FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Nutrially – Efficient and personalised meal recommendations

Ana Margarida Amaro

Mestrado Integrado em Bioengenharia

Supervisor: Carlos Soares | FEUP

Co-supervisor: David Ribeiro | Fraunhofer Portugal

July 29, 2021

Nutrially – Efficient and personalised meal recommendations

Ana Margarida Amaro

Mestrado Integrado em Bioengenharia

July 29, 2021

Resumo

A preocupação em manter uma alimentação saudável tem vindo a crescer nos últimos anos. Porém, o aumento geral no ritmo pessoal de trabalho tem impossibilitado a adaptação dos hábitos alimentares de uma grande percentagem da população. O tempo para preparação das refeições e de ida às compras, assim como a facilidade e baixo preço de opções de comidas rápidas e pouco saudáveis, dificultam a mudança destes hábitos.

Tem existido, assim, um interesse crescente no desenvolvimento de técnicas e tecnologias que facilitem esta mudança e incentivem a sua continuação. Aplicações nutricionais são uma das maiores áreas de interesse.

Recentemente, o interesse na adaptação de sistemas de recomendação ao domínio da alimentação tem aumentado. Sistemas de recomendação são comumente usados em lojas online e sites de catálogos de livros e filmes, por exemplo. Os mais estudados estão relacionados com recomendações individuais porém o estudo de sistemas de recomendação de conjuntos de itens ou de recomendações de grupo tem vindo a aumentar.

Para o problema corrente, o objetivo é o desenvolvimento de um algoritmo de recomendação semanal de refeições considerando várias restrições e as preferências de cada utilizador.

Os passos principais deste projeto são:

- Desenvolvimento de um sistema de recomendação simples;
- Ligação entre *Python* e *Java* para troca de informação;
- Criação de um algoritmo genético para obter o plano semanal.

Assim, foi desenvolvido um algoritmo genético de forma a obter o plano semanal de refeições. Uma base de dados com informações sobre os utilizadores assim como informação relacionada com as receitas foi utilizada para aplicar o algoritmo.

Em paralelo, foi desenvolvido um sistema de recomendação simples. Foram utilizadas as duas técnicas mais comuns: *content-based* e *collaborative filtering*. Uma versão híbrida foi desenvolvida utilizando as duas técnicas.

Relativamente aos sistemas de recomendação, o método que apresentou melhores resultados foi o híbrido enquanto o *content-based* apresentou os piores.

Relativamente ao algoritmo genético, um estudo detalhado das suas variáveis como tamanho da população, probabilidade de *crossover*, entre outras, foi desenvolvido.Depois de encontrar o melhor conjunto de variáveis para o algoritmo , três cenários diferentes foram estudados. Para avaliar os resultados, foi utilizado um conjunto de planos obtidos de forma aleatória e um algoritmo previamente desenvolvido.

Este estudo involveu vários tamanhos de população, assim como modificações na função *fitness*. Os resultados obtidos para o algoritmo genético foram semelhantes aos obtidos pelo algoritmo anterior. O aumento do tamanho da população no algoritmo genético leva a melhores resultados porém com um maior custo computacional. Ambos obtiveram resultados acima dos

melhores resultados obtidos pela procura aleatória. Para um dos cenários, existiu uma diferença substancial entre os dois algoritmos. Neste caso, o algoritmo genético teve melhores resultados. Para este cenário, o algoritmo genético mostrou ser mais robusto a modificações na função *fitness*.

Abstract

There has been a growing worry in keeping a healthy lifestyle for the past few years. However, the stressful working pace has hampered the adoption of healthier eating habits for a huge percentage of the population. The required time for meal preparation and grocery shopping as well as the convenience of fast food options hinder habits changes.

This led to an expansion in the development of techniques and technologies that ease these eating habits changes and encourages their maintenance. Nutritional applications are one of the biggest search areas.

Recently, food recommender systems have received increasing attention due to their relevance for healthy living. Recommender systems have been thoroughly studied and expanded in online stores and catalog sites for books and movies, for example. The most studied systems are related to individual recommendations, however, the study of package and group recommender systems has increased.

For the current problem the goal is the development of an algorithm for a week's worth meal plan considering several constraints and user preferences.

The main steps of this project consist in:

- Simple Recommendation System;
- Python and Java connection for data retrieval;
- Genetic Algorithm to obtain the week plan.

In this study, a genetic algorithm was developed to obtain the week plan. A dataset with the user's information and recipes data was used to apply the framework.

In parallel, a straightforward recommender system was developed. The two most popular frameworks were used - content-based and collaborative filtering. A hybrid version using both was also developed.

For the recommender systems methods, the hybrid model presents the best results while the content-based performed the worst.

To apply the genetic algorithm result, a detailed study of its variables such as population size, crossover rate, etc was made. As for the genetic algorithm, a thoroughly study was performed to choose the best set of conditions and after obtaining the results three different scenarios were tested.To evaluate the results of the algorithm, both a random generation of plans and a previous developed greedy algorithm were used.

Several population sizes were considered and changes in the fitness function were made. Overall the genetic algorithm had similar results to the previously developed algorithm. The increase in the population size led to better results however with higher computational time. Both performed better than the best results obtained by the random search. In one of the scenarios, a significant difference exists between the two algorithms. For this specific case, the genetic algorithm showed better results. For this scenario the genetic algorithm showed more robustness to modifications in the fitness function.

Agradecimentos

A realização deste trabalho não teria sido possível sem a ajuda de várias pessoas. Em primeiro quero agradecer aos meus orientadores: David Ribeiro e Professor Carlos Soares pelo apoio prestado nestes últimos meses. Agradecer também à Fraunhofer pela oportunidade de desenvolver este projeto.

Agradecer também a todas as pessoas, que de uma forma ou outra, me ajudaram durante os últimos 5 anos. Aos meus amigos, quer da faculdade quer aos mais antigos, um obrigada pela companhia e bons momentos. Por último, o maior agradecimento à minha família por ser o meu maior pilar.

Margarida Amaro

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

AUC	Area under Receiver Operating Characteristic Curve
CF	Collaborative Filtering
EA	Evolutionary Algorithm
HRS	Health Recommender Systems
MAE	Mean Absolute Error
MOEA	Multiobjective Evolutionary Algorithms
MOP	Multiobjective Optimization Problem
NMAE	Normalized Mean Average Error
RMSE	Root of Mean Square Error
ROC	Receiver Operating Characteristic
SOP	Scalar Objective Optimization Problem
SVD	Singular Value Decomposition
SD	Standard Deviation

Chapter 1

Introduction

As the ratio of population above 65 years old increases, there is a tendency for a rise in chronic diseases and physical and mental disabilities. Malnutrition occurs due to imbalances in a person's nutritional or energy intake. Malnutrition includes undernutrition, inadequate vitamins or minerals intake, overweight, obesity and resulting diet-related noncommunicable diseases [1]. Its problems are often associated with the elderly population.

The percentage of elderly people has been increasing and life expectancy has seen a continuous rise in the past few years. A growth from 12% to 22% in the proportion of the population over 60 years old is expected between 2015 and 2050 in Europe [2]. This rise in the percentage of the ageing population will ultimately lead to an increase in malnutrition prevalence, as this is an age gap very prone to suffer from malnutrition. The percentage of population suffering from malnutrition is expected to reach 29.1% by 2080 [3].

Several factors affect the growing proportion of population with malnutrition status. Physical health problems, mental health issues, appetite changes, financial autonomy and the socio-environmental context hugely influence food habits [4]. Meal preparation, healthy diet choices and groceries shopping planning difficulties are very common.

Nutritional habits play a key role in preventing diseases, improving quality of life and even receding some health problems.

Although aware of the importance of healthy eating habits, people often neglect suitable behaviors. Many people lead a turbulent lifestyle decreasing the time for meal planning. For these scenarios, fast food presents an easy and comfortable choice. This is where nutrition recommendation systems might play a key role in improving eating habits. They are considered an effective tool in helping its users change their eating behavior and start making healthier food choices [5].

1.1 Motivation

As explained previously, the percentage of the population with malnutrition problems is significant. Malnutrition leads to other health problems that hugely affect the elderly population. The

need for a system to help the user in a transition to a healthier lifestyle is clear. A meal recommender system can influence and help improve eating habits by providing a structured guide to make healthier decisions [6]. Such system should not only help meal preparation, but also inquire users about their drinking habits since water intake presents itself as another issue with the elderly population. Several constraints must be adopted in such scenario. The proposed plan must offer alternatives since food suppliers and supermarket visits might be unpredicted and find unforeseen situations. Therefore, multiple restrictions should be accounted for such as avoiding repeating recent dishes, alternating between fish and meat meals and favouring seasonal products.

For meal planning, user's personal information (food preferences, nutritional needs, allergies) must be contemplated. With the user's information, the next step consists in applying restriction rules – constraints. Another important step is scaling the ingredients quantity to fulfill the nutritional requirements of each individual.

In [7], a nutrition system that provides a weekly meal plan considering the user needs and preferences was developed. This methodology will be thoroughly studied. The main goal of this work is to develop an alternative system that combines a recommender system with an optimization technique to obtain weekly plans adjusted to the user.

Recommender systems have been commonly used for e-commerce, movies and books recommendation. More recently, the nutritional scenario has gained some popularity [8]. Commonly, recommender systems have as an output individual recommendations. However, in this case, several meals must be coupled and some dependency between them must be contemplated in order to fill all the necessities.

Package recommendations systems have recently been studied for several domains [9]. Travel and outfit recommendation are two commonly studied areas. Travel recommendation [10] requires a sequential path, where distance, time and money should be minimized in the created route and must be adapted to the customer's needs. As for outfit recommendation, the connection between items is not sequential but complementary, since when creating an outfit, the top must be chosen considering the already picked bottom or vice versa.

For this work, a strict sequence is not necessary since meals may be switched, Tuesday's lunch may be exchanged with Monday's lunch however dinners must be lighter than lunches and ingredients depend greatly on the grocery shopping days and available products, therefore, a dependent package recommendation must be contemplated.

1.2 Goals

Considering user preferences and the previously mentioned constraints, a weekly meal plan must be recommended. This meal plan must be adapted to the user's needs to guide the user into a healthier lifestyle.

The goal of this work is to apply commonly used techniques for similar problems. To address the meal recommendation problem, an optimization method supported by a recommender system must be developed. Several methods must be tested and adapted to the nutrition scenario

where multiple constraints and dependency between recommendations must be taken into consideration. In order to do so, an off-the-shelf recommendation algorithm followed by an optimization framework to create the weekly plan should be tested. This method will be based on off-the-shelf optimization methods and recommender systems.

A simple recommender system and a genetic algorithm will be evaluated. Several scenarios will considered and the methodology will be tested for different users in order to evaluate its effectiveness.

1.3 Document Structure

The present paper is organized as follows: in chapter 2, an introduction to concepts regarding recommender systems, common optimization techniques is done and a review of multiple frameworks is presented as well as an introduction to the main concepts of genetic algorithms; for chapter 3, the used methods are described in detail; chapter 4 presents the obtained results and its discussion; finally, conclusions and future work are drawn in chapter 5.

Chapter 2

Literature Review

In this chapter, a review of the most important concepts for this project will be presented.

First, a presentation of concepts regarding recommender systems will be done. Common optimization techniques will be briefly explained and genetic algorithms concepts will be addressed. Package recommender systems, that combine recommender systems and optimization methods, will be introduced and a review of multiple frameworks will be presented.

2.1 Recommender systems

Recommendation systems have been commonly used and adapted to the nutrition field. Considered an effective tool, they aid users in eating behaviour changes [6].

In this section, the basic concepts of recommender systems will be addressed. The several types of recommender systems will be briefly explained and some commonly used frameworks will be presented. Finally, a brief description of Health Recommendation Systems (HRS) will be done.

2.1.1 Recommender Systems Concepts

A recommendation system goal is to predict an item which a certain user would prefer, taking into account information about the users and their needs.

Recommendation algorithms can be formulated in one of two ways: the first, consisting in prediction of the rating value of a user-item combination; the second, occurs when the goal is not necessarily a specific rating value of user-item combination but to learn the top-k most relevant items for a particular user [11]. While for the first scenario the idea is to explicitly measure how likely it is for the user to like the recommended item, for example, a book recommendation engine, the second scenario is commonly used for e-commerce websites where, very often, the website shows example of items that other users bought together where the predicted recommendation result is not necessary.

According to [12], three main aspects must be considered for recommender systems: usage context, users and items. Table 2.1 briefly summarizes the main recommender systems concepts.

Usage context includes the contextual factors and multi-factorial goal setting that influences the recommendation. The inclusion of this information will help understanding the context that led to the user's current behavior and preferences. Usage context outlines the environment where items and users interact [13].

Users are the end-users of recommender systems whose personal information must be recorded in order to optimize the results.Finally, items are the elements recommended to users. The items may be movies, tv shows, books or considering the HRS can be diets and medication, for example.

The input given by users for recommendation can be either implicit or explicit. Implicit input is unconsciously appointed by the user (item clicks, time spent at an item page) while explicit input is provided consciously (scale ratings) [9].

The formalization of the recommendation problem must contain, as stated previously, users and items possible for recommendation. A rating matrix R is created to measure each user preference to the items. The preference of a user $u \in U$ to an item $i \in I$ is measured by $r_{u,i}$, a real number within a specified range. The main goal in these systems is the prediction of unknown ratings in r. These ratings will be analysed in order to recommend items to the users based on it. The prediction, $p_{u,i}$, measures the strength between user u and item i.

Recommender systems may consider single or multiple criterion values for the recommendations. Using movie recommendations as an example, multiple criterion recommender systems will be explained. The system analyses the user's overall movie rating. However, this rating is only based on how much the user enjoyed the experience. For a multi criterion recommender system, instead of a single rating, the users rate the movie considering several criteria such as visual effects, story line and actors performance. The increase in the number of criteria can improve the recommendation however it may result in a higher computational cost [14].

To illustrate this system and considering the current problem that is being explored we can separate the concepts. When dealing with nutrition and dietary restrictions, the users, U are the patients whose goal is to lead a healthier lifestyle. The items, I are all the recipes in the database used for the development of the solution. So, in order to create a recommender system the patients must rate (on a specific scale, might be from 0-5, for example) some recipes, r_u and using that information the system will suggest new possibilities the patient might like, $p_{u,i}$. If instead of relying only on ratings, new information such as dietary restrictions, favorite cuisine amongst others is considered, the recommendation system will create better suggestions however, the overall time to obtaining them will increase.

The evaluation of recommender systems must be done for three different scenarios: simple recommendations, recommendation sets and ranked recommendation lists [16].

2.1.1.1 Recommender systems methods

The purpose of Recommender systems is to make suggestions based on user preferences. The methods used can be separated into two major categories: collaborative filtering (CF) methods and content based methods. Recently, both approaches have been combined – hybrid frameworks.

Table 2.1: Recommender systems notation [15].

Variables	Description	Example
U	set of users	patient
I	set of items	recipes
I_u	set of items rated by user U	set of recipes a patient rated
U_i	set of users that rated item i	all the ratings for a specific recipe
R	rating matrix	-
r_u	vector of all user U' ratings	-
r_i	vector of all users rating for item i	-
$p_{u,i}$	predicted recommendation	recipe predicted recommendation

While content based techniques are based on similarity between the attributes of items, collaborative approaches are based on the similarity between interactions. The algorithm's task is to learn a function that predicts the utility of items to each user. Content based techniques compare the item's profile. In food recommendation systems, it compares the nutritional informational and ingredients, for example, of recipes. Therefore, the recipe's information is studied and each recipe described according to the study. Through the analysis of the user's rated recipes, the most similar recipes content wise (ingredients, nutritional values) will be recommended.

As for CF the behaviour is analysed. Therefore, for food recommendations the previous meals and patient history are checked. Each user will have its profile built according to their rated recipes. The user's profiles will be compared and the most similar will be considered. Assuming that users with similar liked recipes will have similar tastes, a highly rated recipe from the similar users set will be recommended.

In figure 2.1b, the difference between the two methods is illustrated using the nutritional problem as an example.

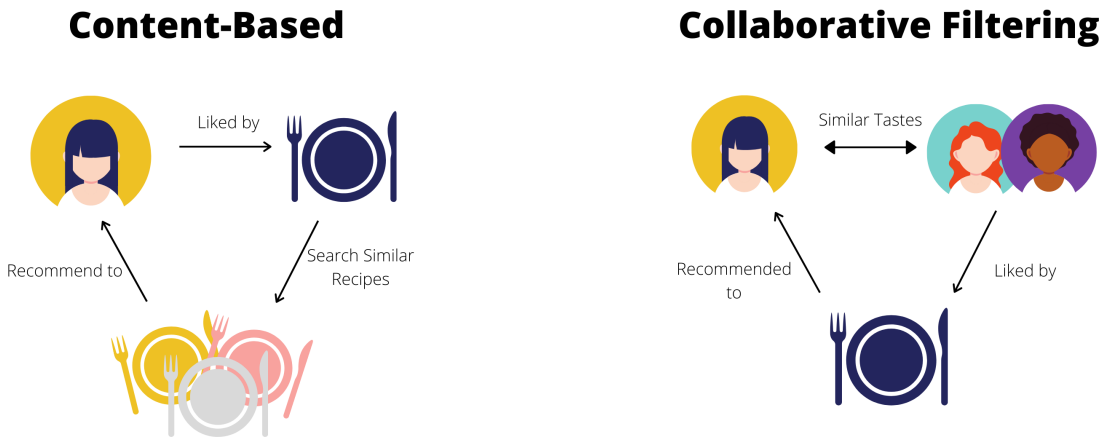

Figure 2.1: Content-based versus collaborative filtering recommendation algorithms.

To summarise, in content based scenarios, the search will consist in gathering the user rating

recipes and using each recipe information such as ingredients, difficulty and time will be used to find the most similar recipes. In collaborative filtering frameworks, the user's preferences will be compared with other users and the similarity between them will be studied. The assumption that if user A preferences are similar to user B, then the recipes that user A rated higher that user B has not tried will probably be of user B's interest is considered.

Collaborative Filtering Collaborative recommendation algorithms can be separated in two classes: memory-based (or user-based) and model-based. For memory-based predictions, the user's previously rated items are studied. This information is used in order to find the users with the highest similarity. This is performed using a method the nearest neighbours method. The nearest neighbours search relies on the assumption that observations with similar characteristics will result in similar outcomes. Therefore, for each problem, several characteristics must be measured. The measured values are compared and the closest ones are clustered.

Unlike memory-based methods that consistently access the entire user database to make predictions, model-based use the database information to create a model to obtain the recommendation [17].

Memory based techniques utilize statistical methods for the search of users with similar transactions history to the active user [18].

The model based approach creates a model where user-item interactions are decoded and new predictions are made. In order to do so, two approaches can be used: probability or rating prediction. The modeling process is done using machine learning techniques such as classification and clustering [19].

In collaborative approaches as more interactions occur, the more accurate the predictions become. These predictions do not rely on the user or items detailed information. Using the nutritional problem as an example, for these approaches the recipe's ingredients, for example, will not be taken as part of the recommender method. These approaches have a significant disadvantage for specific scenarios. Since past interactions are used for the result, it is impossible to make recommendations for new users and recommend new items - cold start problem. These approaches obtain the recommendation through the comparison of the different users ratings, for example, therefore, it is not possible to use them for users with no rated items. This drawback might be handled using random or popular recommendations for new users or through the use of a non collaborative method for new users.

Content-Based Filtering Content-based approaches use information about the users and/or items. Considering a very popular recommendation system, movie recommendation, age, job or other personal user information as well as the movie's genre and its actors are used as information for the prediction. The goal is to build a model, using the features mentioned above, that explain the observed user-item interactions. The user and movies information is analysed and a profile is created. The system gathers all the movies the user rated and finds the similarity between the highest ratings. Considering a user whose highest ratings are Finding Nemo and The Lion King

the system should conclude the user is interested in animation movies whose story line is related to family and friendship. These are then used to find movies within the same genre.

This method has as problems the limited content analysis and over-specialization [12]. Unlike collaborative methods, the new users' drawback does not affect this approach.

Hybrid Models In recent years, the idea of combining the aforementioned techniques to make use of the advantages and fix the disadvantages have surfaced. These are called hybrid systems. These systems result from a combination of multiple techniques to achieve some synergy between them [20].

For example, a combination of collaborative and a knowledge-based system can compensate for the cold-start problem (collaborative) and the collaborative component can find unexpected niches in the users preference space. The knowledge-based component can provide recommendations for new users whose profile information is too sparse for the collaborative technique while the latter can find peer users who share unexpected niches in the preference space [20].

Over the past few years, deep learning has garnered a growing interest in many fields, for example, computer vision. It has shown its effectiveness in recommender systems research. According to [21], methods such as Multilayer Perceptron, Autoencoder, Convolutional Neural Networks and Recurrent Neural Network have been widely used in recommender systems from news to music and video recommendations.

2.1.2 Health Recommender Systems

Recommendation systems are very popular to improve selling, engaging users and helping them select items amongst the available offers.

However, the adaptation of these systems to health recommendation is a relatively new area. Therefore, the study of current methods used in recommendation systems and its application to health is a topic of great interest [22]. The application of these systems for the health domain has attracted researchers in the past few years. The different areas of recommendations go from exercise plans, diets, healthcare services to even assist decision-making tasks regarding the diagnosis, treatment and well-being of the patients [13]. As the possible risks and side effects of these decisions increase, the complexity of implementing such systems rises.

According to the Valdez framework [23], for the design of a HRS, three main aspects must be considered. First, the health domain must be understood. In order to do so, two questions must be answered:

- Which one of following items is recommended (nutrition, medicine, sport)?
- Who is it recommended to (the patient, the doctor, the nurse)?

The next is how to evaluate such system. Ethical implications as well as how to ensure patient trust and security are essential to the success of a HRS. Finally, the technical specifications and data

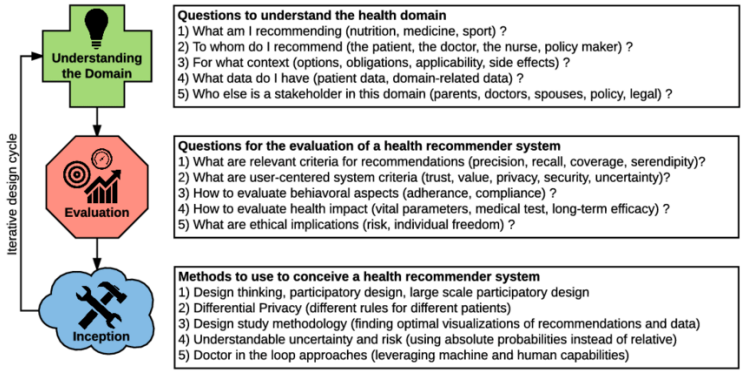

Figure 2.2: Three steps for HRS development (from [23]).

analysis methods used for these systems should be tested and understood. Figure 2.2 illustrates the main steps proposed by [23].

For HRS, rule based and ontology techniques are commonly used. Nutrition knowledge is suitable with the simplicity, transparency, and consistency of these techniques [6].

Besides, other algorithms are also applied for recommendations purposes in the healthcare domain, such as ant colony algorithm, classification, clustering, decision tree, logistic regression, support vector machines and matrix factorization [13].

2.1.3 Nutrition Recommender Systems

Nutrition is one of the basis for health and development of human life since the earliest stage of fetal development into old age [6]. A healthy lifestyle considering the nutritional necessities is an indubitable requirement for survival, development and well-being.

The concept of nutrition informatics results from the combination of information technology, information sciences and nutrition [24].

Due to the increase in the number of patients suffering from diseases related to bad eating habits, the study of food recommender systems has grown. Current studies focus on recommendations based on the user preferences and/or health problems. Nutritional consumption is also one of the most important functionalities in such a system [8].

According to [6], the recommender systems used in the nutrition field go from collaborative filtering, content-based to hybrid frameworks. For these systems, the most commonly used artificial intelligence techniques are rule-based approaches. Rule-based systems are one of the simplest form of artificial intelligence. These systems use human expert knowledge to solve real-world problems that would otherwise need human intelligence. This knowledge is generally presented in rules [25]. Ontology techniques have also have been used since the knowledge and concepts of relations can be dealt in a clear manner. An ontology is used to model the structure of a system (entities, relations). An example can be a company with its employees and their interrelationships [26].

Ontology techniques present a structural way of representing the concepts, relations, attributes and hierarchies present in the problem's domain [27].

2.1.4 Evaluation

To evaluate the performance of any recommender system, for each object recommended to the user, the proximity with the preferences of the user must be measured. Some recommendation scenarios have as a result an ordered list. In that case, the place that each object has on the list must also be taken into consideration.

2.1.4.1 Data Partition

To evaluate the performance of a recommender system, the data must be split in two different parts: the training set (for model construction) and the testing set (for model testing). The training set may be partitioned into a model and a validation set (tune the algorithm's parameter). This partitioning can be done using different techniques [28]:

- **Holdout** - a random subset of the data is the training set and the remaining the testing set.
- **Leave-one-out** - one of the items rated by the user is hidden and the model is built on the remaining data. The procedure is repeated for all ratings.
- **k-fold cross validation** - the users are divided in k partitions, k-1 are used for the model while the remaining is used for testing.

2.1.4.2 Metrics

In order to evaluate and compare different methods for recommendation systems several metrics can be used. This comparison is done between the recommendations done by the system with a predefined set of real-world user opinions.

A very commonly adopted metric is accuracy. Accuracy measures how similar are the predicted item ratings with the real ratings. This metric is used by calculating the Mean Absolute Error (MAE), Root of Mean Square Error (RMSE) or Normalized Mean Average Error (NMAE).

$$MAE = \frac{1}{N}\sum_{u,i}|p_{u,i} - r_{u,i}| \tag{2.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \tag{2.2}$$

For set recommendations, Precision, Recall and Area under the ROC(Receiver Operating Characteristic) curve (AUC) are obtained [29].

$$Precision = \frac{N_r s}{N_s} \tag{2.3}$$

Table 2.2: Categorization of recommended items for a user

	Selected	Not Selected	Total
Relevant	N_{rs}	N_{rn}	N_r
Irrelevant	N_{is}	N_{in}	N_i
Total	N_s	N_n	N

$$Recall = \frac{N_r s}{N_r} \tag{2.4}$$

Finally, for rank recommendations the half-life and the discounted cumulative gain are estimated [30]. In the past few years, the discussion has been the use of only accuracy measures for recommender systems. The substitution of this measure by other such as coverage, diversity, novelty, serendipity, utility, and scalability has been tested and encouraged [31].

Coverage measures the percentage of users the system can make recommendations to and/or the portion of items to be recommended.

Diversity measures the differences between recommendations. Regarding novelty and serendipity these parameters evaluate the surprise/unawareness of the recommended item to the user. Utility measures how useful a recommendation is, therefore its value. Two different scenarios can be evaluated, the value to the consumer and to the provider.

Scalability assesses how the method deals with larger datasets (processing power required and speed).

2.2 Optimization problem

As presented previously, the expected result for the nutrition problem consists on adapted meal recommendations for each person. These recommendations are dependent and multiple constraints must be considered for the final result. In order to take into account the relations and constraints between items/meals, optimization techniques may be adapted to solve the problem.

For the nutrition scenario, the goal is to obtain a weekly meal plan that optimizes several conditions such as nutritional needs, ingredient's variety and user preferences. Therefore, an explanation of important optimization concepts, essential to understand the optimization frameworks will be carried out in this chapter. Combinatorial and Multi-objective Optimization are two optimization scenarios that will be introduced. Some possible algorithms will be described.

Several practical problems can be expressed as optimization problems. These problems involve finding the best solution from a large set. The study of all possible combinations is time consuming therefore a quicker solution must be used. The current project goal is to develop a meal plan that fulfills the user needs and preferences. In order to create the best plan criteria such as energy intake, variety of fruits and other ingredients, number of white meat meals and recipes price must be adapted to the user conditions.

2.2.1 Combinatorial Optimization

Several optimization problems involve the search for the best configuration of a set of variables to achieve some goals. They can be separated into two categories: those where solutions contain real-valued variables and those with discrete variables. When dealing with discrete variables these problems are called Combinatorial Optimization problems.

A Combinatorial Optimization problem can be defined as $P = (S, f)$ [32], involving:

- a set of variables $X = \{x_1, x_2, \ldots, x_i\}$
- the variables domains $D = \{D_1, D_2, \ldots, D_i\}$
- constraints
- objective function to be maximised or minimised.

The set of all possible solutions is

$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \| v_i)) \in (D_i, s)$ satisfies as many constraints as possible$\}$.

In the search space, S, each element can be seen as a candidate solution. The goal in combinatorial optimization problems is to find the solution $s_* \in S$ with maximum objective function value. The project goal might be to find a single solution or a set of globally optimal solutions. $f(s_*) \leq f(s) \forall s \in S$. s_* is a globally optimal solution and $S_* \subseteq S$ is the set of globally optimal solutions.

In brief, a Combinatorial Optimization method is based on the search for a solution in a set for a multidimensional function in order to reach its maximum value.

2.2.1.1 Knapsack Problem

Package recommender systems, which will be presented in Section 2.4, combine recommender systems techniques and optimization methods. The knapsack problem is commonly used for these scenarios.

The knapsack problem is one of the most famous problems in combinatorial optimization. It is very common and, therefore, has been thoroughly studied. To understand the problem, an hypothetical hiking trip must be considered. The hiker needs to fill up the knapsack with multiple necessary objects, each with its weight (or volume) and value/necessity that quantifies its importance. However, the capacity is limited and therefore smaller than the number of possible items. The issue, then, is to figure out the combination of items that yields the highest total value [33].

Considering the knapsack capacity, C and a set $N = \{1, \ldots, n\}$ of items where each item has a profit p and a weight w, the Knapsack problem goal is to find the maximum profit subset of items where its total weight does not exceed the full capacity C. This problem can be formulated with the following:

$$max\{\sum_{i \in N} p_i x_i : \sum_{i \in N} w_i x_i \leq C, x_i \in 0, 1, i \in N\} \tag{2.5}$$

where each variable x_i takes value 1 when item i is inserted in the knapsack [34].

2.2.2 Approaches

Cai and Wang [35] separated the optimization methods into two categories:

1. Methods based on biasing feasible over infeasible solutions;

2. Methods based on multiobjective optimization techniques.

As for the first category, the original objective and the constraint violation are considered two separated objectives to be optimized synchronously. As for the second category, a multiobjective problem with m+1 objectives (m is the number of constraints) is applied.

Optimization problems may involve various general constraints. One of the most popular method is the penalty function. It was first introduced by Courant [36]. A penalty term is introduced in the original function to penalize constraint violations and with it the constrained optimization problem switches to an unconstrained one.

The optimization tecnhiques can also be separated in offline and online [37]. Regarding offline optimization, the data used is collected in the past. In this situation, a large number of evaluations can be done in search for a global optimum solution. However, since is optimized based on past data, its result at the time of deployment may be outdated. Also, once the system is optimized it does not react to posterior changes in the environment leading to a deterioration over time.

For online optimization, the evaluation is done in real time therefore considering system changes. The biggest disadvantage in this method is that it is time consuming since interactions must be done in real time and not using previously collected interactions. Time is also affected by noise related with real world testing. The crucial advantage of these methods is the optimization method adaptation considering the environment latest changes.

2.2.3 Multi-objective optimization

Multiobjective optimization is an area of multiple-criteria decision-making used when several objective functions need to be optimized simultaneously. One of its common application is engineering problems, where optimal decisions need to be taken in the presence of trade-offs between two or more objectives.Considering, once again, the nutrition scenario, in order to reach a correct energy intake the variety of ingredients might be compromised. However, its application extends to other fields of science. When considering the nutrition scenario, the user's restrictions and needs can be seen as objective functions and therefore multiobjective optimization can improve the overall framework.

Given m objective functions $f_1 : \mathrm{X} \to \mathbb{R}$, ..., $f_m : \mathrm{X} \to \mathbb{R}$ which maps a decision space (set of all possible solutions) X into $\mathbb{R}$, a multiobjective optimization problem is given by the following problem statement [38] :

$$\text{maximise } f_1(x), \ldots, \text{ maximise } f_m(x), x \in X \tag{2.6}$$

For a multiobjective problem there is a conflict between the objectives where the improvement of one will cause the deterioration of another. A single solution, capable of optimizing all objectives simultaneously, does not exist. The goal is to find the best trade-off solutions, called Pareto optimal solutions, as illustrated in Figure 2.3.

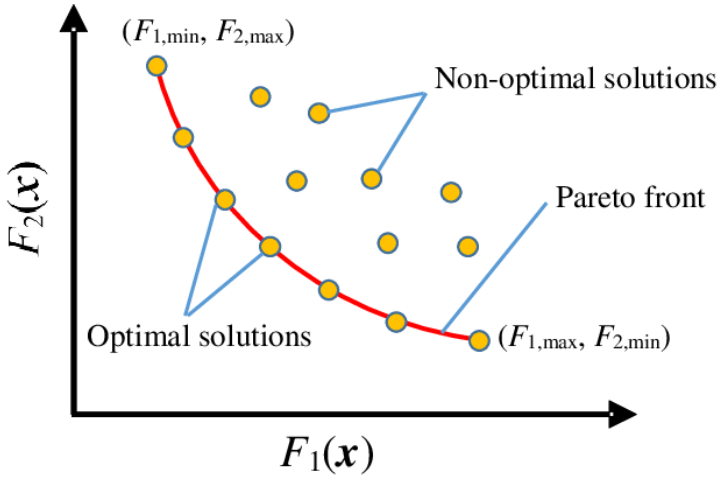

Figure 2.3: Pareto optimal solutions representation (from [39]).

In the last few years, an increase in constraint-handling methods based on multiobjective concepts investigation has been noticed. A multiobjective problem with constraints can be seen in one of the two categories, according to Section 2.2.2.

2.3 Genetic Algorithms

Metaheuristic algorithms are an algorithmic structure generally applied to optimization problems [40]. A majority of the developed metaheuristic algorithms are inspired in biological evolution process, swarm behavior, and physics' law [41]. These algorithms can be separated into two categories: single solution and population based metaheuristic algorithm. Single-solution based metaheuristic algorithms use a single candidate solution and improve it through local search while population-based metaheuristics utilizes multiple candidate solutions [42]. Genetic Algorithms are one of the most popular population-based metaheuristics algorithms.

Genetic Algorithms (GA) were created using the biological evolution process as inspiration. GA mimic the Darwinian theory of survival of fittest in nature. A GA is composed of a chromosome representation, a fitness selection, and biological-inspired operators [42]. Genetic algorithms are used as an optimization technique were the function to maximise (Section 2.2.1) is called the fitness function.

In Figure 2.4, a simple overview of the main steps for implementation of a genetic algorithm are illustrated.

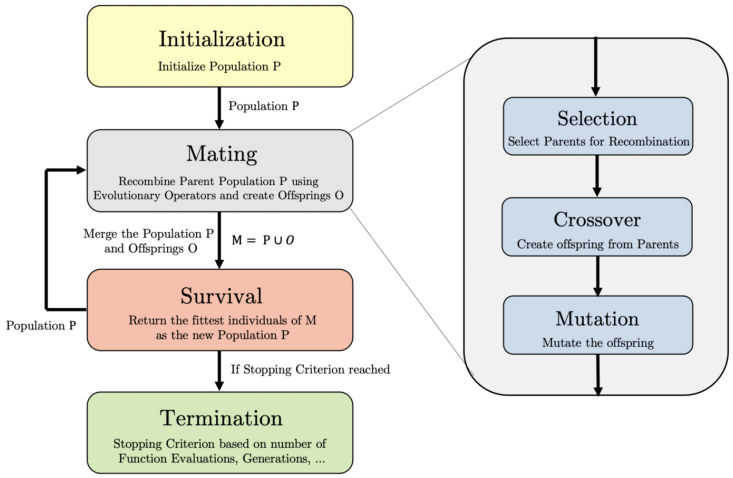

Figure 2.4: Overview of genetic algorithm's main steps [43]).

A genetic algorithm population is organized as multiple genotypes. These genotypes may have one or multiple chromosomes and each chromossome is composed of genes. Depending on the optimization scenario genes may take many forms starting in a binary gene (find the best path for a robot avoiding the obstacles) to a custom gene (meal plan project). An initial population is created and later suffers multiple modifications in order to obtain the optimized solution. A fitness function, adapted to the scenario, is used to assign a value for each genotype in the population.

The genotypes are processed with the use of genetic operators who iteratively replace the population. These operators are biological-inspired. They consist in: selection, mutation and crossover. Selection operators take the entire population and select the best N genotypes using the fitness value as evaluation. Crossover operators extract characteristics from both parents and create an offspring [44]. Crossover is performed by swapping corresponding segments of the parents. Mutation operators change random genes considering the values possibilities. Both crossover and mutation algorithms work on the basis of probability. The presented operators play an important role in the balance between exploitation and exploration.The searching space is exploited through the application of crossovers and explored through introduction of new genes using mutation.

2.4 Package Recommendation

As stated previously, recommendation systems are commonly single-item. However, there are multiple scenarios that require a bundle recommendation. When planning a trip, the itinerary must have multiple stops and when picking an outfit, several pieces of clothing must be selected. These are, then, two examples that require bundle recommendation.

For the present nutrition problem, the goal is to develop a week's worth meal planning. Therefore the expected result consist of a package recommendation. Single-item recommendation is not applicable in this scenario as the goal is to obtain a set of recipes to create a meal plan.

In this section, package systems and its most common uses are explored and the nutrition problem presented. To conclude, some common frameworks and previously developed algorithms for multiple domains are summarised and discussed.

2.4.1 Concepts

Beyond single-item recommendation, two other types of recommendations can be distinguished: group and package recommendation. These two recommendation problems have additional constraints to consider. In group recommendation scenarios [45], the result should take into consideration not individual users but a group. In these situations requirements/preferences of different users are considered for the result. This is still an open topic that needs to be further analyzed. One example of a group recommendation scenario is TV recommendation for a group of friends.

The other type of recommendation problem is package/bundle recommendation. Package recommendation can also used for a group of users. For this the goal is to recommend a group or sequence of items that fit well together. For these problems several user and item constraints must be considered and comparison between the suggestions must be made. These conditions make package recommendation scenarios a difficult problem to solve. One example is the recommendation of a complete meal where the meal variety, health problems and nutrition needs as well as allergies should be considered.

Package recommender systems may take two forms: complementary and sequential scenarios [9].

For sequential recommendations, one of the most studied domains is travel. When preparing a trip, points of interest should be set and combined. For travel planning, the user's goal is to visit as many places as possible, however, some restrictions may exist. The total available time for the trip and budget may be two of them. Therefore, considering the user's points of interest a route must be created. This route must consider, for example, the visiting hours of tourist spots. The sequence of stops should also minimise the total covered distance. The recommendation system should follow this rule. It is considered a sequential recommendation where the algorithm must focus on the order of items.

Regarding clothes recommendation, there is still a need for a top and bottom combination however the sequence of the items is of no importance. Clothes recommendation needs, therefore, a complementary package.

Generally, package recommender systems algorithms can be divided into three main phases [9]:

- **model learning -** user preferences and other aspects necessary for producing recommendations are analysed;
- **package creation -** mix and match of items into packages creating a package candidates list;
- **package selection -** evaluation and selection of packages from the candidates list.

Using the current project as an example, an overview of the main steps in package recommender systems is illustrated in Figure 2.5.

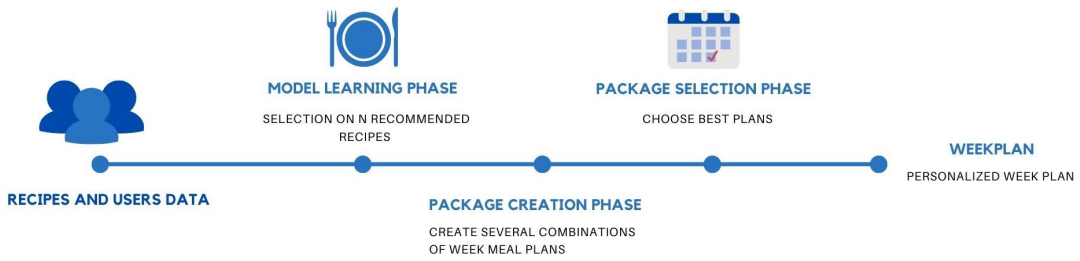

Figure 2.5: Overview of the steps for package recommendation.

For the meal plan scenario, the model learning phase can be seen as a study of the recipes and user information from the database. The goal is to choose the most relevant recipes for the user. After analysing the data, the next step would be to match recipes in a somewhat ordered plan. Each plan would then be evaluated in order to find the best combination of recipes to use as the week plan.

2.4.1.1 Model Learning Phase

The goal of this phase is to do a thoroughly study of the database and adapt the items to the user. The goal is to select, from all data, the items that may interest the user. For this, multiple techniques are tested and used. Some of the most used algorithms are clustering, CF, user preference modelling, item relationship modelling and topic modelling. For this step, single-item recommendation methods can be used since the goal is to select relevant data for the user. Package recommendation systems that use as input data the users' ratings, tend to test CF methods.

Some specific scenarios exist where other frameworks are used. When the input consists of unstructured data (text, for example), topic modelling techniques, such as Latent Dirichlet Allocation, are adopted. Other systems use item relationship modelling (Markov chain, probability model, Apriori and ontology) to model relations between the items. And finally, some systems do not require model learning since their package construction is not based on user preference but on item features and constraints [9].

For the travel recommendation scenario, this phase consists of selecting most relevant tourist attractions considering the user preferences. As for the outfit recommendation scenario, the result of this phase would be the suitable pieces of clothing for the user.

2.4.1.2 Package Creation Phase

Considering travel recommendations the goal can be seen as the need to cover as many stops as possible in an itinerary while minimising distance. The solution for these problems is generally, according to several researchers [9], a combinatorial optimization problem which maximises or

minimises certain features (according to the problem formalization) while taking into consideration its constraints.

Other possible interpretation is seeing package creation as clustering problem - the combination of items is reached through a common characteristic in a group of data.This view is mostly considered for complementary package creation. Considering the previously explained example for the development of an outfit, the clothes can be clustered considering, for example, their style (sporty, casual, punk).

For most cases, in order to make a correct combination one or multiple constraints have to be set. Once again considering the travel domain, the user budget can restrict the itinerary and should be taken into account. Constraints can be either manually constructed or learned [9].

Using, once again, the travel and outfit scenarios as examples, the result of this phase would be to combine the selected items in the model learning phase to best match the problems conditions. For travelling, a set of tourist points must be obtained considering the user's available time and money, for example. For the outfit pick, the pieces of clothing must be combined according to its color, style and pattern.

2.4.1.3 Package Selection Phase

To finish the process, package selection must be made. The package recommender systems select a number of packages from the list of candidates. The most common approach is Top-k – the k best packages are recommended. Some frameworks present a single package solution while others present a unranked list [9].

The final result for the travel scenario would the best itinerary where the distance should be minimised while the number of points of interest is maximised. The outfit scenario result must be the best combination of clothing that satisfies user's preferences and how the pieces match each other.

2.4.2 Package Recommendation Approaches

In order to analyse algorithms developed for package recommendation problems, an analysis of existing approaches was done according to several dimensions, such as domain, input data and evaluation metrics. In Table 2.3, a review of the obtained results is presented.

Domain and Recommender Sytem Type To analyse the collected data, for each approach several domains were studied. From books [52] [57], to movies [51] [58], travel [46] [59]and education [55] [60], algorithms were analysed and presented.

For each article, the separation between sequential and complementary recommender systems was made. In the majority of the domains, complementary systems (ex. [47] [51]) were created. The domain in which sequence is always present and a big variety of articles available is travel routes.

Table 2.3: Summary of evaluated frameworks.

Title	**Domain**	**Datasets**	**Package type**	**Features**	**Feedback**	**Constraints**	**Technique** Model training	Package creation	Package selection	**Evaluation metrics**	**Results**
Personalized Tourist Package Recommendation using Graph Based Approach [46]	Travel	check-in data Jie Pang (Public)	Sequential	No	Explicit	Yes	CF	Graph Based	Top-k	Longest Common Subsequence and Jaccard Similarity Coefficient	No comparison with other methods
Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation [47]	Fashion	check-in data Jie Pang (Public)	Complementary	Yes	NA	No	Convolution Neural Networks	Multi-layered perceptron	unranked	Mean Average Precision, Mean Reciprocal Rank and AUC	The model performs significantly better as given by the 95 percent confidence interval in the official ROUGE script.
ProductRec: Product Bundle Recommendation Based on User's Sequential Patterns in Social Networking Service Environment [48]	E-commerce	Amazon (Public)	Complementary	No	Explicit	No	Personalized Markov Chain	Stochastic Gradient Descent	Not specified	AUC, precision and recall	Tested with 4 baseline frameworks and AUC is approximately 0.02 higher for the proposed algortihm
Customized Bundle Recommendation by Association Rules of Product Categories for Online Supermarkets [49]	Supermarket	Company's Database (Private)	Complementary	No	Explicit	No	Apriori Algorithm	K-means	Top-k	None	No comparison with other methods
Modeling Buying Motives for Personalized Product Bundle Recommendation [50]	E-commerce	Transaction records from Tao Bao (Public)	Complementary	Yes	NA	No	Gibbs sampling	Bayesian Rule	Top-k	Discounted Cumulative Gain, Precision and Recall	Metrics are generally higher in the developed algorithm compared with the 4 other methods

Title	Domain	Datasets	Package type	Features	Feedback	Constraints	Technique			Evaluation metrics	Results
							Model training	Package creation	Package selection		
Learning from Sets of Items in Recommender Systems [51]	Movies	MovieLens (Public)	Complementary	No	Explicit		Matrix factorization (CF)	Stochastic Gradient Descent	Top-1	RMSE	Different developed algorithms tested. No external frameworks.
Mining Revenue-Maximizing Bundling Configuration [52]	Books	UIC dataset from Amazon.com (Public)	Complementary	No	Explicit	Yes	-	Knapsack	unranked	Execution Time	Comparison between the several algorithms developed, no external comparison.
Learning to embed songs and tags for playlist prediction [53]	Music	Last.fm (Public)	Sequential	Yes	NA	No	Markov chain	Knapsack	-	Precision and AUC	Compared with random and frequency baselines based on the dataset. No comparison between algorithms.
DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation [54]	Music	Million Song Dataset (Public)	Sequential	Yes	NA	No	MArkov Chain	Reinforcement Learning	-	Reward Function	Algorithm leads to a small but significant boost in performance
Recommendation Systems with Complex Constraints: A Course Recommendation Perspective [55]	Education	Student Transcripts (Private)	Complementary	Yes	Explicit	Yes	CF	Knapsack	Top-1	Precision	The algorithm makes better recommendation but takes a longer time to execute.
Constructing a Diet Recommendation System Based on Fuzzy Rules and Knapsack Method [56]	Food	Taiwanese snacks (Private)	Complementary	Yes	Explicit	Yes	fuzzy + JENA rules of inference	Knapsack	Unranked	Accuracy	The system' total accuracy is 72%. No comparison with external algorithms.

Title	Domain	Datasets	Package type	Features	Feedback	Constraints	Technique Model training	Technique Package creation	Technique Package selection	Evaluation metrics	Results
A Package Recommendation Model Based on Credit and Time [57]	Books	DouBan (Public)	Complementary	Yes	Explicit	Yes	Popularity, Personalization and Diversity algorithms	k–nn clustering	Top-k	F-measure	Three algorithms and its combinations are tested. The best result for F-measure is the combination of all three.
Clustering-Based Recommender System: Bundle Recommendation Using Matrix Factorization to Single User and User Communities [58]	Movies	Filmtrust 100k y MovieLens 100k (Public)	Complementary		Explicit		Matrix factorization (CF)	K-means	-	MAE(accuracy), Precision, Recall and F-measure	The developed algorithm is compared to a simple k-means clustering and is superior in all measures
Personalized Travel Package Recommendation [59]	Travel	Travel company in China (Private)	Sequential	-	-	Yes	Gibbs sampling	TAST (hierarchical Bayesian model)	Top-k	Degree of Agreement, Top-k and Normalized Discounted Cumulative Gain	Two comparisons made. 7 Benchmarks. Method performs better than other methods for all metrics.
Personalized course sequence recommendations [60]	Education	Mechanical and Aerospace Engineering department at UCLA (Private)	Sequential	Yes	Implicit	Yes	Clustering	forward-search backward-induction algorithm	Top-1	Package value (utility)	A study of the users of the algorithm is performed where the final GPA for the students is saved and compared with random algorithm or no selection

Input Data Some systems only used features (ex. [53] [54]) while other use implicit [60] or explicit feedback data (ex. [46] [58] or a combination of them (ex. [56] [55]). The used datasets and their privacy are shown. Analysing the results, most datasets are from an exterior source and mainly public (ex. [46] [48]). Another aspect regarding the input data was the existence of constraints (ex. [52]). All sequential recommender systems considered constraints while for complementary recommendations it was not essential however a considerable percentage used it for the final result.

Package Frameworks The analysis of the used techniques was done using the separation in three steps, explained in section 2.4.1. CF is the most used technique [55] [51] [46] for the model learning phase, however, techniques such as Markov chain [54] [53] [48] and Gibbs sampling [50] were also commonly used. For the package creation phase, a majority of the authors applied the knapsack problem, explained in chapter 2.2. Finally, for package selection most articles presented the Top-k created packages (ex. [49]) while others had as an output a single package(ex. [60]) or an unsorted list (ex. [56]).

Evaluation Metrics In section 2.1.4, accuracy was presented as the most popular evaluation metric, however, recently, other metrics have been studied and recommended. Precision, Recall and F-measure were used more than once, nevertheless a big variety of metrics can be observed.

In the last column, a short overview of the results was made. Some methods only compared the several proposals developed while other used previously frameworks as benchmarks. For all the approaches that compare the obtained results with previous works, there is an improvement however, for some situation the approach is computationally more expensive.

2.4.3 Nutrition Problem

In [7], a nutrition system was presented. A discussion on problems conditions and the developed steps is done in Section 2.1.3. The nutrition field can not be directly compared with the most famous package recommendation scenarios. While travel and clothes recommendation systems are very easily separated using the complementary and sequential recommendation analysis, the current project context does not fit in any of the two categories. Since a recommendation for a week's worth meal plan is necessary, several constraints and considerations are essential for a correct algorithm result. For this problem an unordered list is not the ideal output since it may concentrate, for example, several meat meals in a row which is not the healthiest scenario and the nutrition value of each day may also be unbalanced. However, a strict sequence is not necessary since a user can change a Monday meal to another weekday without any disadvantage.

This study presents a analysis of previously developed frameworks for several domains and can be used as a starting point for the project scenario.

Summary A study on recommender systems and optimization techniques has been presented. The different types of recommendations from individual to group and package recommendation were described.

Considering the nutrition problem, package recommender systems were studied since the development of a solution for the current problem may adopt a similar framework or adapt already developed algorithms. Several package recommender systems, for different domains such as travel and e-commerce issues were described and compared.

The study on package recommender systems show that they can be divided into complementary and sequential recommendation depending on the relationship between the package items. However, a gap exists for scenarios, such as described in Section 2.4.3, where a somewhat sequential connection must exist however not strictly followed. Therefore, the goal of this project is to adapt optimization algorithms for the current nutritional scenario.

Chapter 3

Methodology

The goal of this project was to develop an algorithm to create a weekly meal plan according to the user needs, restrictions and preferences. The project can be separated into two phases. First, a recommendation system where each recipe was scored considering previous interactions from the user was developed. These results should be incorporated in the already developed plan evaluation which will be explained in section 3.5.3.1. The main plan is the development of a optimization technique whose aim is to obtain the highest scored meal plan according to the previously developed heuristics. The present chapter will summarize the main methods used for the development of this project.

3.1 Previous Work

In [7] a nutrition recommendation meal plan was developed. In order to create a meal plan considering energy intakes as well as fat, carbohydrate and protein needs, the project development was done with the aid of a nutritionist. The SousChef [7] system has a mobile application, through which occurs user interaction. The implemented system uses a content-based approach and information retrieval techniques. The weekly meal plan creation can be separated in three steps:

- Calculation of nutritional requirements;
- Selection of candidates for each meal;
- Scale meals to match user's needs.

For each meal, a two phase process will result in selection of the best candidates. The first phase consists of, through the use of restriction rules, filtering the suitable candidates for the meal. The second phase goal is to select the most suitable candidate from the limited set. Each candidate will have a score obtained trough the calculation of different heuristic functions. These functions were implemented with the help of the nutritionist. The final score for a candidate is calculated as the weighted average of the values of each heuristic function and its respective weight. For each meal, the recipe with the highest score is picked. A temporary week plan is created and it is later

adapted considering the full plan. Some recipes might be substituted in order to better fulfill the conditions. The recipes are then scaled considering the user's nutritional necessities. This work and the detailed process is presented in [7].

3.2 Project Architecture

The aim of this project is to develop an algorithm for the recommendation of a week meal plan considering the user background and necessities. A developed algorithm [7] will be used to compare this project's results. The system can be separated into two components:

- Recommender system;
- Optimization technique: Genetic Algorithm.

The goal of the system is to combine a recommender system results with several others conditions and restrictions to obtain a week meal plan where the user's information is considered for the result. In order to do so, off-the-shelf recommender system methods were used to obtain the recipes recommendation predictions. These were then added as an additional criteria for the evaluation of the week plans. A genetic algorithm, where the implementation of problem-specific components must be studied, was developed as the optimization technique to reach a plan that optimizes the overall evaluation of all criteria.

The two algorithms were developed in different environments (*Java* and *Python*) therefore a connection between the two had to be created.

The overview of the system's architecture is illustrated in Figure 3.1.

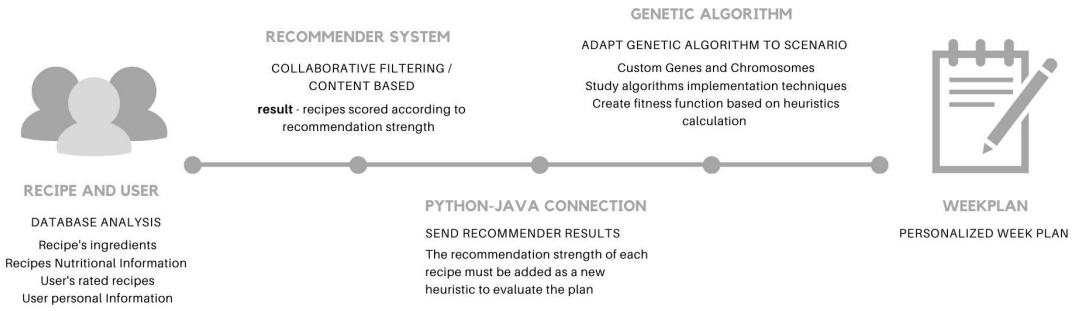

Figure 3.1: System's architecture.

3.3 Recipes and User Database

The data to test the developed algorithms had been already created. The available recipes were previously created with the help of a nutritionist. A selection of ingredients and posterior recipe

creation were the main steps performed. The used ingredients were based on the Portuguese Food Composition database elaborated by INSA [61] which contains not only the ingredients but also their nutritional composition. The database was adapted to the target population specific energy and nutritional daily intakes [7]. Each meal of the day is subdivided in meal divisions and for each multiple recipes are suitable. The database also contains information about the system's users.

Regarding the user's information, it includes not only the user's personal information such as height, weight, date of birth and gender but also their nutritional needs, food preferences and restrictions, activity levels and diseases. Another information stored in the database is the user meal planning history to avoid repetition between plans [7].

In order to correctly analyse the plan, several measurements must be considered. The nutritional value of each recipe, the avoidance of comfort food, the minimisation of ingredients categories to easy the shopping task, etc, are considered to score each recipe and therefore play an essential part in the final result. Not only are there heuristics used for each meal part but also daily and weekly heuristics are considered. The variety of fruit for each day and the integration of white meat recipes during the week were the two scenarios contemplated for the day and the week heuristics, respectively.

3.4 Meal Recommender System

As stated previously, one of the goals of this work was the development of a recommendation system in order to improve the final week plan. This recommendation would make the algorithm more personal since each recipe would have a score to estimate the user preference for the recipe.

To create a system adapted to the project scenario, several recommender systems algorithms were tested. In chapter 2.1, the main recommender systems frameworks were presented. For this work a content-based, a collaborative filtering and a hybrid algorithm were created and evaluated.

3.4.1 Data Preparation

With the users and recipes data, the information was processed. All the recipes data was maintained. As for the users interactions, several users have a sparse number of ratings. Recommendation systems rely on the user and items information. As explained in chapter 2, some recommender systems frameworks suffer from the cold-start problem, therefore, it is hard to provide personalized recommendations for users with none or a very few number of rated items. This situation hampers the use of information to create the model considering the user's preferences. To avoid this problem, only a percentage of the users (20 users with the most ratings) were used to develop the recommendation model.

3.4.2 Content-based Model

Content based recommender systems rely on the attributes and description of the user interacted items. This type of algorithm is, therefore, only dependent on the user's choices, making it robust

to the cold start problem. Considering the current project, the user's choices consist of rated recipes.

The goal is to convert unstructured text and words (recipes data) into a vector structure. The information used to study the recipes similarity was their names and ingredients. For this work, a very popular technique for information retrieval called TF-IDF was used. TF-IDF stand for term frequency-inverse document frequency and was created for document search. With the use of the TF-IDF technique, in the obtained vector structure each element is represented by a position in the vector and its value measures the relevancy of the word in the recipe. In order to use this algorithm, the *TfidfVectorizer* from the *sklearn* library was used. For the *TfidfVectorizer*, the vectors elements can be single words but also strings up to a maximum of six words.

To model the users profiles, all the their interacted items are collected and their average rating is computed. The recipes to which the user gave a higher rating will also have a higher strength in the final profile. This final profile will then have multiple items the user has interacted with and their corresponding strength.

The content based model will then take the created profile and compute the cosine similarity between the profile and the vectors for all the items in order to find the closest matches.

Cosine similarity is used to measure the similarity between two items. Considering two vectors, x and y, the cosine similarity is calculated in Equation 3.1.

$$sim(\overrightarrow{v_1}, \overrightarrow{v_2}) = cos\phi = \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \, \|\overrightarrow{v_2}\|} \tag{3.1}$$

This measure computes the cosine of the angle between vectors v_1 and v_2. A cosine value of 0 means that the two vectors are orthogonal and have no match. The closer the cosine value is to 1, the greater the match between vectors.

The items with a higher cosine similarity will be the model's recommended recipes.

3.4.3 Collaborative Filtering Model

As presented in chapter 2.1.1.1, Collaborative Filtering has two implementation strategies: memory-based and model-based.

A model was developed using an off-the-shelf algorithm for this project. This approach compresses a user-item matrix into a low-dimensional representation in terms of latent factors. Therefore, a high dimensional matrix with multiple missing values will be replaced by a smaller matrix in a lower-dimensional space. Singular Value Decomposition(SVD) is used for this work.

SVD is a matrix factorization technique commonly used for dimensionality reduction in machine learning. For collaborative filtering, the used matrix contains the users evaluations for each item. Given an $m \times n$ matrix A, with rank r, the singular value decomposition is calculated through the following equation:

$$SVD(A) = U \times S \times V^T \tag{3.2}$$

Where U,S and V are of dimensions m × m,m × n, and n × n, respectively. Since matrix S contains r non zero entries, the effective dimensions of the U, S and V ares m × r, r × r, and r × n, respectively. U and V are two orthogonal matrices and S is a diagonal matrix, called the singular matrix [28].

A matrix decomposition is performed using the *SciPy* library, which helps the interactions representation between users and items. The number of used factors for the user-item matrix must be set. Some compromises must be done since the higher the value, the more precise is the factorization, however, if too many details are memorized it will be specific for the data and not a generalized model. After the factorization,the original matrix is reconstructed through the multiplication of its factors. The highest predicted ratings are then recommended to the user.

3.4.4 Hybrid Model

Finally, a hybrid model that considers both frameworks and combines them is created. Hybrid methods have shown great results and the interest in these systems has increased amongst researchers. A simple method is developed that calculates the weighted average of the normalized collaborative filtering and the content-based scores. The top N recommendations for each model are obtained and the final score is calculated.

3.5 Genetic Algorithm

The development of this algorithm was done using *Java*. Previous work had been done in this environment. This work included the heuristics to evaluate the week plans as well as the algorithm to remove recipes that match the user's dietary restrictions.Therefore, this step was created using *Java* in order to reuse this information.

3.5.1 Population

In chapter 2.1, an introduction to genetic algorithms and their components was made. Since the genetic algorithm was developed in the *Java* environment the *Jenetics* Library was used. This library contains all the necessary classes for use of these algorithms. The first step in a genetic algorithm is the creation of the initial population. Each individual is defined by its genotype that contains the chromosomes and these contain the genes. For the week plan problem, each gene consists in a recipe and only one chromosome is used that represents the full week plan. Therefore, the solution is a one chromosome genotype with the best set of genes (recipes).

The *Jenetics* Library is prepared for bit genes, double genes, character genes, etc. However, for this scenario, the gene must contain the recipe information, therefore, a custom gene class and a custom chromosome class were created to ensure that all the recipe information was saved and to avoid time consuming steps. These classes have a factory in order to create new instances of genes and chromosomes.

Since the desired result is a week plan that considers all week days and all meal divisions, each gene has specific constraints considering not only the meal division but also its parts. Therefore, a chromosome is composed of genes with different constraints considering the meal part and division. When creating a new instance of a gene, only some recipes can be considered. In order to do so, for the creation of the new gene the meal scope must be given as input. The meal scope consists in the necessary meals to be planned. The scope can be seen as an meal plan with only empty slots. Therefore, for each day, for each meal division and for each meal part a recipe must be chosen. Using this information, for each gene only a fraction of the recipes database is considered. This means that, for example, when choosing the fruit for lunch only the fruit recipes are considered. The user restricted ingredients are also removed from the candidates.

3.5.2 Operators

The library has already implemented selection, mutation and crossover operators. For the selection only the offspring fraction must be set. The offspring fraction dictates the percentage of individuals from generation N available to be mixed in generation N+1.

As for crossover, the two most popular algorithms to be used are: single-point crossover and multi-point crossover. In single point crossover, each child can be separated into two parts, one from each parent. As for multi-point crossover, the combination between the two parents results in a child with multiple parts of the chromosome from each parent (depending on the number of crossover points).

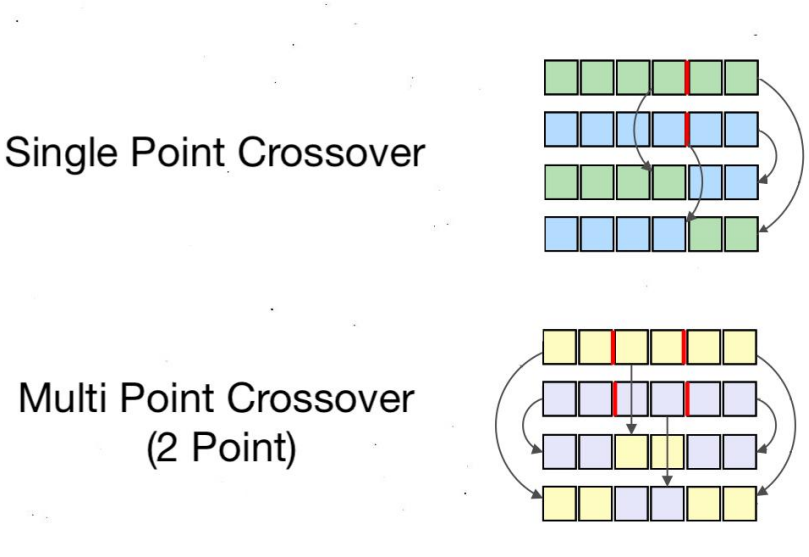

Figure 3.2: Crossover Operators.

The library has implemented several types of mutation. However, for this scenario, only one of the mutators operators can be used since each gene has specific constraints and consecutive

genes can not be switched, for example. Therefore the used mutator consists in a simple gene substitution where a gene is replaced by a new instance (taking into consideration its constraints).

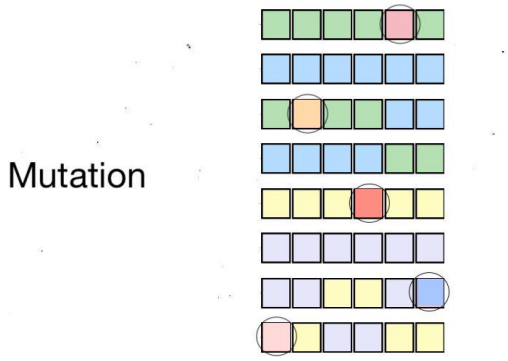

Figure 3.3: Mutation Result.

Finally, selection operators are also available in the library.For each generation, new genotypes change while some are maintained and others are discarded. The selection of the genotypes to keep and to mix can be done using several selectors. The library has multiple types of selection operators implemented, some based on probability and others based on the fitness value. Two selections must be performed: offspring and survivors to determine the offspring and unchanged phenotypes (between two iterations) to maintain, respectively.

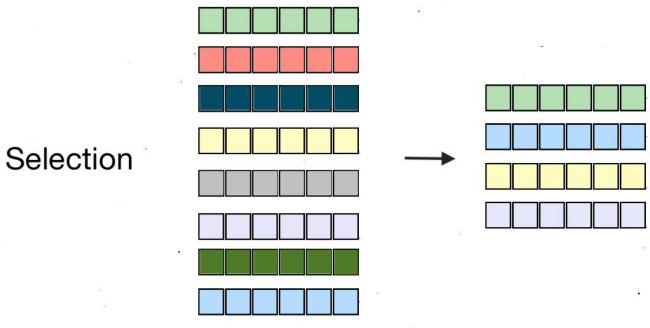

Figure 3.4: Selection Result.

3.5.3 Fitness Function

To evaluate the population at each generation, a fitness function must be created where each genotype has a corresponding value as an evaluation. The fitness function, for an optimization problem,

is the function to be maximised or minimised according to the problem's conditions. The previously developed algorithm used heuristic functions not only for evaluation proposes but also as a guide to the algorithm's final result. The developed greedy algorithm had, therefore, an already developed evaluation available [7]. This evaluation was used in the calculation of the fitness function. Considering Equation 2.6, each heuristic is a function to be maximised and the obtained fitness value is a result of the maximisation of the multiple conditions.

3.5.3.1 Heuristics Calculation

The goal of the genetic algorithm is the maximisation of the fitness function. As stated previously, the fitness function evaluates the week plan as a whole, however, in order to get the week score, it uses each recipe score and also the score of each week day.

Each heuristics calculation follows a different purpose. All heuristics results have the same minimum and maximum values. Some calculations were already developed within the desired interval while others were normalized for the [0,1] space. The fitness function combines all calculated heuristics in order to obtain the week score.

Next, the used heuristics are presented along with a brief summary of its calculation and respective weights (for the fitness function initially developed for the greedy algorithm).

Carbohydrate Suitability Heuristic Measures how the the carbohydrate necessities are fulfill by the chosen recipe.

Calculus - The carbohydrate value of the recipe is compared with the meal division's necessities and a score is calculated accordingly.

Weight - 1.2

Cheaper Food Heuristic Test the price of the recipe valuing cheaper options.

Calculus -The score is calculated based on the recipe's price when compared with the meal division set price.

Weight - 1.0

Diabetes Heuristic Test recipe's suitability for diabetic users.

Calculus - A list of unsuitable recipes for diabetic diets is available. Binary score based on the recipe's suitability.

Weight - 20.0

Energy Suitability Heuristic Measures how much the chosen recipe fulfills the energy values for the meal division.

Calculus - The recipe's energy value is compared with the meal division necessities and a score is set.

Weight - 1.2

Fat Suitability Heuristic Measures how much the chosen recipe matches the fat necessities for the meal division.

Calculus - The recipe's fat value is compared with the meal division necessities and a score is set.

Weight - 1.2

Food Restrictions Enforcement Heuristic Double check for restricted ingredients in recipes. **Calculus** - Accessing the user's restricted ingredients, a binary score is set by evaluating whether the recipe contains or not restricted ingredients.

Weight - 10.0

Fruit Variety Day Heuristic Evaluates daily amount of fruit types.

Calculus - According to the number of fruit types (up to a maximum of 3), a score is given. A lower score is attributed to non fruit desserts and to the most repeated fruit types.

Weight - $\sum weights_{mealOfDayHeuristics}$

Include White Meat Heuristic Count the weekly number of white meat meals.

Calculus - Two weekly white meat meals is the proposed goal. If the goal is not achieved, the white meat meal is scored higher.

Weight -

$$weight_{dayHeuristic} \times numberofdaysinplan \tag{3.3}$$

Ingredient Variety Heuristic Test plan's ingredient diversity.

Calculus - Penalization of recipes with already used ingredients taking into account the recentness of the ingredient's use.

Weight - 1.0

Meat Fish Separation Heuristic Check if dinner meals have fish recipes while lunch meals have meat meals. Favour lighter dinner meals.

Calculus - A score of 1 is given to the a set of recipes where the lunch meal is meat while dinner is fish. A specific score is set for double meat and double fish recipes for the day.

Weight - 1.0

Minimize Product Categories Heuristic Avoid a huge variety of ingredients in a week plan.

Calculus - For each new recipe, if it contains a new product category ingredient the recipe's score is set to 0.

Weight - 1.0

No Comfort Food Heuristic Avoid comfort food in the meal plan.

Calculus - The user's comfort foods are set in the database. If the recipe is in the user's comfort food list its score is set to 0, otherwise to 1.

Weight - 5.0

Protein Suitability Heuristic Measures the protein values for the meal division.

Calculus - The protein value of the recipe is compared with the meal division's necessities and a score is calculated accordingly.

Weight - 1.2

Same Recipe For Meal Division Constraint Heuristic The user can specify certain meals that consumes daily. For example, a user may want the breakfast meal to always be the same.

Calculus - The user's specific recipes are favoured compared with the other recipes in the database.

Weight - 9.0

User Preferences Heuristic Favours preferred meals according to the user's information.

Calculus - The score is directly attributed from the database. All recipes with no preference specified are scored 0.

Weight - 1.5

The heuristics calculation and respective weights were modified in order to study the genetic algorithm performance. These changes will be explained in detail in Chapter 4.2.3.

The genetic algorithm's fitness function used the same evaluation as the previous algorithm not only because it measures important heuristics created with the help of a nutritionist but also because the use of the same function allows for a direct comparison between the final results of the two approaches.

For the calculation of each heuristic function, the candidate recipes and the meal planning context are considered. The meal planning context contains the user's general information, previous meal plans, nutritional necessities, food prices, amongst others.

The used function output consist of a week score, where the combination of all the recipes is evaluated but also, daily and recipe scores. For the fitness score only the week score is considered which is a pondered result of the days and recipes scores.

3.5.4 Python Connection

The developed recommendation system results had to be imported to the *Java* code. In order to do so, a connection between *Python* and *Java* was created. The goal was to export from java the user's ID for it to be used to access its information and provide a recommendation. In brief, a process to access *Python* is created in *Java*. The *Java* code runs the recommendation engine which will use the user ID to connect to *Python* and get the results. After the recommendation is completed, since the result consists in a list of recipes with the respective strength (likelihood of being liked by the

user), the information must be stored. A csv file is then created and is subsequently read by the *Java* code. As a result, a matrix with recipes and their corresponding recommendation strength is available in Java to be used. This information can be used to select the best N recipes and only use those for the week plan creation or, a more diverse and less strict scenario, used as an additional heuristics for the plan's evaluation.

Summary In this chapter, the main methods used for this project were presented as well as the connection between the two used environments. In the next chapter, the evaluation of both the recommender system and the genetic algorithm will be described followed by the results. These will be analysed and discussed.

Chapter 4

Evaluation and results

In this chapter the obtained evaluation and results are presented for both the recommendation system and the genetic algorithm. The results are then analysed. The recommender system will be analysed, starting with the presentation of the evaluation techniques used followed by the presentation results. Next, the genetic algorithm evaluation is detailed and the results shown and examined. To conclude, a summary of the obtained results for both techniques is done.

Experimental Setup As explained previously, the goal of this project was to develop an alternative approach to the work done in [7] and test it in real world data. In [7], a private dataset was created that contains the users and recipes informations. This dataset consists in 142 Users and 427 recipes. For each user data such as age, weight, height, ingredients restrictions and preferences are saved. Regarding the recipes, the ingredients, nutritional values, price, ingredients category amongst other information are stored.

4.1 Recommendation System

The main goal of the development of a recommender system was to study whether off-the-shelf algorithms would be successful for single meal recommendation. Since two approaches - content based and collaborative filtering - were developed, the comparison between the two to study which was more adapted to the meal recommender scenario was also one of the aims of this step.

The data described previously was used to evaluate the approach. In addiction to the users and recipes information, the dataset also contains the users interaction, that is, the recipes rated by the users and its value. This information was used for the development and posterior evaluation of the approaches. 2996 ratings are saved in the database. However, since the collaborative filtering method suffers from the cold start problem, only the 20 users with the highest number of interactions were considered. Therefore, instead of 2996 interactions, for the recommender system development and evaluation only 2023 ratings were considered.

The developed frameworks must be evaluated in order to test its success. One of the most important things to evaluate is the generalization of the model. A good algorithm must work in

data where it was not trained otherwise it means it is too tailored to the data. For this step, a simple cross-validation method called holdout was used. The used data was split in train (80%) and test (20%). The models were trained on the train set and later tested on the test set.

Since the result of these methods is the Top-N recommended items, for the evaluation, 50 items which the user has not interacted with and 1 positively rated item are put together. The recommendation is then performed for those 51 items and the scores are obtained.

The position of the interacted item in the Top-51 will be evaluated. The used evaluation metric was Recall. The evaluation was done for the top 5 and the top 10 therefore recall@5 and recall@10 were used as the evaluation metrics. The evaluation was performed on the data from the test set.

In order to review and compare the intermediate results for the three developed methods, a single user from the database was chosen as an example to illustrate the intermediate steps of the developed system. In order to be tested for all methods, the user must have a significant number of reviews or it will suffer from the cold start problem. In order to present and analyse the intermediate results, a user with 82 ratings was chosen.

The content-based algorithm was described in Chapter 3. To illustrate the algorithm user 91 was used. The content based algorithm creates the user's profile based on their ratings. Through the study of the recipes ingredients and names combined with the user's ratings, a user profile is created.

This profile comprises the relevance of the elements to the user. For user 91, elements such as cheese and tomatoes have the higher relevance (Table 4.1).

Table 4.1: Results for user 91's profile.

Ingredient	**Relevance**
queijo	0,122
tomate	0,109
azeite brócolos	0,104
ameixa	0,102
estufado	0,097
trigo	0,096
frita arroz	0,089
brócolos	0,089
arroz cenoura	0,089
pão trigo	0,089

This information is used to then analyse the recipes for recommendation. The obtained results for user 91 are illustrated in Table 4.2.

For the collaborative filtering method, a model is developed. The results for user 91 are presented in Table 4.3.

The hybrid model is a simple combination of the recommendations of both models. The results for user 91 are illustrated in Table 4.4.

Table 4.2: Top-10 recommendations for user 91 using the content based method.

id Recipe	Name	Recommendation Strength
249	Pescada frita, Arroz de tomate com azeite	0,354
927	Linguado frito, Arroz de tomate com azeite, Azeite e Brócolos cozidos	0,353
105	Peito de Frango sem pele estufado com azeite, Puré de batata, Azeite, Brócolos cozidos e Couve-flor cozida	0,321
717	Pescada frita, Arroz de tomate com azeite e Brócolos cozidos	0,312
843	Lula estufada com cebola, tomate e azeite, Puré de batata, Azeite, Cenoura cozida e Ervilhas grão, congeladas cozidas	0,302
984	Peito de Frango sem pele estufado com azeite, Arroz de cenoura com azeite, Azeite e Brócolos Cozidos	0,301
48	Peito de Frango sem pele estufado com azeite, Puré de batata, Azeite, Cenoura cozida e Ervilhas grão, congeladas cozidas	0,301
1141	Pão de trigo integral com requeijão 8% de proteína	0,294
42	Peito de Peru sem pele panado, Arroz de tomate com azeite e Brócolos cozidos	0,293
144	Lula estufada com cebola, tomate e azeite, Puré de batata, Azeite, Cenoura cozida e Ervilhas grão, congeladas cozidas	0,292

Table 4.3: Top-10 recommendations for user 91 using the collaborative filtering method.

id Recipe	Name	Recommendation Strength
43	Maça sem casca	0,980
105	Peito de Frango sem pele estufado com azeite, Puré de batata, Azeite, Brócolos cozidos e Couve-flor cozida	0,833
84	Lula estufada com cebola, tomate e azeite, Puré de batata, Alface crua e Tomate cru	0,820
825	Lombo de Vitela assada com azeite , Castanha assada com sal, Azeite, Grelos de couve cozidos e Cenoura cozida	0,782
120	Vaca para Cozer ou Estufada , Puré de batata, Azeite, Cenoura cozida e Ervilhas grão, congeladas cozidas	0,771
67	Kiwi	0,769
927	Linguado frito, Arroz de tomate com azeite, Azeite e Brócolos cozidos	0,769
183	Salmão grelhado, Arroz cozido simples, Feijão verde fresco cozido e Molho verde	0,751
1327	Sopa de feijão com agrião	0,746
127	Clementina	0,744

Table 4.4: Top-10 recommendations for user 91 using the hybrid method.

id Recipe	Name	Recommendation Strength
43	Maça sem casca	0,735
105	Peito de Frango sem pele estufado com azeite, Puré de batata, Azeite, Brócolos cozidos e Couve-flor cozida	0,705
84	Lula estufada com cebola, tomate e azeite, Puré de batata, Alface crua e Tomate cru	0,685
927	Linguado frito, Arroz de tomate com azeite, Azeite e Brócolos cozidos	0,665
120	Vaca para Cozer ou Estufada , Puré de batata, Azeite, Cenoura cozida e Ervilhas grão, congeladas cozidas	0,638
825	Lombo de Vitela assada com azeite , Castanha assada com sal, Azeite, Grelos de couve cozidos e Cenoura cozida	0,629
183	Salmão grelhado, Arroz cozido simples, Feijão verde fresco cozido e Molho verde	0,619
1154	Pão de coração (trigo) com requeijão 8% de proteína	0,618
249	Pescada frita, Arroz de tomate com azeite, Azeite e Brócolos cozidos	0,606
720	Dourada grelhada, Batata cozida, Cenoura cozida e Molho verde	0,606

4.1.1 Discussion and Final Results

Table 4.5 reviews the obtained results using the previously presented metrics. Analysing the overall results, is it clear that the hybrid model presents the best possibility to use as a single meal recommender system. From the two off-the-shelf approaches, the content based had lower recall values. The use of the recipe's ingredients and names and the separation in words and strings might not be a great method to evaluate the user's likes. Other information such as type of cuisine, time to make, descriptive characteristics of the recipe such as salty or sugary may be an option to increase the validity of the technique.

The collaborative filtering technique showed higher results however they might be improved. The sparse number of users with a significant number of ratings is an obstacle for the effectiveness of the algorithm.

Since the collaborative filtering model performed better than the content based, its weight in the hybrid model had to be higher in order to obtain a better result. The final weights for each model were: 0.25 of the content based and 0.75 of the collaborative filtering.

The developed recommendation system methods considered very simple techniques and made assumptions that may hinder the final results. The development of a system more adapted to the

Table 4.5: Summary results for the three models.

Model	Recall 5	Recall 10
Content-based	0.1481	0.2938
Collaborative Filtering	0.3679	0.4864
Hybrid	0.3728	0.5407

current scenario may improve the overall results.

4.2 Genetic Algorithm

For the development of this approach several hypothesis and studies were performed in order to thoroughly evaluate how operators, population size and fitness function alterations would influence the obtained plan result.

A study of multiple operators for the genetic algorithm was done in order to evaluate how the adjustment of several parameters would alter the overall results. To test how the algorithm adapts to different restriction scenarios, two users were evaluated to assess the impact on the algorithm's performance. The final study was the algorithm's robustness to modifications in the fitness function. Since the heuristics calculation and weight had been set in previous work, the goal was to evaluate the adaptability of the algorithm to different scenarios.

For each test the algorithm's running time, the final fitness value, the number of iterations and a detailed analysis of the different heuristics values were saved.

4.2.1 Tested Users

An evaluation of both algorithms when dealing with different users was considered. As stated previously, the goal was to study how the greedy approach and the approach presented in this project would perform for two different scenarios. The idea was to test how ingredients restrictions and diabetes would affect the performance of the algorithm. The existence of ingredients restrictions reduces the number of possible week plans. If a user has multiple ingredients restrictions, for example: rice, milk and chicken, the number of candidate recipes for each meal significantly slims, impacting the overall creation of the week plan. The presence of diabetes also limits the number of candidates. The results of these two limitations must be studied in order to evaluate how the algorithm behaves in two different search spaces. The overall week result is not comparable because for diabetic users, as explained in Chapter 3, the diabetes heuristic is considered therefore each recipe has an additional heuristic to complete the evaluation.

As presented in section 3.3, the user information contains their height, weight, date of birth but also possible food allergies or restrictive diets as well as disease information such as diabetes. The performance of the algorithms in each scenario was tested for a user with few restrictions and no diabetes, referred to as *normal* and for a user with multiple ingredients restrictions and with diabetes, referred to as *restricted*.

The two users could be either created from the beginning or the database users could be examined and two chosen for the tests. In order to simulate a closer real life scenario, the users where retrieved from the database, which was created considering several assumptions. The two chosen users were user 6 and user 9.

As introduced in section 4.2, a standard user with little to no restrictions and no existing conditions and a more limited user, with several food restrictions and with diabetes were the two chosen users to test. A summary of the two users information is illustrated in Table 4.6.

Table 4.6: Summary of the two users tested.

	restricted	normal
Gender	Male	Male
Date of birth	1980-12-11	1939-01-04
Height (kg)	90	60
Weight (cm)	174	172
Restrictions	64 ingredients	2 ingredients
Diseases	Diabetes	-

The results shown in Table 4.7 illustrate the heuristics differences between the two users. The results were obtained considering the fitness function modifications which will be presented later.

Table 4.7: Heuristics scores for each algorithm and each user.

	Population Size 20		Population Size 200		Greedy Algorithm	
	User 6	User 9	User 6	User 9	User 6	User 9
Carbohydrate Suitability Heuristic	0,24	**0,55**	0,38	**0,69**	0,46	**0,90**
Cheaper Food Heuristic	**0,68**	0,66	**0,85**	0,76	**0,90**	0,65
Diabetes Heuristic	0,96	0,00	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,20	**0,51**	0,38	**0,66**	0,47	**0,73**
Fat Suitability Heuristic	0,36	**0,46**	0,40	**0,58**	0,45	**0,72**
Food Restrictions Enforcement Heuristic	1,00	1,00	1,00	1,00	1,00	1,00
Fruit Variety Day Heuristic	1,00	1,00	0,86	**0,99**	0,01	**0,99**
Include White Meat Heuristic	1,00	1,00	1,00	1,00	0,00	**0,75**
Ingredient Variety Heuristic	0,67	**0,72**	0,50	**0,65**	0,19	**0,34**
Limit Main Dish Recipe Repetition	0,99	1,00	1,00	**1,00**	0,88	**0,92**
Litre Of Soup In A Row	0,45	0,45	**0,51**	0,48	0,49	**0,52**
Meat Fish Separation Heuristic	0,52	0,52	0,54	0,55	0,55	0,57
Minimize Product Categories Heuristic	1,00	1,00	1,00	1,00	1,00	1,00
No Comfort Food Heuristic	0,96	**0,99**	0,94	**1,00**	0,94	**1,00**
Protein Suitability Heuristic	0,21	**0,44**	0,40	**0,59**	0,41	**0,68**
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	**0,62**	0,52	**0,64**	0,53	**0,68**	0,53

Analysing the obtained results, is it clear that the introduction of restrictions in a user's profile interferes with a vast number of heuristics. The heuristic's results that measure nutritional intake significantly deteriorate in the presence of a restricted user. With restrictions, the number of available recipes to choose from is smaller which can explain why the algorithm performs best when the restrictions are very sparse. A higher number of recipes leads to a higher number of possible combinations for the plan.

The introduction of changes in some heuristics calculations did not affect user 9 as it affected user 6. Two heuristics - the include white meat and fruit day variety - perform significantly better for user 9.

The overall results show that the introduction of restriction deteriorate the performance of multiple heuristics. For user 6, the only heuristic for which both algorithms had a better performance compared to user 9 was the user preferences heuristic. The introduction of restrictions has no impact in the this heuristic. The user will not rate a recipe that has an ingredient from the restrictions list therefore, this heuristic only dependent on the number of recipes the user has rated.

4.2.2 Genetic Algorithm Operators

The study of the best combination of operators and hyperparameter values was done to evaluate the performance of the genetic algorithm in the different scenarios.

In order to study the algorithm's behaviour for differences in the fitness function (calculus and weights),first the algorithm's operators must be tuned to find the best combination. The *Jenetics* library has several different operators to change and test, such as:

- Crossover
- Mutation
- Alterers
- Population Size

Crossover For the crossover, both single point and multi point crossover can be added to the algorithm. Other than the type of crossover its probability can also be set. To begin with, the crossover rate was tested. Two crossover rates were chosen: 0.6 and 0.8. These values were set taking into consideration the used rates in [62] and [63]. Both the use of a multi point crossover and a higher crossover rate should be associated with a higher exploitation of the search space. When using multi point crossover, each resulting plan (child) contains multiple parts of each parent whereas in single point crossover the child contains a part from each of the two parents chromosome. With the use of multi point crossover, a combination between multiple parts of the week plan is possible. Therefore, a higher crossover rate and the use of multi point crossover should improve the algorithm's performance since more combinations must be explored.

The two crossover rates were tested for both multi point and single point crossover. The overall results show slight differences between the tested operators values, however, there is no indication that the quality of the results is improved. Nonetheless, for the current problem and for both users, the best results were obtained with the multi point crossover with a probability of 80% as presented in Table 4.8.

Table 4.8: Results for user 6 and 9 with two crossover rates and types.

	CrossOver					
	Type	Rate	Mean	Max	Min	SD
User 6	Multipoint	0.6	83.90	84.40	82.83	0.43
		0.8	83.93	84.99	82.86	0.54
	Singlepoint	0.6	83.46	84.08	82.78	0.36
		0.8	83.67	84.34	83.24	0.31
User 9	Multipoint	0.6	85.16	85.63	84.49	0.32
		0.8	85.38	86.11	84.48	0.47
	Singlepoint	0.6	85.00	85.50	84.06	0.38
		0.8	85.12	85.91	84.09	0.60

Mutation Three other hyperparameters must be tested. The second one is the mutation rate. Once again, two values for the probability of mutation were tested. According to [63], the mutation rate for genetic algorithms is significantly smaller than the crossover rate.Two mutation rates - 0.05 and 0.1 - were tested in order to find the best variables combination, the results are shown in Table 4.9.

The mutation operator for this problem is used to find recipe substitutes. When the operator is applied to a gene/recipe, the candidate list for that meal will be used to randomly replace the gene. Without the mutation operator, the initial population recipes would be maintained with no alteration throughout the iterations. A higher mutation rate must, consequently, lead to a bigger search space since it might add new recipes to the population.

Table 4.9: Results for user 6 and 9 with two mutation rates.

	Mutation	Mean	Max	Min	SD
User 6	0.05	83.93	84.40	82.83	0.43
	0.10	84.10	84.54	83.37	0.36
User 9	0.05	85.38	86.11	84.48	0.50
	0.10	85.46	86.23	84.67	0.50

Analysing the results from Table 4.9, the conclusion is that the fitness value is the greatest when using a mutation rate of 10% for both users. However, the observed differences are too small, therefore, there is no evidence that the quality of results is altered with the mutation rate changes.

Selection For this work, two selection operators were tested: the roulette wheel and the truncation operator. For both the selection of the offspring and the survivors genotypes, the same selector was tested.

The Roulette Wheel Selector is implemented on the *Jenetics* as a probability selector. The probability of the individual x to be selected is calculated using their fitness value f according to

equation 4.1.

$$P(x) = \frac{f_x}{\sum_{j=0}^{N-1} f_j} \tag{4.1}$$

Using this selector for n individuals is equivalent to playing n times the roulette wheel.

As for the truncation selector, it has a very simple method where it selects the best n individuals from the population.

The obtained results for the two tested selectors are displayed in Table 4.10.

Table 4.10: Results for the roulette wheel and truncation selector.

	Selectors	Mean	Max	Min	SD
User 6	Truncation	84.10	84.54	83.37	0.36
	RouletteWheel	81.63	82.53	80.84	0.46
User 9	Truncation	85.38	86.11	84.48	0.50
	RouletteWheel	82.88	83.20	82.53	0.21

Since the truncation selector uses the best genotypes to keep in between generations, the overall results are superior. Another conclusion taken from the analysis of the results is that while for the Roulette Wheel selector the initial and final best fitness are not too distant, a clear increase in the fitness function throughout the generations is observed for the Truncation selector.

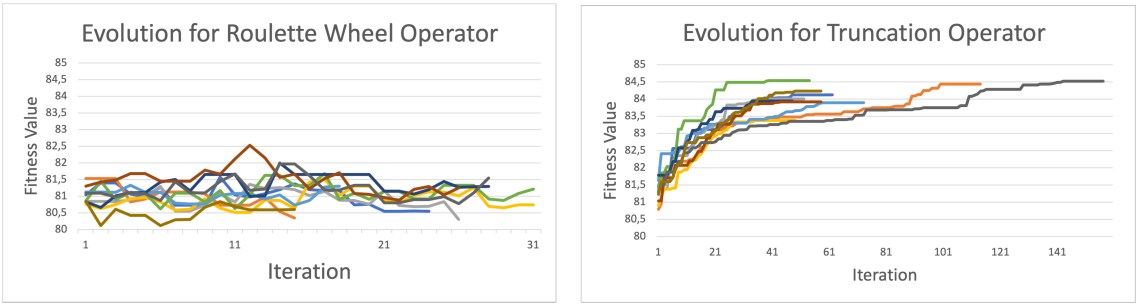

Figure 4.1: Fitness Evolution for the Roulette Wheel Operator (above) and the Truncation Operator (below).

Population Size The last hyperparameter to test is the population size. The goal of this test was to study how the fitness value would change with the population growth. The population growth is expected to improve the results of the algorithm as a higher number of genotypes should lead to a bigger search for week plans possibilities. More week plans should imply more tested combinations and recipes. Four sizes were used: 20, 50, 100 and 200. This growth leads to a higher computational cost therefore only if the improvement is considerable does the population size growth can be pondered.

Analysing the results it is clear that through the increase in the population size, an improvement in the majority of the heuristics is obtained. This improvement can be confirmed by looking at the day and week score. For a population size of 20 genotypes, multiple heuristics manage to either

Table 4.11: Heuristics values for each population for user 6.

	Genetic Algorithm							
	Population 20		Population 50		Population 100		Population 200	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
WeekScore	84,08	0,38	85,94	0,36	87,50	0,30	**88,74**	0,25
DayScore	9,87	0,23	10,13	0,22	10,36	0,21	**10,53**	0,16
CarbohydrateSuitabilityHeuristic	0,24	0,03	0,28	0,02	0,35	0,02	**0,38**	0,02
CheaperFoodHeuristic	0,68	0,03	0,76	0,03	0,81	0,02	**0,85**	0,01
DiabetesHeuristic	0,96	0,01	0,98	0,02	0,99	0,01	**1,00**	0,00
EnergySuitabilityHeuristic	0,20	0,02	0,26	0,02	0,33	0,02	**0,38**	0,02
FatSuitabilityHeuristic	0,36	0,03	0,39	0,03	0,38	0,03	**0,40**	0,01
FoodRestrictionsEnforcementHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
FruitVarietyDayHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	0,86	0,07
IncludeWhiteMeatHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
IngredientVarietyHeuristic	**0,67**	0,01	0,61	0,04	0,57	0,02	0,50	0,02
LimitMainDishRecipeRepetition	0,99	0,01	0,99	0,01	1,00	0,01	1,00	0,00
LitreOfSoupInARow	0,45	0,01	0,47	0,02	0,48	0,02	**0,51**	0,01
MeatFishSeparationHeuristic	0,52	0,01	0,54	0,02	0,54	0,01	0,54	0,01
MinimizeProductCategoriesHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
NoComfortFoodHeuristic	0,96	0,02	0,96	0,01	0,94	0,01	0,94	0,01
ProteinSuitabilityHeuristic	0,21	0,02	0,27	0,03	0,35	0,03	**0,40**	0,04
SameRecipeForMealDivisionConstraintHeuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
UserPreferencesHeuristic	0,62	0,01	0,63	0,01	**0,64**	0,01	**0,64**	0,01

Table 4.12: Heuristics values for each population for user 9.

	Genetic Algorithm							
	Population 20		Population 50		Population 100		Population 200	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
WeekScore	83,84	0,49	85,65	0,46	87,09	0,37	**88,44**	0,22
DayScore	9,83	0,21	10,09	0,16	10,30	0,17	**10,49**	0,16
CarbohydrateSuitabilityHeuristic	0,55	0,03	0,60	0,02	0,64	0,02	**0,69**	0,02
CheaperFoodHeuristic	0,66	0,03	0,68	0,03	0,74	0,03	**0,76**	0,03
DiabetesHeuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
EnergySuitabilityHeuristic	0,51	0,03	0,55	0,03	0,61	0,02	**0,66**	0,02
FatSuitabilityHeuristic	0,46	0,02	0,51	0,02	0,54	0,03	**0,58**	0,01
FoodRestrictionsEnforcementHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
FruitVarietyDayHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	0,99	0,05
IncludeWhiteMeatHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
IngredientVarietyHeuristic	**0,72**	0,03	0,73	0,03	0,68	0,02	0,65	0,03
LimitMainDishRecipeRepetition	1,00	0,01	1,00	0,00	1,00	0,00	1,00	0,00
LitreOfSoupInARow	0,45	0,01	0,45	0,01	0,47	0,01	**0,48**	0,02
MeatFishSeparationHeuristic	0,52	0,02	0,54	0,01	**0,55**	0,01	**0,55**	0,01
MinimizeProductCategoriesHeuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
NoComfortFoodHeuristic	0,99	0,00	1,00	0,00	1,00	0,00	1,00	0,00
ProteinSuitabilityHeuristic	0,44	0,04	0,50	0,04	0,55	0,04	**0,59**	0,02
SameRecipeForMealDivisionConstraintHeuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
UserPreferencesHeuristic	0,52	0,00	0,53	0,00	0,52	0,00	0,53	0,00

reach the maximum result or a result very close to the maximum. In these cases, the increase in the population size would not be necessary. This is the case for heuristics such as: diabetes, food restrictions enforcement, include white meat, limit main dish recipe and minimize product categories. Other heuristics - fruit variety day and ingredient variety - actually deteriorate with the population size change from 20 to 200. The increase in the population size, however, leads to a considerable improvement in the remaining heuristics. The heuristics related to the nutritional needs suffer the greatest upgrade.

Throughout the evaluation of the results it clear that the increase in the population size leads to better results however at the cost of a higher computational time. The improvement of some heuristics may have interfered with the result of others as was clear by the results from Table 4.14. The improvement in the nutritional values of the week plan led to a lower variety in the plan's fruits and used ingredients.

4.2.3 Fitness Function Changes

The heuristics used for the fitness calculation were designed in order to help the previously developed greedy algorithm reach an optimal solution. Each heuristic also had different weights. In order to test the effectiveness of both the genetic algorithm and the greedy algorithm, some changes were made. The goal of these modifications was to evaluate the approaches robustness to changes in both the calculation and the weights of each heuristic. Since the heuristics calculations had been previously developed according to the greedy approach needs, the hypothesis was that without the orientation in the calculation the greedy approach might under perform. The work can be divided in three steps:

- Equal weights and modification of the heuristics's implementation;
- Different weights' Heuristics;
- Fitness function adapted to the greedy algorithm and weights changes.

To study the genetic algorithm and the greedy approach performance for the three considered scenarios, user 6 and user 9 were tested for the four population sizes. Since genetic algorithms are stochastic, each experiment was repeated 10 times.

4.2.3.1 Equal weights and modification of heuristics implementation

As described in Section 3.5.3.1, the fitness calculation used for the previous work had heuristics with substantially different weights. The day heuristic had the same value as all the meal of day division heuristics combined while the week heuristic had an even higher weight.

For an unbalanced weight distribution, the heuristics with the highest weights will be favoured. These discrepancies might hamper the search for the genetic algorithm since differences in one meal of day division would be discarded when considering a change in the week's heuristic. With these weights distribution, the algorithm would favour a week plan that fulfill the include white

meal heuristic however with a poor overall performance in all other heuristics over a more balanced week plan that did not fulfill the include white meat heuristic.

In order to remove this gap, all heuristics values were set to the same weight. When all heuristics have the same weight the algorithm equally tries to maximise all heuristics.

Another change performed in the algorithm was the calculation of some of the heuristics. The performed changes on the heuristics are summarised next.

- Heuristics Calculation Changes:
  - Include White Meat Heuristic
    - **Greedy Based Calculus**:When the set number of white meat meals is not reached,the white meal recipes in the plan are scored higher to guide the algorithm
    - **Modified Calculus**: All recipes in the plan have the same score for the heuristic. If the plan has one white meat meal the score is set to 0.5. If it does not contain any white meat recipe is 0. The value is set to 1 if two meals exist in the plan.
  - Fruit Variety Day Heuristic
    - **Greedy Based Calculus**:Lower scores are given to non fruit dessert and to the most repeated fruits.
    - **Modified Calculus**: The number of different fruits for the 1.0 score is three. If the plan only has two the score is set to 0.6, if one to 0.3 and if none 0.
  - Food Restrictions Enforcement Heuristic
    - **Greedy Based Calculus**:Maximum score is set to 0.5.
    - **Modified Calculus**: Maximum score is set to 1.
  - Minimize Product Categories Heuristic
    - **Greedy Based Calculus**:Maximum score is set to 0.5.
    - **Modified Calculus**: Maximum score is set to 1.

Two of the four changed heuristics calculation suffered major changes: include white meat and fruit day variety. The include white meat heuristic calculation had been adapted to guide the algorithm. In order to do so, when the heuristic condition was not fulfilled, the meals with white meat score was higher than the meals with fish or other meats. This way, the algorithm would know what recipes to substitute. For the modification, for the maximum and minimum score no alteration was done however, when only one meal had white meat (the goal is two meals) a score of 0.5 to all meals was set.

Regarding the fruit variety heuristic a similar calculation had been developed. For this heuristic, the goal was to have three types of fruit per day. When the condition was not ensured the meal divisions for fruit that had desserts or the most common fruit type in the plan had a significantly lower score than other fruit meals. The modification somewhat linearized the results. For one fruit type a score of 0.3 was set and for two 0.6. These modifications hinder the greedy algorithm

from using clues to orient its search. Both these changes had the goal of evaluating whether the greedy approach would still succeed in the search for an optimal solution or if the performance would deteriorate. And at the same time evaluate the genetic algorithm performance and compare its robustness to the previous approach.

As for the other two heuristics: minimize product categories and food restrictions enforcement the maximum score was changed from 0.5 to 1 in order to ensure that all heuristic had the same weight in the final result.

The obtained results for the modified fitness function are summarized in Table 4.13. There is a clear increase in the fitness function from the initial best fitness to the final result for all population sizes. As the population size increases so does the fitness function, however, it also leads to a higher computational time. The differences between the two users in the overall value are not significant. In Table 4.14, a detailed result of the heuristics values is presented.

Table 4.13: Summary results for fitness function with equal weights.

	Genetic Algorithm						Greedy Algorithm		
	Population	Final Fitness		Initial Fitness	Time(s)	Generations			
		Mean	SD	Mean			Mean	SD	Time
User 6	20	84,08	0,38	81,18	20	77	88,16	0,22	4
	50	85,94	0,36	81,51	58	100			
	100	87,50	0,30	81,53	154	141			
	200	**88,74**	0,25	81,63	362	158			
User 9	20	83,84	0,49	81,26	22	82	**88,94**	0,12	6
	50	85,65	0,46	81,38	69	109			
	100	87,09	0,37	81,48	179	146			
	200	88,44	0,22	81,53	372	160			

In order to detailedly study the fitness function, results for both algorithms are illustrated in Table 4.14. The detailed results for user 9 are in Section A.3. In order to evaluate the population size influence in each heuristic, the information regarding the results for a population size of 20 and 200 is shown.

Regarding the comparison with the greedy algorithm results, for a majority of the heuristics its values are very similar to the population of 200 genotypes. Most heuristics related with the nutritional needs for the user are better for the greedy algorithm. However, the two heuristics whose calculation was linearized compared with the previously developed algorithm function – fruit variety day and include white meat – performed substantially better for the genetic algorithm since the greedy algorithm results for these two heuristics are either close or equal to 0.

4.2.3.2 Different weights' Heuristics

For this study, the modified heuristics calculation was considered however, each heuristic had considerable different weights, as described in Section 3.5.3.1. This way, a study on how the algorithm adapts to different weights with the modified calculations can be done. The goal is to understand how the solutions quality is affected. When different weights are set, it is clear that

Table 4.14: Detailed heuristics results for the two algorithms (user 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 20		Population 200			
	Mean	SD	Mean	SD	Mean	SD
Week Score	84,08	0,38	**88,74**	0,25	88,16	0,22
Day Score	9,87	0,23	**10,53**	0,16	10,45	0,19
Carbohydrate Suitability Heuristic	0,24	0,03	0,38	0,02	**0,46**	0,05
Cheaper Food Heuristic	0,68	0,03	0,85	0,01	**0,90**	0,01
Diabetes Heuristic	0,96	0,01	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,20	0,02	0,38	0,02	**0,47**	0,03
Fat Suitability Heuristic	0,36	0,03	0,40	0,01	**0,45**	0,03
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	**1,00**	0,00	**0,86**	0,07	0,01	0,03
Include White Meat Heuristic	**1,00**	0,00	**1,00**	0,00	0,00	0,00
Ingredient Variety Heuristic	**0,67**	0,01	**0,50**	0,02	0,19	0,01
Limit Main Dish Recipe Repetition	**0,99**	0,01	**1,00**	0,00	0,88	0,01
Litre Of Soup In A Row	0,45	0,01	**0,51**	0,01	0,49	0,02
Meat Fish Separation Heuristic	0,52	0,01	0,54	0,01	0,55	0,01
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	**0,96**	0,02	0,94	0,01	0,94	0,02
Protein Suitability Heuristic	0,21	0,02	0,40	0,04	0,41	0,05
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,62	0,01	0,64	0,01	**0,68**	0,01

some heuristics will be favor and a higher effort in finding optimal scores for those same heuristics will be done. However, setting different weights may lead to an under performance of heuristics with the lowest weights which may compromise the final result.

Regarding the detailed results of both the genetic and greedy algorithm, for user 9 the genetic performs slightly worst for all population sizes. For the 200 population the results are similar. However, user 6 genetic algorithm's results are significantly higher compared with the greedy algorithm. Analysing the detailed heuristics, illustrated in Table 4.16, a conclusion can be taken. For the greedy algorithm the final plans do not fulfill the conditions for the include white meat heuristic, therefore, the significant drop in the value is explained.

Table 4.15: Summary results for fitness function with different weights.

	Genetic Algorithm						Greedy Algorithm		
	Population	Final Fitness		Initial Fitness	Time	Generations			
		Mean	SD	Mean			Mean	SD	Time
User 6	20	288,59	0,25	285,90	18	72	219,97	24,76	5
	50	289,40	0,11	286,29	100	176			
	100	289,84	0,08	286,34	213	193			
	200	**290,11**	0,08	286,61	403	186			
User 9	20	256,48	0,13	255,37	29	103	**258,03**	0,04	8
	50	256,81	0,11	255,17	81	132			
	100	257,47	0,07	255,55	217	183			
	200	257,82	0,06	255,55	424	180			

Table 4.16: Detailed heuristics results for the two algorithms (user 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 20		Population 200			
	Mean	SD	Mean	SD	Mean	SD
Week Score	288,59	0,25	**290,11**	0,08	219,97	24,76
Day Score	26,23	0,08	**26,44**	0,05	26,17	1,80
Carbohydrate Suitability Heuristic	0,20	0,03	0,37	0,02	**0,39**	0,03
Cheaper Food Heuristic	0,65	0,05	**0,80**	0,02	0,69	0,03
Diabetes Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,15	0,03	0,32	0,03	**0,34**	0,02
Fat Suitability Heuristic	0,35	0,02	0,38	0,02	0,39	0,02
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	1,00	0,00	0,99	0,05	0,99	0,05
Include White Meat Heuristic	**1,00**	0,00	**1,00**	0,00	0,50	0,24
Ingredient Variety Heuristic	**0,65**	0,03	0,56	0,02	0,42	0,02
Limit Main Dish Recipe Repetition	1,00	0,01	1,00	0,00	1,00	0,00
Litre Of Soup In A Row	0,44	0,01	0,49	0,02	0,50	0,01
Meat Fish Separation Heuristic	0,50	0,02	0,53	0,01	0,54	0,00
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	0,99	0,01	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,19	0,03	**0,36**	0,04	0,31	0,01
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,61	0,01	0,65	0,01	**0,67**	0,01

For this study, the fruit variety day heuristic did not under performed with the increase in the population size. Since, in this study, the weight of this heuristic for the fitness calculation is significantly superior, the algorithm is capable of finding solutions that simultaneously reach closer values related with the user nutritional needs while maintaining the fruit diversity. The same does not occur for the ingredient variety. This heuristic suffers a decay with the population growth.

When comparing the results for the genetic and the greedy algorithm, the clear difference that explains the overall week score gap is the include white meat heuristic. Although the algorithm is capable of reaching a better score in this heuristic compared with the result in Table 4.14, the mean value is 0.5 while for the genetic algorithm is 1. The ingredient variety and the cheaper food heuristics also performs notably better in the genetic algorithm. All the others heuristics have slight variations between the two algorithms.

4.2.3.3 Fitness function adapted to the greedy algorithm and weights changes

In the final scenario, the conditions used for the implementation of the greedy algorithm were used to test how the genetic algorithm would perform compared with the previously developed algorithm. For this scenario, the optimal conditions for the greedy approach are used to compare the results with the genetic algorithm.

After modifying the heuristics weights, the next study was how the change in the heuristics calculation would affect the overall result. These results are obtained using the initial heuristics conditions developed in the previous work.

Table 4.17: Summary results for adapted fitness function with different weights.

	Genetic Algorithm					Greedy Algorithm		
	Population	Final Fitness		Initial Fitness	Time	Generations			
		Mean	SD	Mean			Mean	SD	Time
User 6	20	225,84	0,40	220,09	21	84	227,34	0,06	7
	50	226,80	0,13	221,23	87	152			
	100	227,29	0,10	223,54	198	179			
	200	**227,60**	0,08	223,84	417	192			
User 9	20	194,18	0,15	193,07	27	100	**195,86**	0,05	9
	50	194,74	0,07	193,11	84	134			
	100	195,17	0,10	193,17	207	168			
	200	195,52	0,05	193,20	448	182			

As expected, the increase in the population size leads to better results. Similar to the first scenario, the results for the best fitness are similar, however, the greedy algorithm performs overall better and with a lower computational time.

Table 4.18: Details heuristics results for the two algorithms (user 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 20		Population 200			
	Mean	SD	Mean	SD	Mean	SD
Week Score	225,84	0,40	**227,60**	0,08	227,35	0,06
Day Score	17,26	0,12	**17,51**	0,07	17,48	0,10
Carbohydrate Suitability Heuristic	0,18	0,02	0,35	0,03	**0,37**	0,03
Cheaper Food Heuristic	0,59	0,04	**0,79**	0,01	0,70	0,02
Diabetes Heuristic	1,00	0,01	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,13	0,01	0,32	0,03	**0,36**	0,01
Fat Suitability Heuristic	0,34	0,02	0,38	0,02	0,39	0,01
Food Restrictions Enforcement Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
Fruit Variety Day Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
Include White Meat Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Ingredient Variety Heuristic	**0,55**	0,01	0,45	0,03	0,34	0,03
Limit Main Dish Recipe Repetition	0,98	0,01	0,98	0,00	0,98	0,00
Litre Of Soup In A Row	0,45	0,00	0,48	0,01	**0,50**	0,01
Meat Fish Separation Heuristic	0,50	0,01	0,54	0,01	0,55	0,01
Minimize Product Categories Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
No Comfort Food Heuristic	0,99	0,01	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,18	0,03	**0,36**	0,02	0,31	0,03
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,61	0,01	0,64	0,01	**0,66**	0,02

As for the comparison with the greedy algorithm, the include white meat and the fruit variety day heuristics do not have a significant change. Since the heuristics calculation used were developed in order to somewhat guide the algorithm, this result is expectable.

4.2.3.4 Random Search

For all three scenarios and in order to confirm the genetic algorithm performed better than a random search, a population of 10000 random genotypes was created and their fitness value stored. The goal is to confirm that the genetic algorithm search leads to better results compared with a random search where no data processing is performed. After analysing the best set of operators for the performance of the genetic algorithm, the three scenarios were tested for the four chosen population sizes.

In Figure 4.2 a representation of the fitness values obtained for the ten thousand random genotypes is illustrated. The distribution resembles a normal distribution. As expected, the genetic algorithm results significantly outperform the best fitness values obtained by the random search.

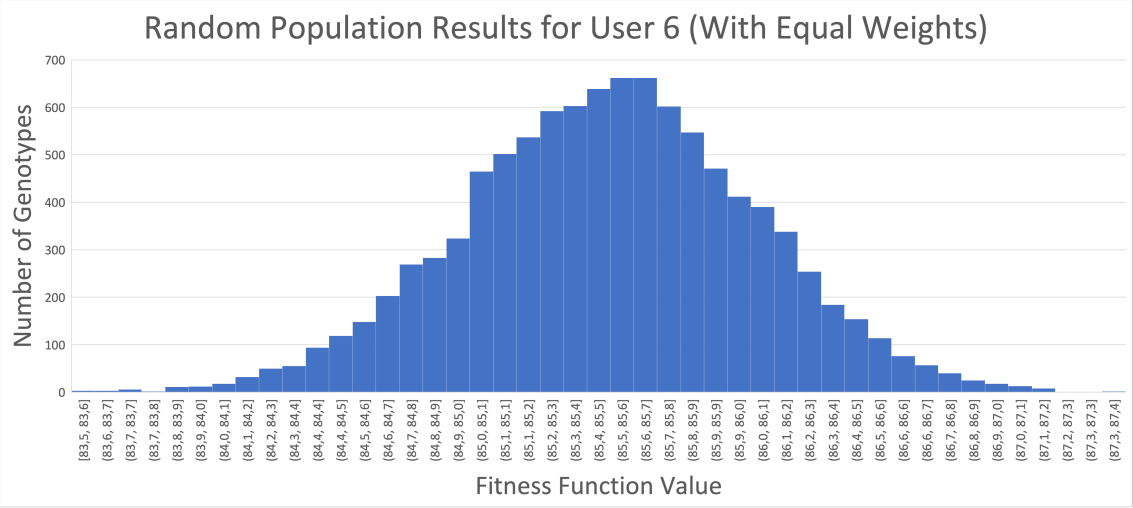

Figure 4.2: Histogram of random population for user 6 with equal heuristics weights and modified calculations.

In Figure 4.3 a representation of the fitness values obtained for the 10000 random genotypes are illustrated. The discontinuous distribution is explained by the uneven weight distribution of the heuristics. The distribution results for user 9 are illustrated in Figure 4.4.

A conclusion taken from the comparison between Figure 4.3 and Figure 4.4 is that for an unrestricted user, such as user 9, the majority of the solutions fitness is located in the right corner, which means that most plans have what can be considered a good solution. As for the diabetic user, the opposite occurs. In this case we have a clear domination of lower fitness values. This result can explain why the greedy algorithm had a significantly lower result. Since a majority of the plans for user 6 have lower fitness values, the greedy algorithm can not find a solution around or above the results for the right corner of the distribution.

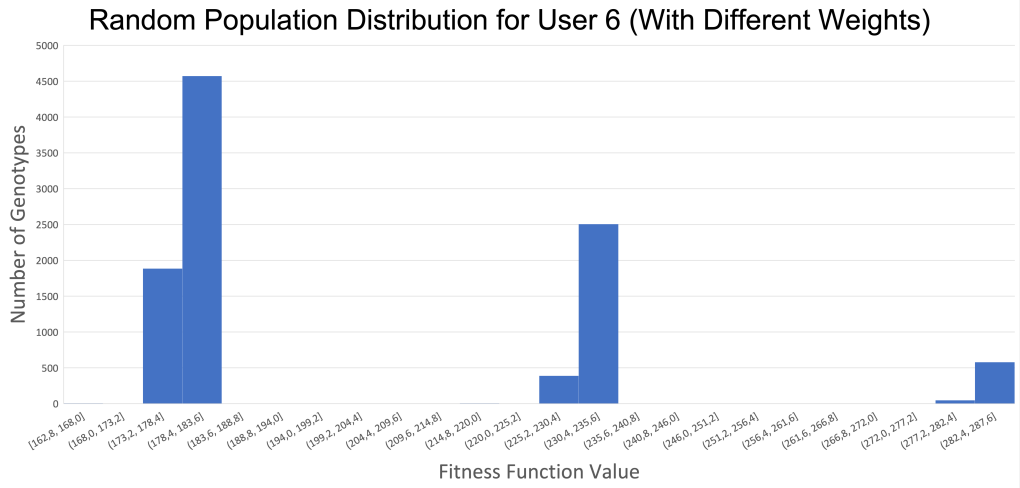

Figure 4.3: Histogram of random population for user 6 with modified heuristics calculation and different weights.

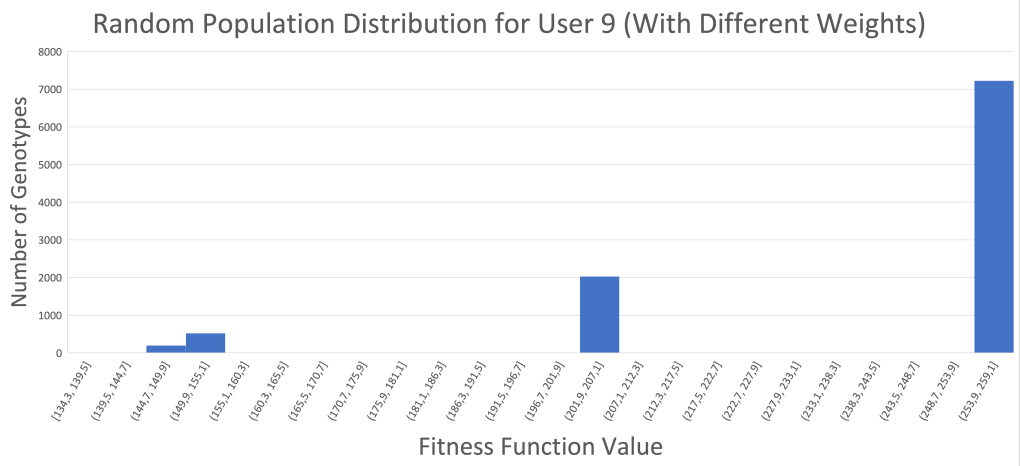

Figure 4.4: Histogram of random population for user 9 with modified heuristics calculation and different weights.

4.2.4 Results conclusions

To summarise, the results indicate that the genetic algorithm is less sensitive to changes in the calculation of the heuristics scores. Nevertheless, the developed studies are still pretty sparse in order to be able to be sure of this possibility. New tests have to be done considering different users and different fitness calculations, however, for this work, it was not possible to do so.

In general the genetic algorithm had a similar performance to the greedy algorithm. Nonetheless, the genetic algorithm has an higher computational time. The results for user 9 are detailed in the appendix section A.3.

4.3 Approaches connection

A recommender system and a genetic algorithm were developed through the course of this work. The connection between the two frameworks was done, however, it was not tested since the recommender system does not contains all the database users. This obstacle prohibited the integration of the recommender system heuristic into the genetic algorithm results.

In order to correctly evaluate the week plan with the recommender heuristic, the plan before and after the heuristic has to be rated by the user in order to check its validity. Therefore, this is a clear limitation of the developed work that must be fixed in the future.

Chapter 5

Conclusion and Future Work

With the increase in number of older population and the growing need to lead a healthy lifestyle, an aid for a correct and balanced nutrition becomes a priority. A meal plan adapted not only to the users needs but also to their preferences is essential. In order to adapt a plan to a specific user, their general information such as weight, height, age and diseases must be taken into account. But, in order to please the user, its preferences must also be studied and favoured. The number of people leading restrictive lifestyles such as plant based and gluten free diets is increasing, therefore, this information is also essential to provide a good diet plan.

This scenario can be regarded as a recommendation problem with multiple constraints. Recommender systems have been widely used for multiple scenarios such as travel, clothes and online shopping. These systems are a valuable tool to predict new offers using the person's information and previous likes. The use of recommender systems together with optimization techniques has been explored in the past few years. This combination allows for set recommendations.

The nutritional scenario can use this combination in order to obtain meal plans to aid users into healthier diets. The goal of this project was to study and implement an approach that combines an off-the-self recommender system with an optimization technique for weekly meal recommendations. This approach must be capable of considering all the previously mentioned conditions - user's nutritional necessities, restrictions, preferences and diseases - and with them create a personalized weekly meal plan. The developed approach must be evaluated and compared with a previously developed approach using real world data.

This project focuses on the development of a recommendation system to model the user's preferences followed by the use of an optimization technique to create the weekly meal plan. The developed work was compared with a previously created greedy algorithm.

These two systems can be coupled in order to provide simultaneously a set of items that maximises certain conditions and at the same time considers the items the user will most likely approve.

The first experiment consisted in the creation of an off-the-shelf recommender system to then be coupled to the optimization framework. Two techniques were tested: collaborative filtering and content-based. A hybrid model using the two was also created.

The next step was to develop a framework to create the plan. Genetic Algorithms are exploratory procedures that aid the search for optimal solutions to complex problems. They are then, used as optimization techniques.Therefore, a genetic algorithm was designed and studied to adapt to the current nutrition scenario.

In order to use it, several conditions must be studied in order to obtain the best set of implementations for the algorithm. Multiple population sizes, crossover techniques and rates, mutation rates and selectors were detailedly studied. The final conditions were added to the algorithm and the results were obtained.

For this work only two users were tested for this algorithm in order to evaluate how restrictions would influence the overall results. A restricted user shrinks the number of recipes to use for the week plan which, as expected, hampered the final results.

For the evaluation of the results, modifications in the fitness function were made in order to evaluate the robustness of the two approaches. The used fitness function was based on heuristics developed in previous work, however, some modifications were tested in order to evaluate both the genetic and the greedy algorithm response. Both the modification in the heuristics calculation and the heuristics weights led to some alterations in the overall results.

The limited number of evaluated users hinders the generalization of some conclusions. All the tested hypothesis must furthered evaluated for a considerable number of users in order to corroborate or reject the premises.

The recommendation system was developed using *Python* while the genetic algorithm used the *Java* environment since all previous work was done in *Java*. The connection between the two environments was done.

From the recommender system implementation, the hybrid model showed the best results. However, since the database had a small percentage of users with a significant number of rated recipes and an off-the-shelf system was used, its performance was not outstanding.

Regarding the genetic algorithm, its performance was, in a majority of the tested scenarios, similar to the greedy algorithm, however, it is a more time consuming framework. For the restricted user (diabetic and several ingredient restrictions) with alterations in the fitness function the greedy algorithm had a significantly lower performance compared with the genetic algorithm. For this scenario, the genetic algorithm showed a higher robustness to changes. Considering the different tested scenarios and the modifications in the fitness function the genetic algorithm manage to find overall best solutions compared with the greedy approach. This shows a limitation of the greedy approach that is solved with the use of the genetic algorithm.

Regarding future work, the complete connection between the two environments must be developed and evaluated using real world users to evaluate the recommender system recommendations and the week plan. In order to confirm the robustness hypothesis, more users should be tested in order to evaluate both approaches performance for the different scenarios.

Appendix A

Appendix

A.1 Histogram results

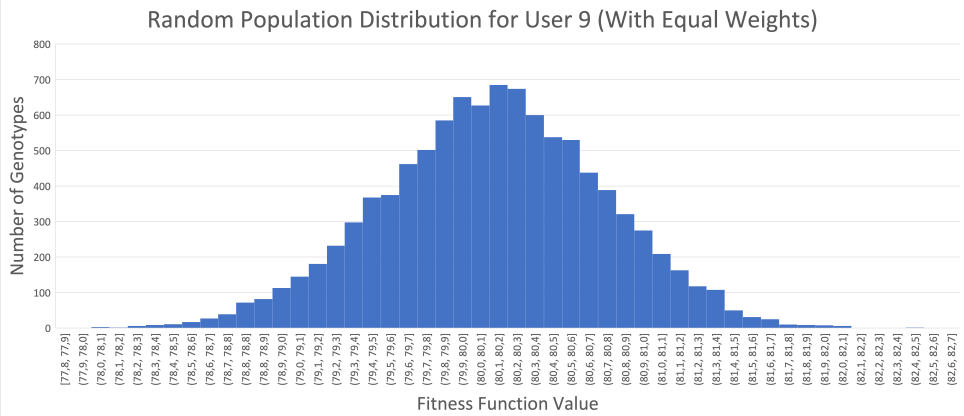

Figure A.1: Histogram of random population for user 9 with equal heuristics weights and modified calculations.

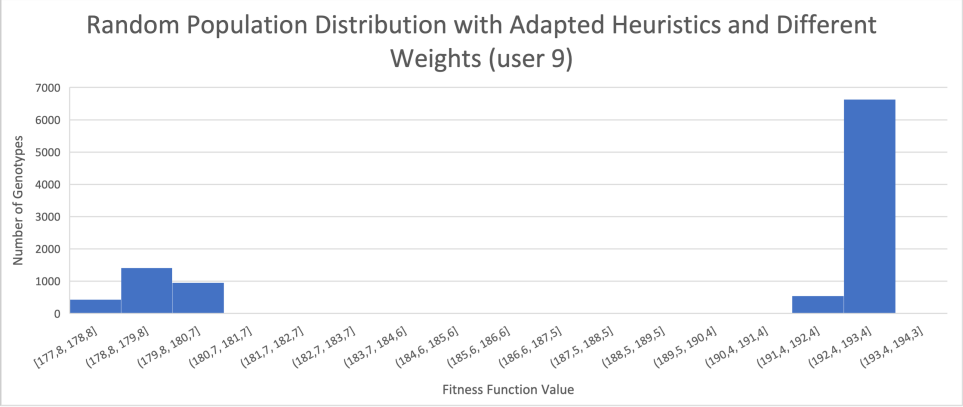

Figure A.2: Histogram of random population for user 9 adapted heuristics calculation and different weights.

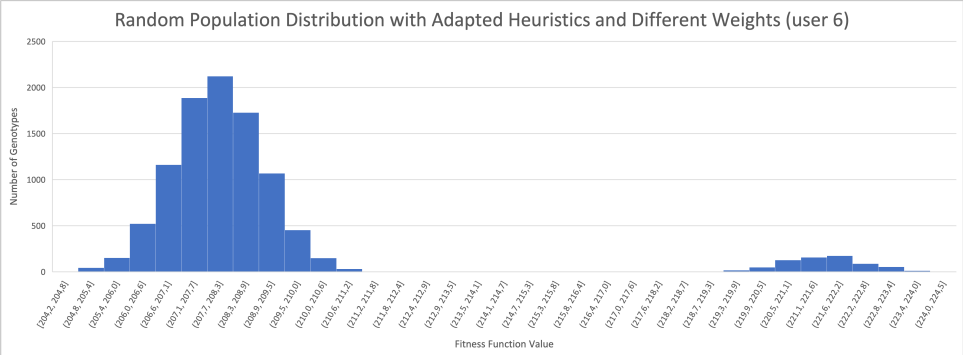

Figure A.3: Histogram of random population for user 6 adapted heuristics calculation and different weights.

A.2 User 6 Detailed Results

Table A.1: Detailed heuristics results for the two algorithms with equal weights heuristics and modified calculation (User 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 50		Population 100			
	Mean	SD	Mean	SD	Mean	SD
Week Score	85,94	0,36	87,50	0,30	88,16	0,22
Day Score	10,13	0,22	10,36	0,21	10,45	0,19
Carbohydrate Suitability Heuristic	0,28	0,02	0,35	0,02	0,46	0,05
Cheaper Food Heuristic	0,76	0,03	0,81	0,02	0,90	0,01
Diabetes Heuristic	0,98	0,02	0,99	0,01	1,00	0,00
Energy Suitability Heuristic	0,26	0,02	0,33	0,02	0,47	0,03
Fat Suitability Heuristic	0,39	0,03	0,38	0,03	0,45	0,03
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	1,00	0,00	1,00	0,00	0,01	0,03
Include White Meat Heuristic	1,00	0,00	1,00	0,00	0,00	0,00
Ingredient Variety Heuristic	0,61	0,04	0,57	0,02	0,19	0,01
Limit Main Dish Recipe Repetition	0,99	0,01	1,00	0,01	0,88	0,01
Litre Of Soup In A Row	0,47	0,02	0,48	0,02	0,49	0,02
Meat Fish Separation Heuristic	0,54	0,02	0,54	0,01	0,55	0,01
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	0,96	0,01	0,94	0,01	0,94	0,02
Protein Suitability Heuristic	0,27	0,03	0,35	0,03	0,41	0,05
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,63	0,01	0,64	0,01	0,68	0,01

A.3 User 9 Detailed Results

Table A.2: Heuristics scores for each algorithm and each user for adapted heuristic calculation and different weights.

	Population Size 20		Population Size 200		Greedy Algorithm	
	User 6	User 9	User 6	User 9	User 6	User 9
Carbohydrate Suitability Heuristic	0,18	**0,57**	0,35	**0,71**	0,37	**0,89**
Cheaper Food Heuristic	0,59	**0,64**	**0,79**	0,75	**0,70**	0,57
Diabetes Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,13	**0,52**	0,32	**0,69**	0,36	**0,74**
Fat Suitability Heuristic	0,34	**0,47**	0,38	**0,62**	0,39	**0,74**
Food Restrictions Enforcement Heuristic	0,50	0,50	0,50	0,50	0,50	0,50
FruitVariety Day Heuristic	0,50	0,50	0,50	0,50	0,50	0,50
Include White Meat Heuristic	1,00	1,00	1,00	1,00	1,00	1,00
Ingredient Variety Heuristic	0,55	**0,63**	0,45	**0,52**	0,34	**0,41**
Limit Main Dish Recipe Repetition	0,98	**1,00**	0,98	**1,00**	0,98	**1,00**
Litre Of Soup In A Row	0,45	0,45	0,48	0,50	0,50	0,49
Meat Fish Separation Heuristic	0,50	**0,53**	0,54	**0,55**	0,55	**0,57**
Minimize Product Categories Heuristic	0,50	0,50	0,50	0,50	0,50	0,50
No Comfort Food Heuristic	0,99	1,00	1,00	1,00	1,00	1,00
Protein Suitability Heuristic	0,18	**0,47**	0,36	**0,62**	0,31	**0,69**
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	**0,61**	0,52	**0,64**	0,53	**0,66**	0,53

Table A.3: Detailed heuristics results for the two algorithms with different Weights heuristics and modified calculation (user 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 50		Population 100			
	Mean	SD	Mean	SD	Mean	SD
Week Score	289,40	0,11	289,84	0,08	219,97	24,76
Day Score	26,34	0,06	26,41	0,05	26,17	1,80
Carbohydrate Suitability Heuristic	0,28	0,02	0,32	0,02	0,39	0,03
Cheaper Food Heuristic	0,69	0,02	0,76	0,03	0,69	0,03
Diabetes Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,22	0,03	0,29	0,02	0,34	0,02
Fat Suitability Heuristic	0,37	0,02	0,39	0,02	0,39	0,02
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	1,00	0,00	1,00	0,00	0,99	0,05
Include White Meat Heuristic	1,00	0,00	1,00	0,00	0,50	0,24
Ingredient Variety Heuristic	0,66	0,03	0,60	0,02	0,42	0,02
Limit Main Dish Recipe Repetition	1,00	0,00	1,00	0,00	1,00	0,00
Litre Of Soup In A Row	0,45	0,01	0,47	0,02	0,50	0,01
Meat Fish Separation Heuristic	0,53	0,01	0,54	0,01	0,54	0,00
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,25	0,03	0,32	0,03	0,31	0,01
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,63	0,01	0,64	0,01	0,67	0,01

Table A.4: Detailed heuristics results for the two algorithms with different weights heuristics and adapted calculation (user 6).

	Genetic Algorithm				Greedy Algorithm	
	Population 50		Population 100			
	Mean	SD	Mean	SD	Mean	SD
Week Score	226,80	0,13	227,29	0,10	227,35	0,06
Day Score	17,40	0,09	17,47	0,08	17,48	0,10
Carbohydrate Suitability Heuristic	0,25	0,02	0,32	0,02	0,37	0,03
Cheaper Food Heuristic	0,69	0,02	0,75	0,04	0,70	0,02
Diabetes Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Energy Suitability Heuristic	0,20	0,02	0,28	0,03	0,36	0,01
Fat Suitability Heuristic	0,38	0,02	0,37	0,02	0,39	0,01
Food Restrictions Enforcement Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
Fruit Variety Day Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
Include White Meat Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Ingredient Variety Heuristic	0,58	0,04	0,50	0,03	0,34	0,03
Limit Main Dish Recipe Repetition	0,98	0,00	0,98	0,00	0,98	0,00
Litre Of Soup In A Row	0,46	0,02	0,47	0,01	0,50	0,01
Meat Fish Separation Heuristic	0,52	0,02	0,53	0,01	0,55	0,01
Minimize Product Categories Heuristic	0,50	0,00	0,50	0,00	0,50	0,00
No Comfort Food Heuristic	1,00	0,00	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,22	0,03	0,30	0,03	0,31	0,03
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,61	0,01	0,63	0,01	0,66	0,02

Table A.5: Detailed heuristics results for the two algorithms with equal weights heuristics and modified calculation (user 9).

	Genetic Algorithm								Greedy Algorithm	
	Population 20		Population 50		Population 100		Population 200			
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Week Score	83,84	0,49	85,65	0,46	87,09	0,37	88,44	0,22	88,95	0,12
Day Score	9,83	0,21	10,09	0,16	10,30	0,17	10,49	0,16	10,56	0,24
Carbohydrate Suitability Heuristic	0,55	0,03	0,60	0,02	0,64	0,02	0,69	0,02	0,90	0,02
Cheaper Food Heuristic	0,66	0,03	0,68	0,03	0,74	0,03	0,76	0,03	0,65	0,02
Diabetes Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Energy Suitability Heuristic	0,51	0,03	0,55	0,03	0,61	0,02	0,66	0,02	0,73	0,02
Fat Suitability Heuristic	0,46	0,02	0,51	0,02	0,54	0,03	0,58	0,01	0,72	0,01
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	0,99	0,05	0,99	0,05
Include White Meat Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	0,75	0,35
Ingredient Variety Heuristic	0,72	0,03	0,73	0,03	0,68	0,02	0,65	0,03	0,34	0,02
Limit Main Dish Recipe Repetition	1,00	0,01	1,00	0,00	1,00	0,00	1,00	0,00	0,92	0,01
Litre Of Soup In A Row	0,45	0,01	0,45	0,01	0,47	0,01	0,48	0,02	0,52	0,01
Meat Fish Separation Heuristic	0,52	0,02	0,54	0,01	0,55	0,01	0,55	0,01	0,57	0,00
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	0,99	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,44	0,04	0,50	0,04	0,55	0,04	0,59	0,02	0,68	0,02
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,52	0,00	0,53	0,00	0,52	0,00	0,53	0,00	0,53	0,00

Table A.6: Detailed heuristics results for the two algorithms with different weights heuristics and modified calculation(user 9).

	Genetic Algorithm								Greedy Algorithm	
	Population 20		Population 50		Population 100		Population 200			
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Week Score	256,48	0,13	256,81	0,11	257,47	0,07	257,82	0,06	258,03	0,04
Day Score	21,64	0,05	21,69	0,06	21,78	0,06	21,83	0,05	21,86	0,07
Carbohydrate Suitability Heuristic	0,56	0,04	0,62	0,03	0,67	0,03	0,71	0,03	0,88	0,01
Cheaper Food Heuristic	0,62	0,03	0,70	0,02	0,70	0,03	0,75	0,01	0,52	0,02
Diabetes Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Energy Suitability Heuristic	0,52	0,02	0,59	0,03	0,65	0,03	0,70	0,02	0,77	0,02
Fat Suitability Heuristic	0,48	0,02	0,52	0,03	0,57	0,02	0,61	0,02	0,73	0,01
Food Restrictions Enforcement Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Fruit Variety Day Heuristic	1,00	0,00	0,50	0,00	1,00	0,00	0,96	0,07	1,00	0,00
Include White Meat Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Ingredient Variety Heuristic	0,76	0,03	0,62	0,03	0,71	0,03	0,64	0,02	0,47	0,03
Limit Main Dish Recipe Repetition	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Litre Of Soup In A Row	0,45	0,01	0,46	0,01	0,47	0,02	0,50	0,02	0,51	0,02
Meat Fish Separation Heuristic	0,53	0,02	0,54	0,01	0,55	0,01	0,55	0,01	0,57	0,00
Minimize Product Categories Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
No Comfort Food Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,48	0,03	0,55	0,05	0,58	0,02	0,62	0,03	0,70	0,01
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,53	0,00	0,52	0,00	0,53	0,00	0,53	0,00	0,53	0,00

Table A.7: Detailed heuristics results for the two algorithms with different weights heuristics and adapted calculation (user 9).

	Genetic Algorithm								Greedy Algorithm	
	Population 20		Population 50		Population 100		Population 200			
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Week Score	194,18	0,15	194,74	0,07	195,17	0,10	195,52	0,05	195,85	0,05
Day Score	12,74	0,07	12,82	0,06	12,88	0,07	12,93	0,06	12,98	0,08
Carbohydrate Suitability Heuristic	0,57	0,02	0,63	0,01	0,67	0,02	0,71	0,02	0,89	0,01
Cheaper Food Heuristic	0,64	0,03	0,66	0,03	0,72	0,03	0,75	0,02	0,57	0,01
Diabetes Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Energy Suitability Heuristic	0,52	0,02	0,58	0,02	0,64	0,02	0,69	0,03	0,74	0,02
Fat Suitability Heuristic	0,47	0,02	0,51	0,04	0,58	0,03	0,62	0,02	0,74	0,01
Food Restrictions Enforcement Heuristic	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00
Fruit Variety Day Heuristic	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00
Include White Meat Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Ingredient Variety Heuristic	0,63	0,03	0,63	0,02	0,59	0,03	0,52	0,03	0,41	0,02
Limit Main Dish Recipe Repetition	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Litre Of Soup In A Row	0,45	0,01	0,46	0,01	0,47	0,02	0,50	0,01	0,49	0,01
Meat Fish Separation Heuristic	0,53	0,01	0,54	0,01	0,55	0,02	0,55	0,01	0,57	0,00
Minimize Product Categories Heuristic	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00
No Comfort Food Heuristic	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00
Protein Suitability Heuristic	0,47	0,03	0,54	0,03	0,58	0,03	0,62	0,02	0,69	0,01
Same Recipe For Meal Division Constraint Heuristic	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
User Preferences Heuristic	0,52	0,00	0,52	0,00	0,53	0,00	0,53	0,00	0,53	0,00

References

[1] WHO. Fact Sheets: Malnutrition, 2020. URL: https://www.who.int/news-room/fact-sheets/detail/malnutrition.

[2] Malnutrition: why is it affecting the elderly | The Nutrition Society. URL: https://www.nutritionsociety.org/events/malnutrition-why-it-affecting-elderly.

[3] Ageing and health, 2018. URL: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health.

[4] Clare A. Corish and Laura A. Bardon. Malnutrition in older adults: Screening and determinants. In *Proceedings of the Nutrition Society*, volume 78, pages 372–379. Cambridge University Press, aug 2019. URL: https://pubmed.ncbi.nlm.nih.gov/30501651/, doi:10.1017/S0029665118002628.

[5] Nadja Leipold, Mira Madenach, Hanna Schäfer, Martin Lurz, Nađa Terzimehić, Georg Groh, Markus Böhm, Kurt Gedrich, and Helmut Krcmar. Nutrilize a Personalized Nutrition Recommender System: an enable study. Technical report, 2018. URL: www.kochwiki.org.

[6] S. Abhari, R. Safdari, L. Azadbakht, K. B. Lankarani, Sh R. Niakan Kalhori, B. Honarvar, Kh Abhari, S. M. Ayyoubza-Deh, Z. Karbasi, S. Zakerabasali, and Y. Jalilpiran. A systematic review of nutrition recommendation systems: With focus on technical aspects. *Journal of Biomedical Physics and Engineering*, 9(6):591–602, dec 2019. URL: /pmc/articles/PMC6943843/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6943843/, doi:10.31661/jbpe.v0i0.1248.

[7] David Ribeiro, Jorge Ribeiro, Maria João M. Vasconcelos, Elsa F. Vieira, and Ana Correia de Barros. SousChef: Improved meal recommender system for Portuguese older adults. In *Communications in Computer and Information Science*, volume 869, pages 107–126. Springer Verlag, apr 2018. URL: https://link.springer.com/chapter/10.1007/978-3-319-93644-4{_}6, doi:10.1007/978-3-319-93644-4_6.

[8] Thi Ngoc Trang Tran, Müslüm Atas, Alexander Felfernig, and Martin Stettinger. An overview of recommender systems in the healthy food domain. *Journal of Intelligent Information Systems*, 50(3):501–526, 2018. URL: https://doi.org/10.1007/s10844-017-0469-0, doi:10.1007/s10844-017-0469-0.

[9] S.N. van Schaik, J. Masthoff, and A.T. Wibowo. Package recommender systems: A systematic review. *Intelligent Decision Technologies*, 13(4):435–452, feb 2020. doi:10.3233/idt-190140.

[10] Kinjal Chaudhari and Ankit Thakkar. A Comprehensive Survey on Travel Recommender Systems. *Archives of Computational Methods in Engineering*, 27(5):1545–1571, 2020. URL: https://doi.org/10.1007/s11831-019-09363-7, doi:10.1007/s11831-019-09363-7.

[11] Charu C. Aggarwal. *Recommender Systems*. Springer International Publishing, Cham, 2016. URL: http://link.springer.com/10.1007/978-3-319-29659-3, doi:10.1007/978-3-319-29659-3.

[12] Upendra Shardanand and Pattie Maes. Social information filtering. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, pages 210–217, New York, New York, USA, 1995. Association for Computing Machinery (ACM). URL: http://portal.acm.org/citation.cfm?doid=223904.223931, doi:10.1145/223904.223931.

[13] Thi Ngoc Trang Tran, Alexander Felfernig, Christoph Trattner, and Andreas Holzinger. Recommender systems in the healthcare domain: state-of-the-art and research issues. *Journal of Intelligent Information Systems*, pages 1–31, dec 2020. URL: https://doi.org/10.1007/s10844-020-00633-6, doi:10.1007/s10844-020-00633-6.

[14] Gediminas Adomavicius and Youngok Kwon. Multi-criteria recommender systems. In *Recommender Systems Handbook, Second Edition*, pages 847–880. Springer US, jan 2015. URL: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6{_}25, doi:10.1007/978-1-4899-7637-6_25.

[15] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems, 2010. doi:10.1561/1100000009.

[16] Félix Hernández del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804, oct 2008. doi:10.1016/j.eswa.2007.07.047.

[17] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. 2013. URL: http://arxiv.org/abs/1301.7363, arXiv:1301.7363.

[18] P H Aditya, I Budi, and Q Munajat. A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study PT X. In *2016 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2016*, pages 303–308, 2017. doi:10.1109/ICACSIS.2016.7872755.

[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*, pages 285–295, 2001. doi:10.1145/371920.372071.

[20] Robin Burke. Hybrid web recommender systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4321 LNCS, pages 377–408. Springer Verlag, 2007. doi:10.1007/978-3-540-72079-9_12.

[21] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives, 2019. arXiv:1707.07435, doi:10.1145/3285029.

[22] Jhonny Pincay, Luis Terán, and Edy Portmann. Health recommender systems: A state-of-the-art review. In *2019 Sixth International Conference on eDemocracy eGovernment (ICEDEG)*, pages 47–55, 2019. doi:10.1109/ICEDEG.2019.8734362.

[23] André Calero Valdez, Martina Ziefle, Katrien Verbert, Alexander Felfernig, and Andreas Holzinger. *Recommender Systems for Health Informatics: State-of-the-Art and Future Perspectives*, pages 391–414. Springer International Publishing, Cham, 2016. URL: https://doi.org/10.1007/978-3-319-50478-0{_}20, doi:10.1007/978-3-319-50478-0_20.

[24] Sarah Rusnak and Pamela Charney. Position of the Academy of Nutrition and Dietetics: Nutrition Informatics. *Journal of the Academy of Nutrition and Dietetics*, 119(8):1375–1382, aug 2019. URL: https://pubmed.ncbi.nlm.nih.gov/31353011/, doi:10.1016/j.jand.2019.06.004.

[25] Ajith Abraham. Rule-Based Expert Systems. In *Handbook of Measuring System Design*. John Wiley Sons, Ltd, jul 2005. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/0471497398.mm422https://onlinelibrary.wiley.com/doi/abs/10.1002/0471497398.mm422https://onlinelibrary.wiley.com/doi/10.1002/0471497398.mm422, doi:10.1002/0471497398.mm422.

[26] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology? In *Handbook on Ontologies*, pages 1–17. Springer Berlin Heidelberg, 2009. URL: https://link.springer.com/chapter/10.1007/978-3-540-92673-3_0, doi:10.1007/978-3-540-92673-3_0.

[27] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018, 2018. URL: https://doi.org/10.1093/database/bay101, doi:10.1093/database/bay101.

[28] Elica Campochiaro, Riccardo Casatta, Paolo Cremonesi, and Roberto Turrin. Do metrics make recommender algorithms? In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 648–653, 2009. doi:10.1109/WAINA.2009.127.

[29] F O Isinkaye, Y O Folajimi, and B A Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015. URL: https://www.sciencedirect.com/science/article/pii/S1110866515000341, doi:https://doi.org/10.1016/j.eij.2015.06.005.

[30] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, jul 2013. doi:10.1016/j.knosys.2013.03.012.

[31] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, pages 257–297. Springer US, 2011. URL: https://link.springer.com/chapter/10.1007/978-0-387-85820-3{_}8, doi:10.1007/978-0-387-85820-3_8.

[32] Christian Blum and Andrea Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, 2003. URL: https://dl.acm.org/doi/10.1145/937503.937505, doi:10.1145/937503.937505.

[33] Joseph Geunes. The Knapsack Problem. Technical report, 2014. doi:10.1201/b17414-6.

[34] Fabio Furini, Michele Monaci, and Emiliano Traversi. Exact approaches for the knapsack problem with setups. *Computers and Operations Research*, 90:208–220, 2018. URL: https://hal.archives-ouvertes.fr/hal-02098420, doi:10.1016/j.cor.2017.09.019.

[35] Z. Cai and Y. Wang. A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 10(6):658–675, 2006. doi:10.1109/TEVC.2006.872344.

[36] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, 1943. URL: https://projecteuclid.org/euclid.bams/1183504922.

[37] Benjamin Kille, Andreas Lommatzsch, Roberto Turrin, András Serény, Martha Larson, Torben Brodt, Jonas Seiler, and Frank Hopfgartner. Stream-based recommendations: Online and offline evaluation as a service. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9283, pages 497–517, 2015. URL: http://clef-newsreel.org/, doi:10.1007/978-3-319-24027-5_48.

[38] Michael T.M. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, sep 2018. URL: https://doi.org/10.1007/s11047-018-9685-y, doi:10.1007/s11047-018-9685-y.

[39] Panagiotis Mergos and Anastasios Sextos. Multi-objective optimum selection of ground motion records with genetic algorithms. 06 2018.

[40] Mohamed Abdel-Basset, Laila Abdel-Fatah, and Arun Kumar Sangaiah. Metaheuristic algorithms: A comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pages 185–231. Elsevier, jan 2018. doi:10.1016/B978-0-12-813314-9.00010-4.

[41] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm intelligence: From Natural to Artificial Systems, oct 1999. URL: https://oxford.universitypressscholarship.com/view/10.1093/oso/9780195131581.001.0001/isbn-9780195131581, doi:10.1515/itit-2019-0034.

[42] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021. URL: https://doi.org/10.1007/s11042-020-10139-6, doi:10.1007/s11042-020-10139-6.

[43] Julian Blank. An Introduction to Genetic Algorithms: The Concept of Biological Evolution in Optimization | by Julian Blank | Towards Data Science, 2020. URL: https://towardsdatascience.com/an-introduction-to-genetic-algorithms-the-concept-of-biological-evolution-in-opt

[44] Raúl Hector Gallard and Susana Cecilia Esquivel. Enhancing evolutionary algorithms through recombination and parallelism. *Journal of Computer Science and Technology*, 1(05):13 p., 2001. URL: https://journal.info.unlp.edu.ar/JCST/article/view/985.

[45] Anthony Jameson and Barry Smyth. *Recommendation to Groups*, pages 596–627. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. URL: https://doi.org/10.1007/978-3-540-72079-9_20, doi:10.1007/978-3-540-72079-9_20.

[46] Rashmi Hti and Maunendra Sankar Desarkar. Personalized tourist package recommendation using graph based approach. In *UMAP 2018 - Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 257–262, New York, NY, USA, jul 2018. Association for Computing Machinery, Inc. URL: https://dl.acm.org/doi/10.1145/3213586.3225233, doi:10.1145/3213586.3225233.

[47] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten De Rijke. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1502–1516, aug 2020. arXiv:1806.08977, doi:10.1109/TKDE.2019.2906190.

[48] Wenli Yu, Li Li, Xiaofei Xu, Dengbao Wang, Jingyuan Wang, and Shiping Chen. ProductRec: Product Bundle Recommendation Based on User's Sequential Patterns in Social Networking Service Environment. In *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, pages 301–308. Institute of Electrical and Electronics Engineers Inc., sep 2017. doi:10.1109/ICWS.2017.127.

[49] Yan Fang, Xinyue Xiao, Xiaoyu Wang, and Huiqing Lan. Customized bundle recommendation by association rules of product categories for online supermarkets. In *Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018*, pages 472–475. Institute of Electrical and Electronics Engineers Inc., jul 2018. doi:10.1109/DSC.2018.00076.

[50] Guannan Liu, Yanjie Fu, Guoqing Chen, Hui Xiong, and Can Chen. Modeling buying motives for personalized product bundle recommendation. *ACM Transactions on Knowledge Discovery from Data*, 11(3):1–26, mar 2017. URL: https://dl.acm.org/doi/10.1145/3022185, doi:10.1145/3022185.

[51] Mohit Sharma, F. Maxwell Harper, and George Karypis. Learning from sets of items in recommender systems. *ACM Transactions on Interactive Intelligent Systems*, 9(4):27, 2019. URL: https://doi.org/10.1145/3326128, arXiv:1904.12643, doi:10.1145/3326128.

[52] Loc Do, Hady W Lauw, and Ke Wang. Mining Revenue-Maximizing Bundling Configuration. In *Proceedings of the VLDB Endowment*, volume 8, pages 593–604. 2015. doi:10.14778/2735479.2735491.

[53] Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pages 349–354, 2012. URL: http://api.yes.com.

[54] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. DJ-MC: A reinforcement-learning agent for music playlist recommendation. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 1, pages 591–599. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), jan 2015. URL: http://arxiv.org/abs/1401.1880, arXiv:1401.1880.

[55] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective, dec 2011. URL: https://dl.acm.org/doi/10.1145/2037661.2037665, doi:10.1145/2037661.2037665.

[56] Rung Ching Chen, Yung Da Lin, Chia Ming Tsai, and Huiqin Jiang. Constructing a diet recommendation system based on fuzzy rules and knapsack method. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7906 LNAI, pages 490–500. Springer, Berlin, Heidelberg, 2013. URL: https://link.springer.com/chapter/10.1007/978-3-642-38577-3{_}50, doi:10.1007/978-3-642-38577-3_50.

[57] Jing-jie ZHU, Ling-ling SHEN, Huai-gang WU, Ting YUAN, and Gang QIAN. A Package Recommendation Model Based on Credit and Time. *DEStech Transactions on Computer Science and Engineering*, 0(wcne), mar 2018. URL: http://dpi-proceedings.com/index.php/dtcse/article/view/19906, doi:10.12783/dtcse/wcne2017/19906.

[58] Remigio Hurtado Ortiz, Rodolfo Bojorque Chasi, and César Inga Chalco. Clustering-based recommender system: Bundle recommendation using matrix factorization to single user and user communities. In *Advances in Intelligent Systems and Computing*, volume 787, pages 330–338. Springer Verlag, jul 2019. URL: https://link.springer.com/chapter/10.1007/978-3-319-94229-2{_}32, doi:10.1007/978-3-319-94229-2_32.

[59] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 407–416, 2011. doi:10.1109/ICDM.2011.118.

[60] Jie Xu, Tianwei Xing, and Mihaela Van Der Schaar. Personalized Course Sequence Recommendations. *IEEE Transactions on Signal Processing*, 64(20):5340–5352, dec 2016. URL: http://arxiv.org/abs/1512.09176http://dx.doi.org/10.1109/TSP.2016.2595495, arXiv:1512.09176, doi:10.1109/TSP.2016.2595495.

[61] Porto A. Martins I., Oliveira L. Tabela da Composição de Alimentos. URL: http://www2.insa.pt/sites/INSA/Portugues/AreasCientificas/AlimentNutricao/AplicacoesOnline/TabelaAlimentos/Paginas/TabelaAlimentos.aspx.

[62] Seyedali Mirjalili. Genetic algorithm. In *Studies in Computational Intelligence*, volume 780, pages 43–55. Springer Verlag, 2019. doi:10.1007/978-3-319-93025-1_4.

[63] Ahmad Hassanat, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri, and V B Surya Prasath. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information*, 10(12), 2019. URL: https://www.mdpi.com/2078-2489/10/12/390, doi:10.3390/info10120390.