfile(pBuffer));
    if (similarity(h, receivedProfile) > th) then
        addBuffer(pBuffer, receivedProfile);
        sortBySimilarity(pBuffer);
        if nUser(pBuffer) > bufferSize then
            discard(lastUser(pBuffer));
        end
        return true
    end
    else
        secondaryStrategy(receivedProfile);
        return false
    end
```

size of the user's neighbor buffer will be set as the number of neighbors used in the collaborative filtering algorithm.

In order to calculate the similarity between users, the user ranking module employs the cosine similarity (Equation 3.1) (ABELLO; PARDALOS; RESENDE, 2013), which proved to be the most efficient for our system in the previous work (BARBOSA et al., 2018). However, the standalone metric did not produce optimal recommendations. Therefore, the system employs a threshold in order to improve it. The users needs at least 10 items in common to be considered a relevant neighbor, not discarding the profile, but lowing its tier.

$$Cosine_{sim}(A,B) = \frac{\sum_{i=0}^{n} A_i B_i}{\sqrt{\sum_{i=0}^{n} A_i^2}\sqrt{\sum_{i=0}^{n} B_i^2}} \tag{3.1}$$

The user rank module is responsible for ranking and sorting stored profiles according to their relevance, then discarding obsolete ones. Profiles will be discarded if it meets one of these following conditions: the buffer is full and the received profile similarity is lower than the least similar profile stored, or the profile is outdated. Since our system aims for daily use application, profiles stored would be temporary. When a modeled profile is received, the framework will calculate the similarity between host profiles and receive profile, if it is greater than the threshold (least similar stored profile), it will be stored, otherwise, the profile will be discarded, or stored on a secondary buffer

The secondary buffer was implemented due the nature of recommender systems and opportunistic networks. The majority of users will consume popular items, this is a fact. Howe-

ver, those who have consumed peculiar items will be affect by a communication protocol that prioritize most relevant information. Furthermore, users which did not exchanged enough information due to lack of movement, consequently, lack of encounters with an abundance of user profiles, will not have most suitable profiles stored on their devices.

The strategy of a secondary buffer is to carry relevant information to users in need. Therefore, before discarding dissimilar profiles the communication module will store a portion of those profiles accordingly to a strategy in order to delivery relevant information to grey sheep. The communication module is explained step by step on Algorithm 3, the different techniques for the secondary buffer will be discussed in depth on the Section 4.

3.6 Recommendation

Algorithm 4: Recommendation.

```
Function Recommend (Buffer hostBuffer)
    h = getHostProfile(hostBuffer);
    for p in hostBuffer do
        I_u = itemsRatedBy(p)
        s = score(p, h, I_u)
        recommendations += (p, s);
    end
    sortByScore(recommendations);
    return recommendations;
```

The last component is responsible for computing the recommendations based on profiles collected in the system's buffer. Collaborative filtering techniques assumes that users who consumed an item in the past, will consume similar items in the future. It creates a neighborhood of similar items or users, then employs the knowledge of the past to recommend items. Moreover, the recommendation component will employ the collected data as the user neighborhood and make predictions based on the items and ratings that his neighbors have consumed.

When these recommenders rely on user similarity, they are called user-based collaborative filtering (UBCF). Given a user u and item i a predicted rating $p(u,i)$ is produced by identifying a neighborhood, N, of the k most similar users and leveraging their ratings, $r(n,i)$, on the item:

$$p(u,i) = \frac{\sum_n^N sim(u,n) \cdot r(n,i)}{\sum_n^N sim(u,n)} \tag{3.2}$$

Algorithm 5: Recommendation

```
Function Recommend(Buffer hostBuffer, Integer N)
    h = getHostProfile(hostBuffer);
    P = getStoredProfiles(hostBuffer);
    I_h = itemsRatedBy(h)
    I_c = ∅
    for p in P do
        I_p = itemsRatedBy(p)
        I_c = I_c ∪ (I_p − I_h)
    end
    I_r = ∅
    for i in I_c do
        I_r = I_r + predictRating(h, i, P)
    end
    sortItemsByRating(I_r)
    recommendations = topNItems(I_r, N)
    return recommendations;
```

In our framework, the recommendation component receives the k most relevant profiles from previous units, limiting its responsibility to apply the adopted method and to compute the recommendations. The computation is performed in a periodic manner (Algorithm 5). In order to recommend an item for the host, the recommendation module will employ all profiles stored on his buffer. The module will create a list of items the host did not consume but the host neighbors did (I_c), which are the items to be recommended. Ratings the host would assign to items in I_c are predicted based on users which are similar to the host and the ratings they have assigned those items, following Eq. 3.2. Then, the list is sorted by the predicted ratings, defining the most relevant recommendations that will be presented to the host.

4 EXPERIMENTAL RESULTS

In this section, we discuss the methods employed to evaluate the distributed recommender system. The following section presents the metric employed to evaluate the system. Section 4.2 deliberates on scenarios and setups of each simulation. On section 4.3 we explain which datasets were employed, and what techniques were exploited to produce modeled profiles for each user. Finally, we discuss the obtained results on each scenario, using two datasets.

4.1 Evaluation Metric

Our proposed framework was evaluated based on its ability to predict a user's preference for an item with regard to the ranking and the trade-off between the cost of exchanging n profiles with the quality of the recommendations. In this work, results were reported employing the normalized discounted cumulative gain (*nDCG*).

The *nDCG* (JÄRVELIN; KEKÄLÄINEN, 2002) (Equation 4.2) measures the performance of a recommendation system based on the graded relevance of the recommended entities.

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{4.1}$$

$$nDCG_k = \frac{DCG_k}{iDCG_k} \tag{4.2}$$

Where k is the maximum number of entities that can be recommended, the rel_i is the importance of that prediction, $iDCG_k$ is the maximum possible (ideal) DCG for a given set of ratings.

4.2 Simulation Configuration

In order to simulate a real-world environment we created distinct movement simulations on The ONE simulator (KERäNEN; OTT; KäRKKäINEN, 2009). The application is capable of simulating mobile nodes in different speed and movement model, further, it generates routing reports of delivering messages. Therefore, we can create different scenarios where users exchange messages. The simulator allows us to create groups with different behaviours, so we permuted groups setups in order to create 30 different simulations.

Each user profile was divided equally across five partitions, employing four folds to train and build the recommender, and the fifth one to test. First, we simulate the path of users, their encounters and message exchanged. Then, the report generated by The ONE was employed as input of encounters. On the framework, it will check the occurred encounters and exchange messages between users. The user representation component is the only one which will be reproduced before the simulation starts, producing each user profiles. Experiments were performed five times, each using a different training set as users profile content and their respective testing set.

In order to compute the *nDCG*, we calculated the metric employing the following steps. First, we trained the system on each device with the profiles stored on buffer, emphasizing that only information from the training set of that fold would be available for the computation of the similarity among users. Then, the framework attempts to recommend 20 items for that user, rating and then ranking each one of them. The *iDCG* is calculated using the top 20 rated profiles on the corresponding testing set and is calculated considering the ranking of items. The observed rating is employed to calculate the *iDCG* and *DCG*.

The ONE simulator has a default scenario representing the city Helsinki. It contains four different routes: roads, main roads, pedestrian paths, and malls. We design our simulations to use the default scenario as baseline. Wheres nodes can have different speeds and movement models, we create four different vehicles: pedestrians, bicycles, motorcycles, and cars. Therefore, we could generate different simulations selecting specifics regions for each node category. For the communication module, we set the transmission speed of 100 Mbps, the transmission range of 30 m, and the network interface of WiFi Direct. Those information are presented on Table 4.1

The first scenario is intended to reproduce a university campus. Imagine a campus where students would be moving around to perform their activities, their smartphones would be exchanging information with their college to compute recommendations later on. The set of students is divided into two groups: some stationary (e.g. taking their classes) and others with movement (walking or even running). We represented those students as pedestrians `(P)`, they can walk on every location of the campus.

The second scenario intends to reproduce a city downtown. People would be walking close to malls or in pedestrian paths around the city. On the other hand, high speed vehicles would be moving on main roads. Therefore, the pedestrians `(P)` had their walking path limited

Table 4.1 – The ONE simulator parameters for communication interface, world map, routes, and vehicles. Values employing for the world was provided as default for simulating Helsinki city. Parameters used for created vehicles was an approximation of maximum and minimum speeds in general.

Key	**Value**	**Key**	**Value**
Map Size	4500 m X 3400 m	Interface	WiFi Direct
Transmit Speed	100 Mbps	Transmit Range	30 m
Movement Model	MapBased	Buffer Size	10 profiles
Route 1 (R1)	Roads	Route 2 (R2)	Main Roads
Route 3 (R3)	Pedestrian Paths	Route 4 (R4)	Mall
Pedestrian (P)	0-5.4 Km/h	Bicycle (B)	7.5-23 Km/h
Motorcycle (M)	9-40 Km/h	Cars (C)	10-50 Km/h

Source: Author

to routes 3 and 4, bicycles (B) were limited to routes 2 and 3, motorcycles (M) were limited to routes 1 and 4, and cars (C) were limited to routes 1 and 2

Those scenarios were designed in order to illustrate the progressive enhancement behavior of the distributed system on crowded and non-crowded areas. Since on campus area the nodes maximum speed would be $5.4km/h$, the aid of exchanging multiple profiles becomes more clear. On downtown scenario, the aid of high speed nodes to distribute information among distant nodes. Likewise, we expected that different densities would affect the quality of the recommendation, considering that a higher number of users represents a high number of encounter and possible similar neighbors. Therefore, on each scenario we have population of 200 and 500 nodes, evaluating the recommender progression on different densities.

The simulations for of scenarios considered a time period of 60 minutes, generating recommendations every 5 minutes. A small timelapse for each recommendation was preferable due to fast convergence of recommendations. The simulations were performed using 10 as the maximum number of profiles stored on buffer, disregarding the host. The experiments were executed in order to identify the ideal number of messages that should be exchanged on each encounter. We simulated different exchanging methods. On each encounter, users could exchange 0, 2, 5, or 10 profiles per encounter (n). Therefore, when an encounter occurs at least the sender profile would be exchanged ($n = 0$). Then, if the similarity of the sender and receiver is greater than the threshold, the receiver will request n profiles from the buffer. In order to obtain statistically relevant results, each experiment was executed 30 times. The configuration of each scenario are presented in Table **??**.

Table 4.2 – The ONE scenarios specific configurations for each scenario. Those configurations were employed as an attempt to emulate real-world scenarios as a campus and a downtown of a city. We increase the population on both scenarios in order to determine the behaviour of the framework with the increase of data.

Scenario	**Nodes**	R1	R2	R3	R4
Campus Sparse (CamS)	200 P1	✓	✓	✓	✓
Campus Dense (CamD)	500 P1	✓	✓	✓	✓
Downtown Sparse (DowS)	120 P			✓	✓
	20 B		✓	✓	
	20 M	✓			✓
	40 C	✓	✓		
Downtown Dense (DowD)	300 P			✓	✓
	50 B		✓	✓	
	50 M	✓			✓
	100 C	✓	✓		

2gray!5gray!15

Source: Author

Subsequently, we have built a scenario that would represent our routine. The maps made available by Open Street Map (OpenStreetMap contributors, 2017) were employed as baseline for the creation of a second map to The ONE simulator. We recreated the Brazilian's suburban cities such as Lavras, Perdões, Itumirim, Ribeirão Vermelho, and Nepomuceno. Those are small cities from the state of Minas Gerais. The highways which connects those cities were removed. Instead, we added small bridges. Therefore, each city would have just one way in and out, except Lavras, that was putted in the center of the map, connecting everything (as represented on Figure 3.2).

The objective of building the a realistic scenario was to expose the flaws of having segregated groups on each city and the lack of information collected by the main technique to grey sheep users. The regions will have a specific niche of users, since the majority of them will have encounters with people from their own region. Consequently, a secondary buffer strategy would be necessary in order to improve recommendation for stationary users and to those ones which consume peculiar items.

In Section 4.5.1 will be discussed the trade-off exchanging βN profiles on each encounter. The optimal number of profiles stored and exchanged are key to develop a functional distributed recommender system. In Section 4.5.2 we deliberate alternatives to support grey sheep users. Due to the nature of opportunistic networks and recommender systems, users which have peculiar item consumption will undoubtedly have disadvantages on their recommendations. Therefore, we propose a secondary buffer strategy in order to help them.

4.3 Dataset

The simulations employed users from the "Last.fm Dataset - 1K users" (LASTFM, 2018; CELMA, 2010) and "MovieLens 100K Dataset" (HARPER; KONSTAN, 2016). As discussed in Section 3.3 some data processing was needed. LastFM dataset contains artist name, track name, user ID, artist ID, track ID, and timestamp. On the other hand, MovieLens contains user id, item id, rating, and timestamp. User-based collaborative filtering requires user ID, items ID, and ratings of each profile to produce recommendations.

Considering the amount of information provided from LastFM, removing missing values, redundancy, and unnecessary information is only the first step. To compute the preference for an artist, we adopted, after exhaustive empirical evaluation, the logarithm min-max normalization:

$$rating_a = \frac{\log p_a - \log min_{p_u}}{\log max_{p_u} - \log min_{p_u}} \tag{4.3}$$

Given an artist a and a user u, the rating is calculated considering the number of plays that artist had (p_a), the most and least listened artist by u ($max(p_u)$ and $min(p_u)$).

The data transformation technique (Equation 4.3) was applied on the dataset adjusting values measured on different scales to a common scale. The new ratings range between $[1,5]$.

The logarithm transformation was applied on the number of plays in order to scale down their values, since the artists who people like the most have a much higher value than others. Then, the min-max normalization technique is employed to define the ratings limits, having the most and least favorite artists as its min and max.

On MovieLens the given ratings were provided. We removed any redundancies and missing values. Then, we removed users which did not consume at least 10 items. Those users were remove because our work does not dealing with cold start problems.

4.4 Assessment of the Centralized Approach

In order to select the best method and metrics for our distributed approach, the collaborative filtering methods were first appraised using a centralized approach. The user-based and item-based collaborative filtering algorithm were evaluated considering its coverage, the percentage of items the system was able to recommend. We employed different metrics to com-

pute the similarity between users. The results of our experiments are presented in Tables 4.3 and 4.4.

Table 4.3 – Coverage results in different metrics employed on a data centralized item-based collaborative approach.

	Metric	K5	K10	K20	K50	K80
2gray!5gray!15	Manhattan	0.111	0.159	0.212	0.291	0.337
	Euclidean	0.176	0.239	0.308	0.402	0.451
	Chebyshev	0.250	0.336	0.422	0.525	0.571
	Cosine	0.336	0.413	0.493	0.557	0.609

Source: Author

Table 4.4 – Coverage results in different metrics employed on a data centralized user-based collaborative approach.

	Metric	K5	K10	K20	K50	K80
2gray!5gray!15	Manhattan	0.082	0.181	0.342	0.578	0.715
	Euclidean	0.129	0.217	0.386	0.609	0.723
	Chebyshev	0.384	0.507	0.599	0.719	0.776
	Cosine	0.556	0.644	0.711	**0.776**	0.794

Source: Author

Both collaborative filtering algorithms presented the same relation regarding the metrics. The cosine similarity metric performed better for all neighboring configuration in both methods, followed by the Chebyshev, Euclidean and Manhattan distances. Therefore, the Cosine metric was chosen as the metric for our distributed approach.

Considering the neighborhood k, as k increases so does the Coverage with diminishing returns. The results presented in $k50$ and $k80$ were only slightly different. In order to save memory space of the mobile devices in the distributed approach, we decided to fix $k50$ as the maximum number of neighbors.

On this dataset, the user-based collaborative filtering performed better than the item-based filtering for all tested metrics, which led to the decision to use the user-based method in our distributed approach. It should also be noted that the sharing of a user profile (stored on the user device) is simpler than sharing item profiles (potentially stored across multiple devices) thus making user-based collaborative filtering a more natural choice in this framework. The results presented here are used as a base comparison for the distributed approach, whose results are introduced in the next section.

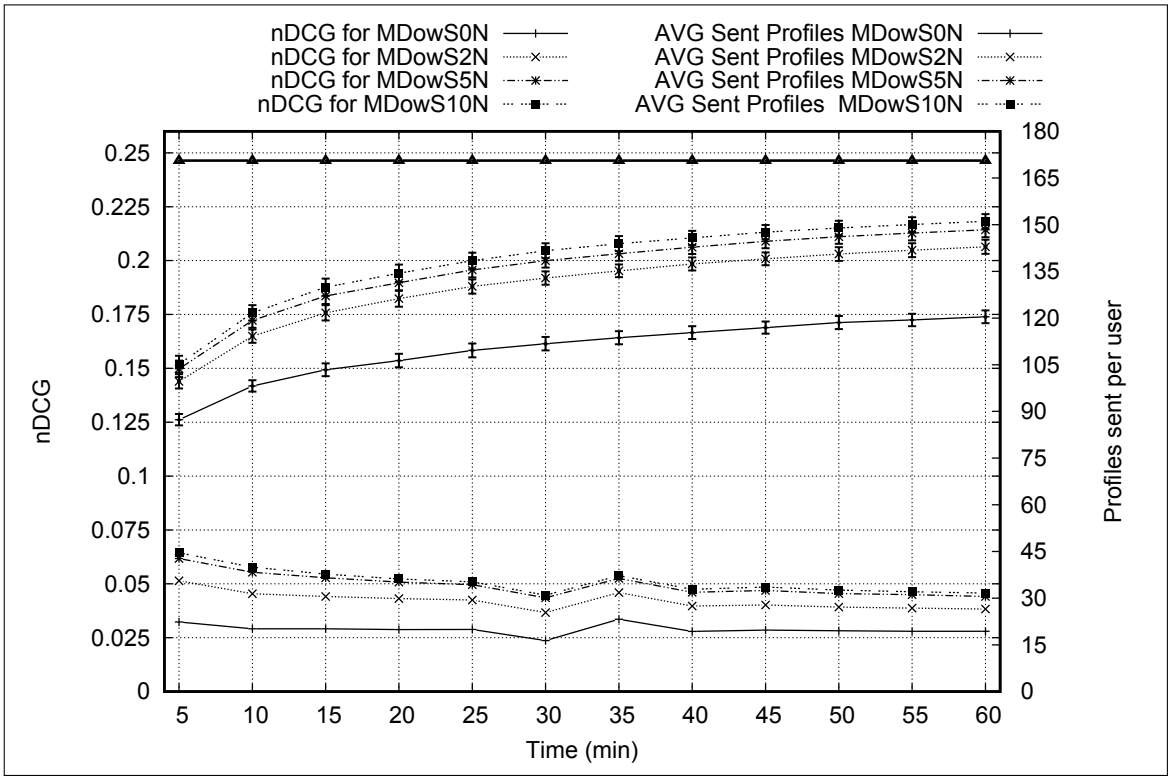
4.5 Assessment of the Distributed Approach

In this section, the results of our experiments are presented. The simulations were executed for a different limit of exchanged profiles, different scenarios, and different storage dispositions. The configurations are named as βN, where β stands for the number of exchanged profiles besides the host profile, i.e., 0N, for example means that only host profile is exchanged on the handshake message. It is important to remark that β is the additional maximum number of exchanged profiles per encounter, since only new profiles will be transferred. When a profile is already in the recipient buffer, it will not be requested again. The only profile always sent is the host.

4.5.1 Primary Buffer

The first experiments were carried out employing the MovieLens dataset. Figure 4.1 presents the *nDCG* obtained in one hour simulation for the MovieLens Downtown Sparse (*MDownS*) scenario, for different number of maximum exchanged profiles per encounter (0N, 2N, 5N, 10N). Moreover, the optimal results obtained when nodes have global knowledge is also displayed. It is possible to notice that, for all number of exchanged profiles, the *nDCG* rises when time elapses. As expected, the less exchanged profiles the lower *nDCG* improvement. In the beginning of the simulation, it is possible to notice a fast development of the recommendations since every encounter brings a lot of new profiles to the nodes. As time elapses, the encounters bring less novelty, therefore *nDCG* improvement is slowed down. It is also possible to notice that the number of maximum exchanged profiles (β) have a large impact in the results when comparing the experiment where nodes just exchanging their own profile (0N) to the test where their own profile plus other two (2N) were transferred. From this point, increasing the number of exchanged profiles (and network traffic) brought smaller improvements. In spite of that fact, a trade-off between the number of maximum profiles exchanged (β) and the convergence speed could be observed. Another important remark is that, after simulating one hour of encounters, the *nDCG* reached levels not so far from the optimal point when β is equal or greater than 3 (2N). This fact demonstrate that distributed recommenders may be employed in certain real applications.

Figure 4.1 – Results for MovieLens Downtown Sparce Scenario. The plot presents the *nDCG* for the `MDowS` with $n = 0$, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.

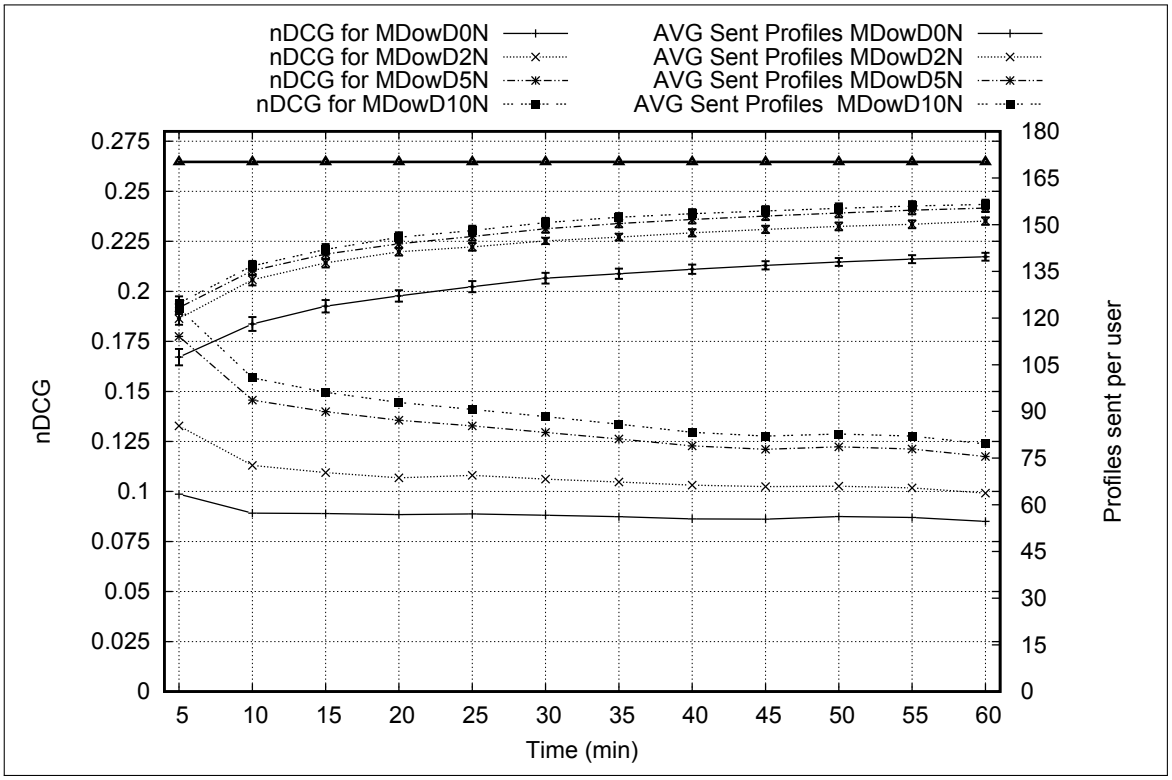


Source: Author

The same figure also presents the effective number of sent profiles per node. This reflects the network traffic. For the 0N experiment, the number of sent profiles is equivalent to the number of encounters, since it always sends its own profile. For the other β, in addition to its own profile, a node may sent from 0 to β additional profiles, depending on the existence of these profiles in the recipient buffer. Thus, at the beginning of the simulation, a sightly larger number of profiles is exchanged for $\beta \geq 2$ when compared to the end of the simulation. It is also possible to observe that for 5N and 10N cases, the number of sent profiles is similar due to the high probability of encountering the same users several times, since it is a sparse scenario the probability of encountering new information is relatively low. This means 5N is enough to incorporate new profiles.

In a scenario with slower movements, as shown in Figure 4.3, we have a behavior similar to the already described. Nonetheless, the *nDCG* stars at a much lower level. This can be explained by the reduced speed of nodes in this scenario. In the previous one, it could be noticed that fast nodes distribute valuable information through the network as the system starts to run. Here, the information dissemination has a reduced speed. This low starting point is followed

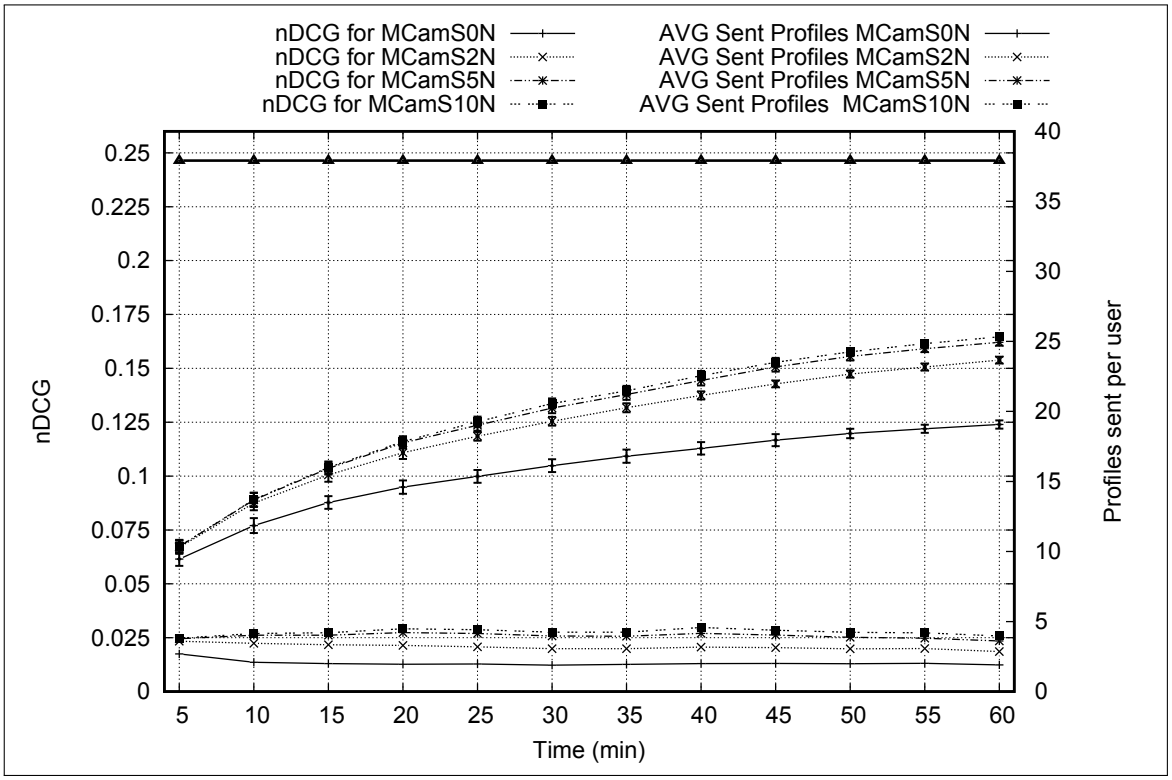
Figure 4.2 – Results for MovieLens Downtown Dense Scenario. The plot presents the *nDCG* for the MDowD with $n = 0$, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.



Source: Author

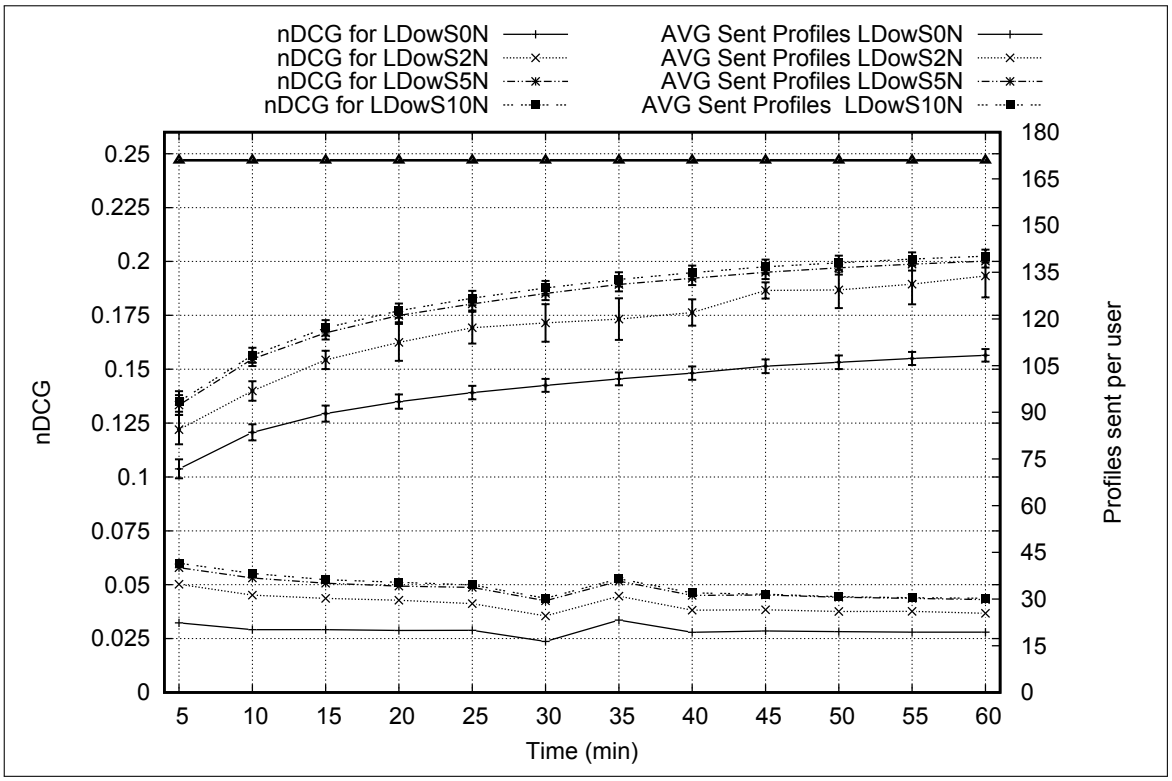
by a rapid improvement for scenarios with $\beta \geq 2$. Similarly to the previous experiment, when each node just exchanged its own profile, the improvements were very slow. The final results, after one hour, were far behind the downtown sparse scenario. This can be explained by the slow movement of the nodes, which brings less encounters per time unit. As expected, the movements of the nodes play a very important role in the convergence of our system. Similarly, the profiles sent per node are also affected by the speed of the nodes. The number of encounters per time unit is impacted, resulting in a lower network traffic.

Figure 4.3 – Results for MovieLens Campus Sparce Scenario. The plot presents the *nDCG* for the `MCamS` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.



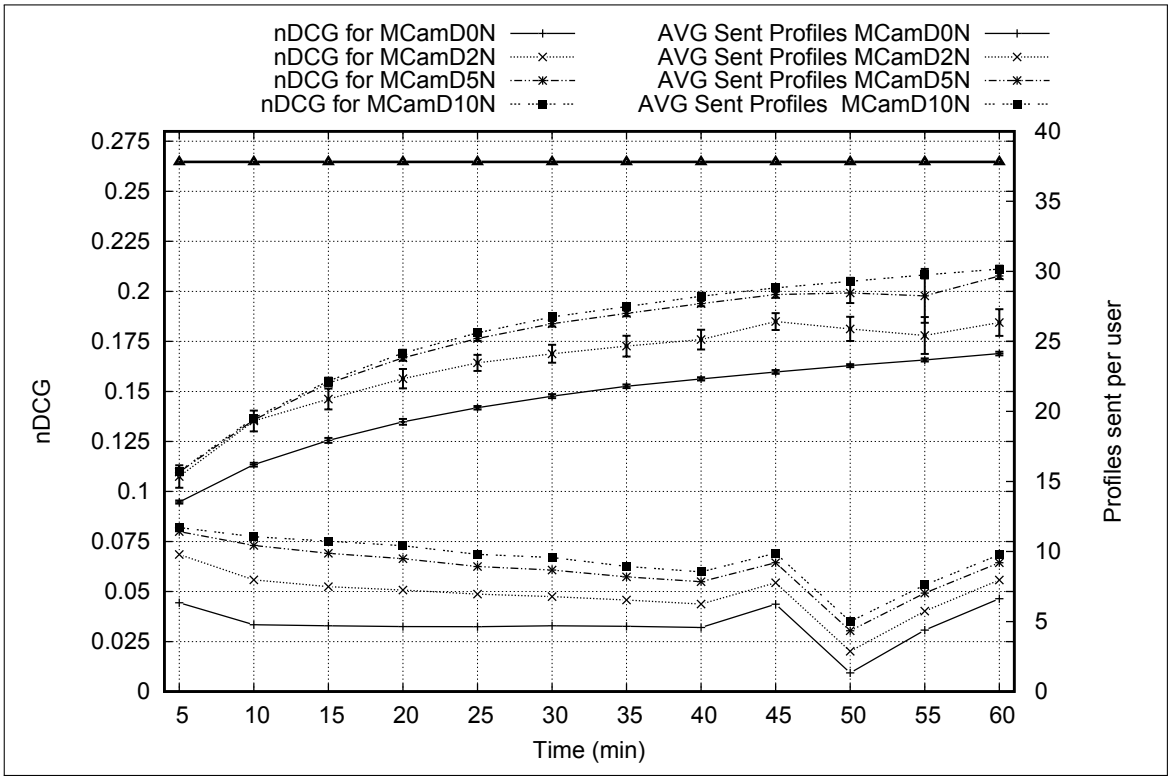
Source: Author

Figure 4.6 – Results for LastFM Downtown Sparse Scenario. The plot presents the *nDCG* for the `LDowS` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.



Source: Author

Figure 4.4 – Results for MovieLens Campus Dense Scenario. The plot presents the *nDCG* for the `MCamD` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.

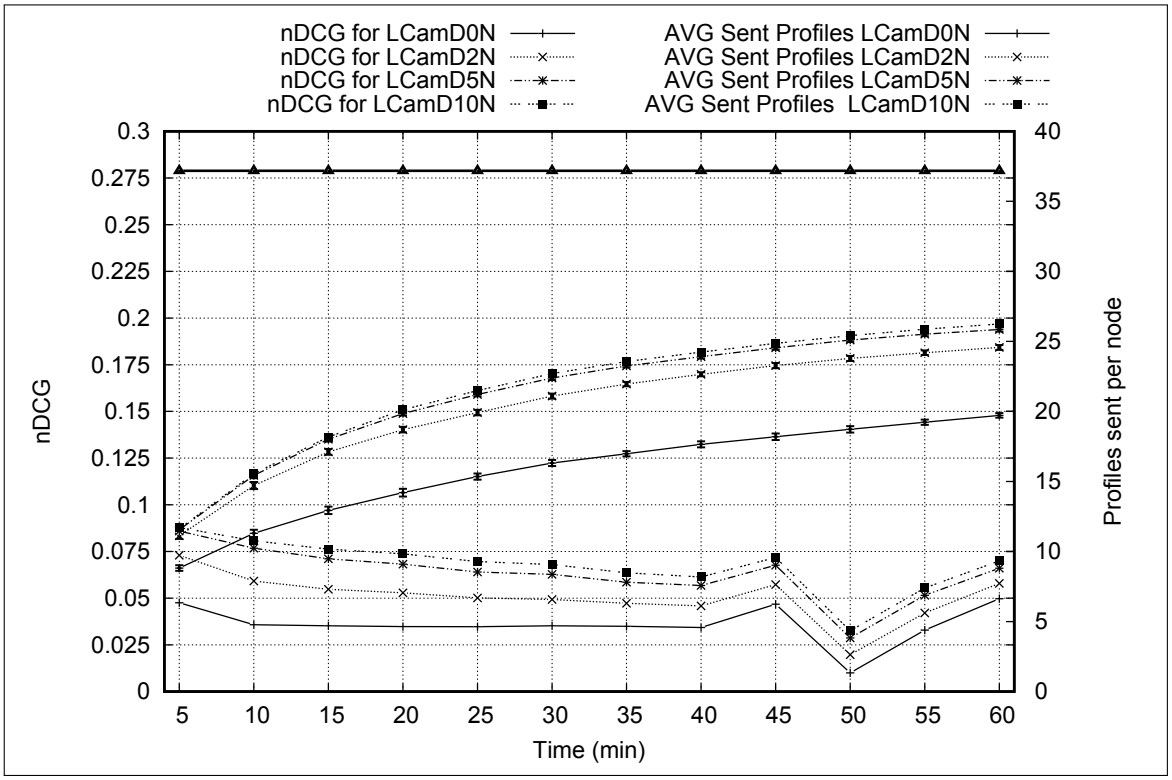


Source: Author

Figure 4.2 presents the *nDCG* for the MovieLens Downtown Dense Scenario (*MDowD*). The results are similar to the *MDowS* scenario, with the following difference: *nDCG* increases faster in the first minutes of the system. This can be explained by the greater amount of encounters at the beginning of the experiment. The convergence to the optimal line is also faster with higher density. When just its own profile is exchanged (0N), the results are considerably lower than the other configurations (2N, 5N, 10N). The network traffic in this experiment was, in average, much higher than the Downtown Sparse scenario. When compared with the Campus Scenarios, the difference is even higher. In this experiments, the β also influenced the number of sent messages, since with high density, novel information is very frequent among nodes.

Figure 4.4 depicts the results for the MovieLens Campus Dense scenario (*MCamD*). This scenario has a similar behavior to the *MCamS* (Campus Sparse Scenario). Nevertheless, it is possible to notice an increased network traffic due to the higher density and a faster nDCG convergence. This reinforces the relation between the number of exchanged profiles and the quality of the recommendations. It is possible to notice than, after running one hour, the results are much closer to the optimal line when compared with the sparse scenario. As general con-

Figure 4.5 – Results for LastFM Campus Dense Scenario. The plot presents the *nDCG* for the `LCamD` with n = 0, 2, 5, and 10. Moreover, the number of exchanged profiles per five minutes is also depicted.



Source: Author

clusion, for the MovieLens scenarios presented here, if a fast convergence to good recommendations is an application requirement, a considerable number of nodes with movement should be present in the environment.

The second set of experiments were carried out employing the LastFM dataset. When observing this dataset in comparison with the MovieLens, it is possible to notice a much greater diversity of users. For example, one user may have consumed a lager number of repeated items and the disparity among users is much greater then the previous dataset. These facts were mirrored in the simulation results. We performed experiments using the same scenarios (Downtown Sparse, Downtown Dense, Campus Sparse and Campus Dense scenarios). In general, each LastFm scenario presented result similar to the correspondent MovieLens scenario. Nonetheless, the nDCG was always behind the MoviLens matching scenario. This can be explained by the higher disparity of the LastFM dataset. To exemplify this fact, Figures 4.5 and 4.6 illustrates the results for the Campus Dense and Downtown Sparse scenarios for the LastFM dataset.

4.5.2 Secondary Buffer

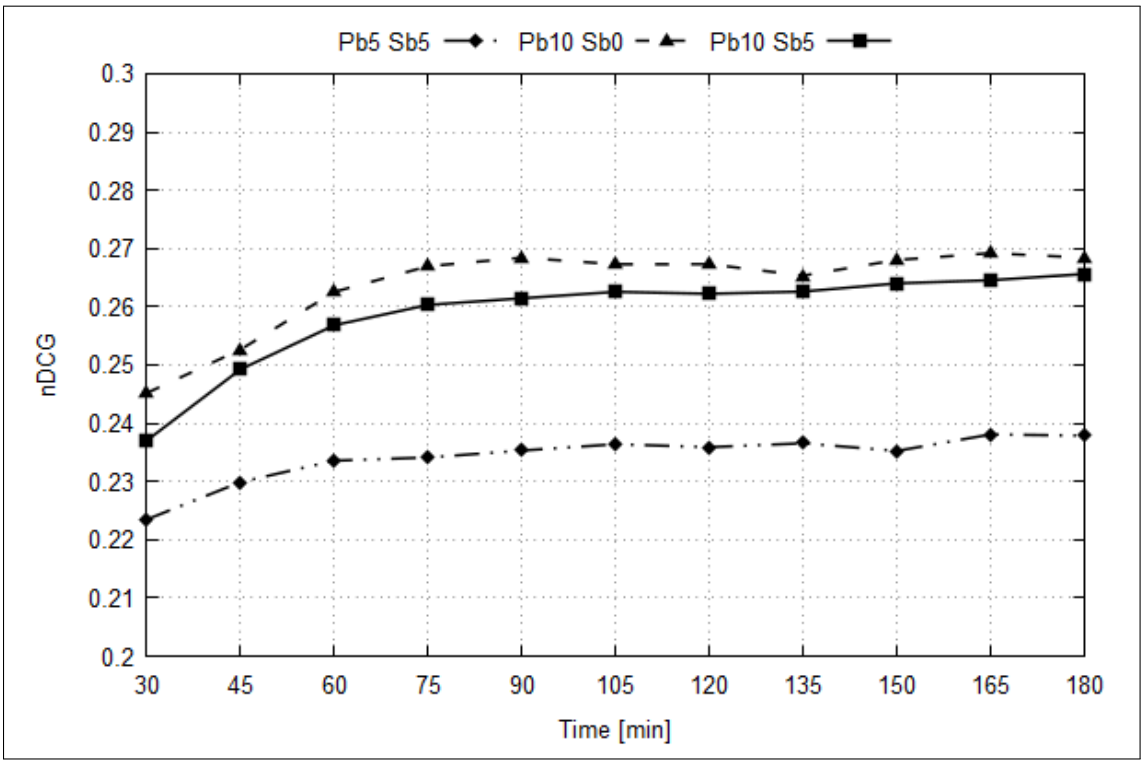
The following experiments employed a realistic scenario and MovieLens dataset. Simulating five different cities on the region of Universidade Federal de Lavras, as presented on Section 4.2. The following results employed the complete dataset, distributing users randomly throughout the map. The initial results demonstrate the average result across the network. It established that increasing the number of profiles on the secondary buffer decreases the average result. The fact can be explained since each user will store dissimilar profiles in order to support grey sheep users.

A derivative experiment was performed to determine if employing the profile information from primary and secondary buffer on the recommendation would have a better outcome when compared with a scenario with only information from the primary buffer. The result demonstrated that recommendation considering only the primary buffer would have a better results, since the remaining profiles will only generate noises on the recommendations. Moreover, the secondary buffer technique would increase general performance on the network, since it would have tradable information on each encounter. Specially when considering a niche of users.

On Figure 4.7 we demonstrate how ineffective the secondary buffer approach is when considering the entire dataset. Furthermore, we present a series of experiment considering a selected portion of users. We have employed the K-means technique with the Euclidean distance in order to clusterize the MovieLens dataset. Then, selected the cluster which presented considerable improvement on outlier users. Figures 4.8, 4.9, and 4.10 displays the impact on the recommendations for grey sheep users. The obvious drawback is a decreased performance for users with popular taste.

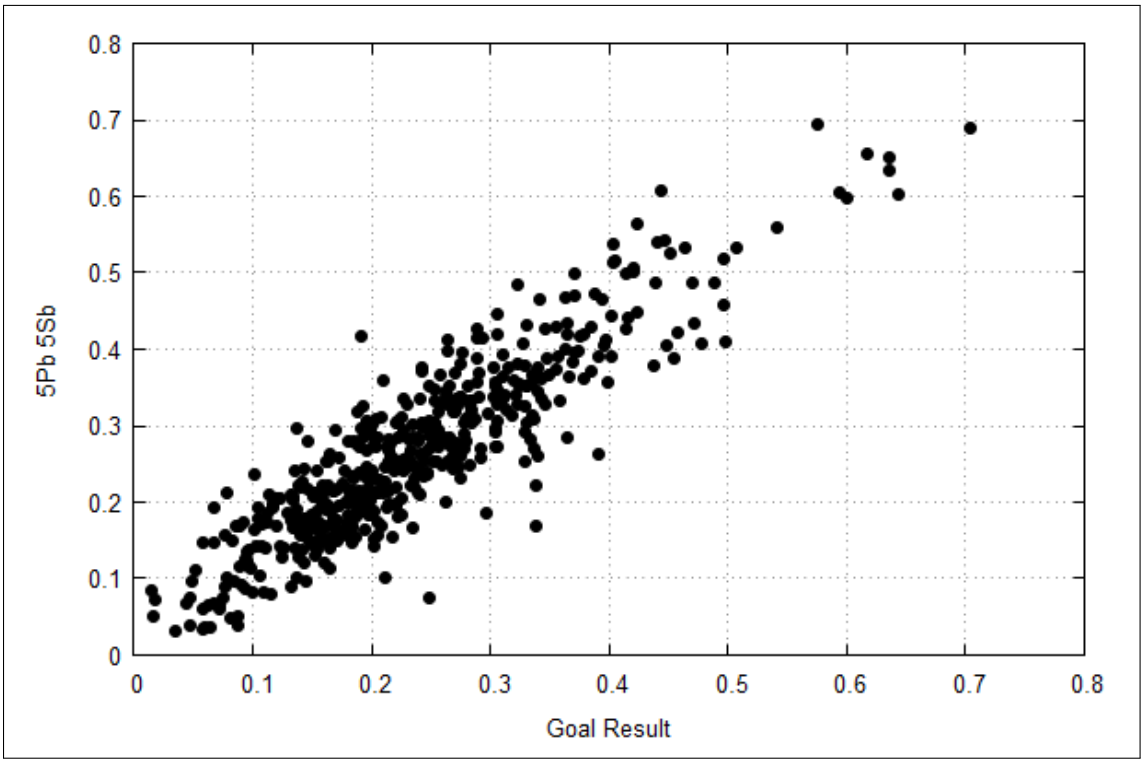
The impact of the secondary buffer on grey sheep users is notable. However, the drawback of helping those in need are too high. Since we employed constant buffers, we have impacted the entire environment. An alternative for this flaw is a dynamic buffer. Users which have stable recommendations would contribute by donating part of their buffer to collect critical data for the network. Those users could be selected by the number of encounter and the relevance of profiles collected.

Figure 4.7 – Secondary buffer result across five different cities, considering three hours of constant movement.



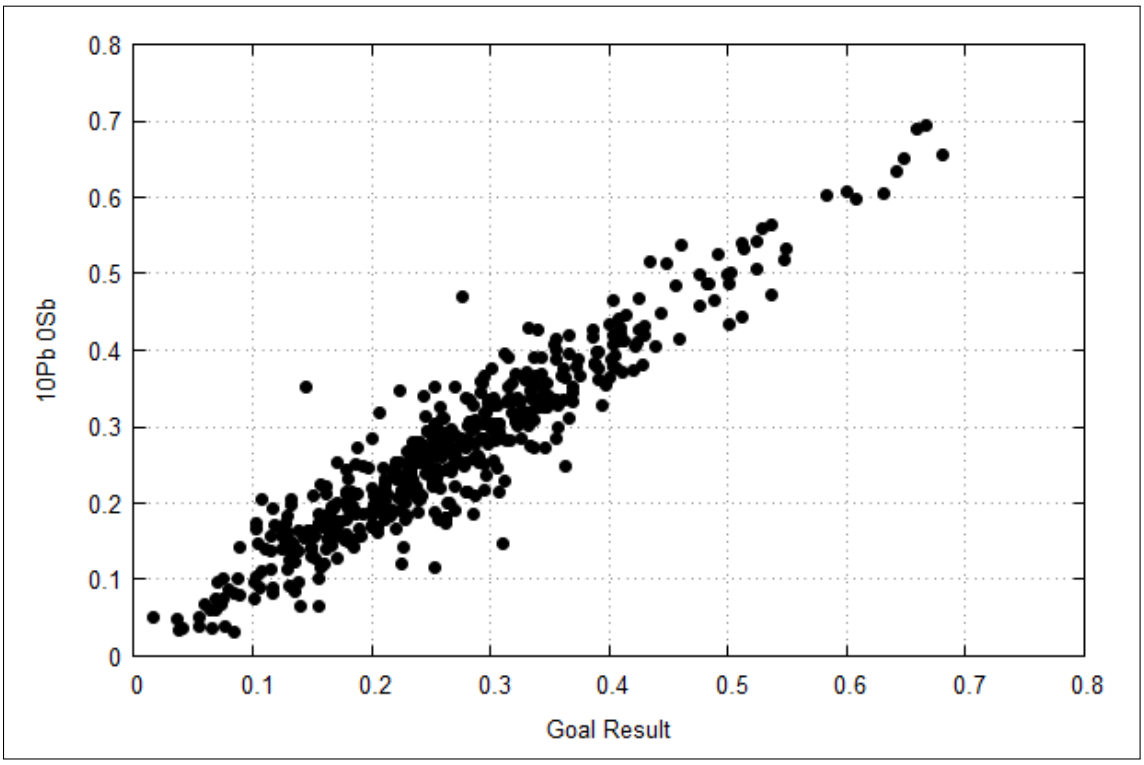
Source: Author

Figure 4.8 – Scatter plot for the technique employing 5 profiles for the primary buffer and 5 profiles for the secondary buffer.



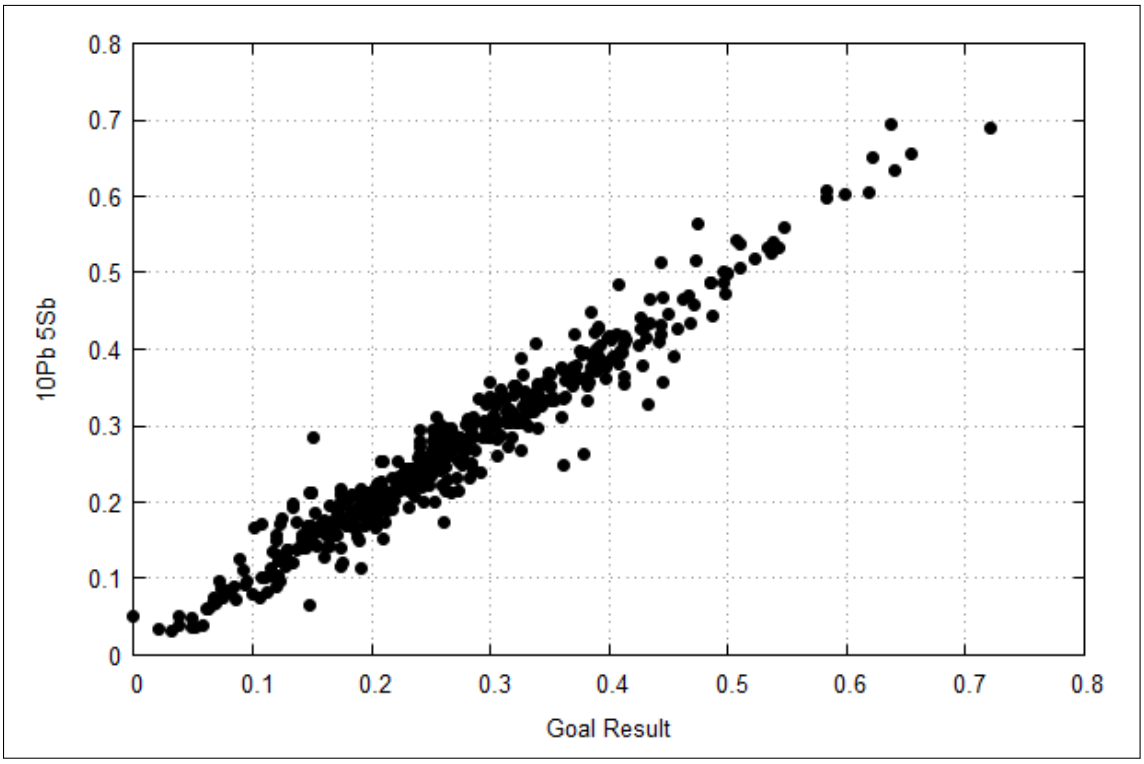
Source: Author

Figure 4.9 – Scatter plot for the technique employing 10 profiles for the primary buffer and no profiles for the secondary buffer.



Source: Author

Figure 4.10 – Scatter plot for the technique employing 10 profiles for the primary buffer and 5 profiles for the secondary buffer.



Source: Author

5 CONCLUSIONS AND FUTURE WORK

On the current state of the project, we proposed an opportunistic distributed recommender system employing the user-based collaborative filtering technique. The framework collects, processes, shares, ranks and computes the recommendation for the users in a collaborative distributed manner. User profiles are exchanged when an opportunity is presented, i.e., two nodes enter into direct communication range.

In order to produce recommendations, the following steps are performed. The user representation component gathers users activities on entertainment applications and then performs the data preprocessing in order to transform the information to an appropriate model. The communication component will opportunistically establish a connection between users and exchange relevant preprocessed profiles. Each node has a profile buffer and the ranking component will discard the least relevant profiles when the application buffer is full or when the profile is outdated. Then, the recommender is responsible for computing the recommendation for each user.

A conventional central server approach was used to select the optimal parameters for the distributed approach and to be used as a baseline for comparison. Simulations were performed using The One Simulator, in order to assess the quality of the recommendations of our distributed system compared to a centralized recommender which contains the knowledge of all users' data. Three scenarios were designed to simulate real-world environments, a campus, a suburb and a downtown.

Furthermore, we studied different scenarios for an opportunistic distributed recommender system employing user-based collaborative filtering technique. Initially, the system collects information from user activities and builds an user model. A communication system is responsible to detect nearby users and, after a handshake, exchange relevant profiles which will populate the buffer of our recommendation system. To decide which profiles are relevant, each node calculates the similarity between the host and the neighbor. If this profile has a higher value than a threshold, the TopN profiles of the neighbor will be received.

The recommendations of each node are computed based on the profiles stored on each local buffer. In order to assess the performance of our system, we selected different scenarios combined with two different node densities and run simulations employing The One opportunistic network simulator. Two different datasets were applied in our assessment. The optimal result for each scenario was obtained by given each node the global knowledge of the network.

Based on these results, we compared the performance of our system during the run time of one hour. For all the scenarios, the nDCG was improving over time, since the nodes acquire more relevant information with the increasing number of encounters. The speed was a relevant factor in the improvement of the quality of recommendations: scenarios with nodes with higher speeds tend to have faster improvements. The density also played a important role: a higher exchange of information is noticed in scenarios with high density. Nevertheless, the communication costs are much higher in high density, high speed scenarios. Another important factor was the structure of the dataset: the LastFM dataset has larger disparity among users when compared with the MovieLens. This leads to a difficulty to recommend quality items, as more time was needed to achieve good recommendations. In general, for scenarios with an adequate amount of movement and considerably density of nodes, after one hour, it was possible to have good recommendations, not so far from the optimal ones. The presented results suggest that a distributed recommendation system can be used in real applications assuming that a minimum density of moving users are present in the environment. Secondary buffer to carry on messages for specific necessities on the system. As the users which consumed most popular items would already have their most relevant neighbors, however, users which have consumed unusual items would probably need more information to have a good recommendations. Therefore, we would implement several strategies to fix the gray sheep problem.

The improvements on the presented components and features that still needs to be included are presented below. The current communication module employs fixed settings regarding transmission range and speed. However, this could be a reliably on battery consumption. Furthermore, an algorithm to adapt those settings in relation to the information gathered from the network. Increasing those variables on a moment of need and decreasing otherwise. Regarding the recommendation module, different techniques can be explored as the context-aware, content-based, and even hybrid approach. Therefore, testing different approach for the recommendations combined with an optimal selection of information from the network would result in a decentralized recommender system with low battery consumption, low cost, and great recommendations.

REFERÊNCIAS

ABELLO, J.; PARDALOS, P. M.; RESENDE, M. G. **Handbook of massive data sets**. New York, NY: Springer, 2013. v. 4.

ADOMAVICIUS, G.; TUZHILIN, A. Context-aware recommender systems. In: **Recommender systems handbook**. Boston, MA: Springer, 2015. p. 191–226.

ALLICENCE, W.-F. **How far does a Wi-Fi Direct connection travel?** 2018. Accessed: 2018-08-22. Disponível em: <https://www.wi-fi.org/knowledge-center/faq/how-far-does-a-wi-fi-direct-connection-travel>.

ALLICENCE, W.-F. **How fast is Wi-Fi Direct?** 2018. Accessed: 2018-08-22. Disponível em: <https://www.wi-fi.org/knowledge-center/faq/how-fast-is-wi-fi-direct>.

ANASTASIADES, C.; BRAUN, T.; SIRIS, V. A. Information-centric networking in mobile and opportunistic networks. In: **Wireless Networking for Moving Objects**. New York, NY: Springer, 2014.

ARNABOLDI, V. et al. A personalized recommender system for pervasive social networks. **Pervasive and Mobile Computing**, v. 36, p. 3–24, 2017.

ARNABOLDI, V. et al. Droidopppathfinder: A context and social-aware path recommender system based on opportunistic sensing. **2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)**, p. 1–3, 2013.

ASADI, A.; WANG, Q.; MANCUSO, V. A survey on device-to-device communication in cellular networks. **IEEE Communications Surveys & Tutorials**, IEEE, v. 16, n. 4, p. 1801–1819, 2014.

BALABANOVIĆ, M.; SHOHAM, Y. Fab: content-based, collaborative recommendation. **Communications of the ACM**, ACM, v. 40, n. 3, p. 66–72, 1997.

BARBOSA, L. N. et al. Distributed user-based collaborative filtering on an opportunistic network. In: IEEE. **2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)**. Chicago, IL, 2018. p. 266–273.

BENNETT, J.; LANNING, S. et al. The netflix prize. In: **Proceedings of KDD cup and workshop**. New York, NY: Netflix, 2007. v. 2007, p. 35.

BOBADILLA, J. et al. Recommender systems survey. **Knowl.-Based Syst.**, v. 46, p. 109–132, 2013.

BOSTANDJIEV, S.; O'DONOVAN, J.; HÖLLERER, T. Tasteweights: a visual interactive hybrid recommender system. In: ACM. **Proceedings of the sixth ACM conference on Recommender systems**. Santa Barbara, CA, 2012. p. 35–42.

BOUTET, A. et al. Whatsup: A decentralized instant news recommender. In: IEEE. **Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on**. Cambridge, MA, 2013. p. 741–752.

BREESE, J. S.; HECKERMAN, D.; KADIE, C. M. Empirical analysis of predictive algorithms for collaborative filtering. In: COOPER, G. F.; MORAL, S. (Ed.). **Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence**. Redmond, WA: Citeseer, 1998. p. 43–52.

BURKE, R. Hybrid recommender systems: Survey and experiments. **User modeling and user-adapted interaction**, Springer, v. 12, n. 4, p. 331–370, 2002.

BURKE, R. Hybrid web recommender systems. In: **The adaptive web**. Chicago, IL: Springer, 2007. p. 377–408.

CELMA, O. **Music Recommendation and Discovery in the Long Tail**. Barcelona, Spain: Springer, 2010.

CONTI, M. et al. From opportunistic networks to opportunistic computing. **IEEE Communications Magazine**, v. 48, 2010.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107–113, 2008.

DESROSIERS, C.; KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods. In: **Recommender systems handbook**. Boston, MA: Springer, 2011. p. 107–144.

DHURANDHER, S. K. et al. Probability-based controlled flooding in opportunistic networks. **2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)**, v. 06, p. 3–8, 2015.

DIDELES, M. Bluetooth: a technical overview. **XRDS: Crossroads, The ACM Magazine for Students**, ACM, v. 9, n. 4, p. 11–18, 2003.

EKSTRAND, M. D. et al. Collaborative filtering recommender systems. **Foundations and Trends® in Human–Computer Interaction**, Now Publishers, Inc., v. 4, n. 2, p. 81–173, 2011.

FALL, K. R. A delay-tolerant network architecture for challenged internets. In: **SIGCOMM**. New York, NY: ACM, 2003.

FELFERNIG, A. et al. Developing constraint-based recommenders. In: **Recommender systems handbook**. Berlin, Heidelberg: Springer, 2011. p. 187–215.

FELFERNIG, A. et al. Basic approaches in recommendation systems. In: **Recommendation Systems in Software Engineering**. Berlin, Heidelberg: Springer, 2014. p. 15–37.

FOURSQUARE. **Foursquare Developer**. 2018. Accessed: 2018-08-21. Disponível em: <https://developer.foursquare.com/>.

GANCHEV, I.; JI, Z.; O'DROMA, M. A distributed cloud-based service recommendation system. **2015 International Conference on Computing and Network Communications (CoCoNet)**, p. 212–215, 2015.

GOLDBERG, D. et al. Using collaborative filtering to weave an information tapestry. **Communications of the ACM**, ACM, v. 35, n. 12, p. 61–70, 1992.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. **Acm transactions on interactive intelligent systems (tiis)**, ACM, v. 5, n. 4, p. 19, 2016.

HUANG, Z.; CHUNG, W.; CHEN, H. A graph model for e-commerce recommender systems. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 55, n. 3, p. 259–274, 2004.

IMDB. **IMDb Datasets**. 2018. Accessed: 2018-08-21. Disponível em: <https://www.imdb.com/interfaces/>.

IPPISCH, A.; KÜPER, T.; GRAFFI, K. Time and space in android-based opportunistic networks. In: IEEE. **2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)**. Krakow, Poland, 2018. p. 244–250.

JAIN, S.; FALL, K. R.; PATRA, R. K. Routing in a delay tolerant network. In: **SIGCOMM**. New York, NY: ACM, 2004.

JÄRVELIN, K.; KEKÄLÄINEN, J. Cumulated gain-based evaluation of ir techniques. **ACM Transactions on Information Systems (TOIS)**, ACM, v. 20, n. 4, p. 422–446, 2002.

JEDARI, B.; XIA, F. A survey on routing and data dissemination in opportunistic mobile social networks. **arXiv preprint arXiv:1311.0347**, 2013.

KERäNEN, A.; OTT, J.; KäRKKäINEN, T. The ONE Simulator for DTN Protocol Evaluation. In: **SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques**. New York, NY, USA: ICST, 2009. ISBN 978-963-9799-45-5.

KERMARREC, A.-M. et al. Application of random walks to decentralized recommender systems. **Principles of Distributed Systems**, Springer, p. 48–63, 2010.

LASTFM. **Last.FM**. 2018. Accessed: 05-31-2017. Disponível em: <https://www.last.fm/>.

LEI, L. et al. Operator controlled device-to-device communications in lte-advanced networks. **IEEE Wireless Communications**, IEEE, v. 19, n. 3, 2012.

LI, S.; XU, L. D.; ZHAO, S. 5g internet of things: A survey. **Journal of Industrial Information Integration**, Elsevier, University of the West of England, UK, 2018.

LIN, Y.-D.; HSU, Y.-C. Multihop cellular: A new architecture for wireless communications. In: IEEE. **INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE**. Tel Aviv, Israel, 2000. v. 3, p. 1273–1282.

MARTÍN-CAMPILLO, A. et al. Evaluating opportunistic networks in disaster scenarios. **J. Network and Computer Applications**, v. 36, p. 870–880, 2013.

MAVROMOUSTAKIS, C. X.; MASTORAKIS, G.; BATALLA, J. M. **Internet of Things (IoT) in 5G mobile technologies**. New York, NY: Springer, 2016. v. 8.

NIU, J. et al. Fuir: Fusing user and item information to deal with data sparsity by using side information in recommendation systems. **Journal of Network and Computer Applications**, Elsevier, v. 70, p. 41–50, 2016.

OpenStreetMap contributors. **Planet dump retrieved from https://planet.osm.org** . 2017. <https://www.openstreetmap.org>.

OWEN, S.; OWEN, S. Mahout in action. Manning Shelter Island, 2012.

PAZZANI, M. J.; BILLSUS, D. Content-based recommendation systems. In: **The adaptive web**. Palo Alto, CA: Springer, 2007. p. 325–341.

PELUSI, L.; PASSARELLA, A.; CONTI, M. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. **IEEE Communications Magazine**, v. 44, 2006.

POUSHTER, J. et al. Smartphone ownership and internet usage continues to climb in emerging economies. **Pew Research Center**, Smartphone ownership and internet usage continues to climb in emerging economies, v. 22, p. 1–44, 2016.

RAHIMI, H.; ZIBAEENEJAD, A.; SAFAVI, A. A. A novel iot architecture based on 5g-iot and next generation technologies. **arXiv preprint arXiv:1807.03065**, 2018.

RESNICK, P. et al. Grouplens: an open architecture for collaborative filtering of netnews. In: ACM. **Proceedings of the 1994 ACM conference on Computer supported cooperative work**. New York, NY, 1994. p. 175–186.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: **Recommender systems handbook**. New York, NY: Springer, 2011. p. 1–35.

RICCI, F. et al. **Recommender systems handbook**. New York, NY: Springer, 2015.

RIEDL, J.; KONSTAN, J. **Movielens dataset**. 1998. Accessed: 05-31-2017. Disponível em: <https://grouplens.org/datasets/movielens/>.

SCHAFER, J. B. et al. Collaborative filtering recommender systems. In: **The adaptive web**. New York, NY: Springer, 2007. p. 291–324.

TAO, Y. Design of large scale mobile advertising recommendation system. **2015 4th International Conference on Computer Science and Network Technology (ICCSNT)**, p. 763–767, 2015.

TREWIN, S. Knowledge-based recommender systems. **Encyclopedia of library and information science**, v. 69, n. Supplement 32, p. 180, 2000.

UNGER, M. et al. Towards latent context-aware recommendation systems. **Knowledge-Based Systems**, Elsevier, v. 104, p. 165–178, 2016.

VERMA, J. P.; PATEL, B.; PATEL, A. Big data analysis: Recommendation system with hadoop framework. **Computational Intelligence & Communication Technology**, p. 92–97, 2015.

VERMA, S. K.; MITTAL, N.; AGARWAL, B. Hybrid recommender system based on fuzzy clustering and collaborative filtering. In: **2013 4th International Conference on Computer and Communication Technology (ICCCT)**. Allahabad, India: IEEE, 2013.

WANG, Z. et al. Decentralized recommender systems. **arXiv preprint arXiv:1503.01647**, 2015.

WHITE, T. **Hadoop: The definitive guide**. Sebastopol, CA: "O'Reilly Media, Inc.", 2012.

YANG, W.-S.; HWANG, S.-Y. itravel: A recommender system in mobile peer-to-peer environment. **Journal of Systems and Software**, Elsevier, v. 86, n. 1, p. 12–20, 2013.

YIN, H. et al. Lcars: a location-content-aware recommender system. In: ACM. **Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining**. New York, NY, 2013. p. 221–229.

ZANOTTI, G. et al. Infusing collaborative recommenders with distributed representations. In: ACM. **Proceedings of the 1st Workshop on Deep Learning for Recommender Systems**. Chicago, IL, 2016. p. 35–42.

ZHAO, Y.; YE, J.; HENDERSON, T. A robust reputation-based location-privacy recommender system using opportunistic networks. In: . St. Andrews, UK: University of St Andrews Research, 2016.

ZIEGLER, C.-N. et al. Improving recommendation lists through topic diversification. In: ACM. **Proceedings of the 14th international conference on World Wide Web**. New York, NY, 2005. p. 22–32.