**Bio-inspired Hybrid Techniques for Generating Food Menu Recommendations Using Cuckoo Search Algorithm**

LICENSE THESIS

|  |  |  |
| --- | --- | --- |
|  | Graduate: | **Akos Tiberiu BOROS** |
|  |  |  |
|  | Supervisor: | **Prof. Dr. Eng. Ioan SALOMIE** |

**2015**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DEAN, |  | | HEAD OF DEPARTMENT, | |
| **Prof. dr. eng. Liviu MICLEA** |  | **Prof. dr. eng. Rodica POTOLEA** | |  |

Graduate: **Akos Tiberiu BOROS**

**DIET4Elders: a Service Oriented Architecture for the Prevention and Self-Management of Malnutrition**

1. **Project proposal:** *With the rapid growth of the population, studies have shown that in Europe, by 2050, the senior population will double to a percentage of 54.2%. This will result in a need of medical care services that the current health care systems simply cannot withstand. A user-oriented nutrition health care system is proposed in this thesis. By using hybrid bio-inspired optimization algorithm and real nutritional data of food ingredients, together with food packages provided by local food vendors, this project succeeds in generating the most fit daily food menus according to the user's health needs.*
2. **Project contents:** *Presentation page, Advisor's Evaluation, Introduction, Project Objectives, Bibliographic Research, Analysis and Theoretical Approach, Hybrid Technique.., Detailed Design and Implementation, Testing and Validation, User’s Manual, Conclusions, Bibliography, appendices.*
3. **Place of documentation**: Technical University of Cluj-Napoca, Computer Science Department
4. **Consultants**: Dr. Eng. Viorica Rozina CHIFU, Dr. Eng. Cristina POP
5. **Date of issue of the proposal:** November 3, 2014
6. **Date of delivery:** June 18th, 2015

|  |  |
| --- | --- |
| Graduate: | Akos Tiberiu BOROS |
|  |  |
| Supervisor | Prof. Dr. Eng. Ioan SALOMIE |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a**)** BOROS AKOS TIBERIU, legitimat(ă) cu C.I. seria XH nr. 779535   
CNP 1921128055076, autorul lucrării *Bio-inspired Hybrid Techniques for Generating Food Menu Recommendations Using Cuckoo Search Algorithm* elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea IUNIE a anului universitar 2015 declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data  18.06.2015 |  | Nume, Prenume  Boros Akos Tiberiu |
|  |  |  |
|  |  | Semnătura |

Table of Contents

[Chapter 1. Introduction 1](#_Toc422322365)

[1.1. Project context 1](#_Toc422322366)

[1.2. Thesis Structure 2](#_Toc422322367)

[Chapter 2. Project Objectives 4](#_Toc422322368)

[2.1. Problem Statement and Motivation 4](#_Toc422322369)

[2.2. Objectives 5](#_Toc422322370)

[2.3. Requirements 5](#_Toc422322371)

[2.3.1. Functional Requirements 5](#_Toc422322372)

[2.3.2. Non-functional Requirements 7](#_Toc422322373)

[Chapter 3. Bibliographic Research 8](#_Toc422322374)

[Chapter 4. Analysis and Theoretical Foundation 14](#_Toc422322375)

[4.1. Ontologies 14](#_Toc422322376)

[4.1.1. OWL and SWRL 15](#_Toc422322377)

[4.1.2. SPARQL 15](#_Toc422322378)

[4.1.3. Ontology Modelling from Medical Perspective 16](#_Toc422322379)

[4.2. Food Modelling 20](#_Toc422322380)

[4.2.1. USDA Database 20](#_Toc422322381)

[4.2.2. USDA Database Structure 21](#_Toc422322382)

[4.3. Metaheuristics 23](#_Toc422322383)

[4.4. Hill-Climbing 26](#_Toc422322384)

[Chapter 5. Hybrid Technique for Generating Food Menu Recommendations using the Cuckoo Search Algorithm 28](#_Toc422322385)

[5.1. Cuckoo Breeding Behavior 28](#_Toc422322386)

[5.2. Cuckoo Search 28](#_Toc422322387)

[5.2.1. Cuckoo Search Solution Representation 30](#_Toc422322388)

[5.2.2. Cuckoo Search Fitness Function 31](#_Toc422322389)

[5.2.3. The Crossover based creation of new solution 34](#_Toc422322390)

[5.2.4. Hill Climbing - hybridization point in CS 35](#_Toc422322391)

[5.3. Cuckoo Search Hybrid Versions 36](#_Toc422322392)

[Chapter 6. Detailed Design and Implementation 39](#_Toc422322393)

[6.1. Design of Experimental Prototype 39](#_Toc422322394)

[6.1.1. Conceptual Architecture 39](#_Toc422322395)

[6.1.2. Database Model 40](#_Toc422322396)

[6.1.3. Ontology Classes 41](#_Toc422322397)

[6.1.4. Use Case Model 43](#_Toc422322398)

[6.1.5. Activity Diagram 44](#_Toc422322399)

[6.1.6. Deployment Diagram 44](#_Toc422322400)

[6.1.7. Modules Architecture 46](#_Toc422322401)

[6.1.8. Class Diagrams 48](#_Toc422322402)

[6.2. Implementation Details 51](#_Toc422322403)

[6.2.1. Tools and Technologies 51](#_Toc422322404)

[6.2.2. Code Sequences 52](#_Toc422322405)

[Chapter 7. Testing and Validation 55](#_Toc422322406)

[7.1. Test Scenarios 55](#_Toc422322407)

[7.2. Validation 56](#_Toc422322408)

[7.3. Best Configuration Setup 58](#_Toc422322409)

[7.3.1. PA Variation 58](#_Toc422322410)

[7.3.2. NestNumber Variation 59](#_Toc422322411)

[7.3.3. MaxIterations Variation 61](#_Toc422322412)

[7.4. Cuckoo Search Hybrid Versions Comparison 62](#_Toc422322413)

[Chapter 8. User’s manual 64](#_Toc422322414)

[8.1. Installation manual 64](#_Toc422322415)

[8.2. User guide 64](#_Toc422322416)

[Chapter 9. Conclusions 66](#_Toc422322417)

[9.1. Contributions and Achievements 66](#_Toc422322418)

[9.2. Critical Analysis of Results 66](#_Toc422322419)

[9.3. Future Development 66](#_Toc422322420)

[Bibliography 68](#_Toc422322421)

[Appendix 1 69](#_Toc422322422)

# Introduction

One of the most important and most difficult things in life is obtaining balance. This of course, goes to nutrition as well. One can observe, that a proper equilibrium in the diet of a person can prevent most diseases from occurring. Because of bad eating habits and a hazardous diet, diseases are increasing in an alarming percentage amongst the population of most countries, with certain predilections to some classes of people.

Despite the strong evolution of the science of nutrition, it has become more and more neglected, and people do not assure their daily recommended nutrients intake, which eventually, in a point in life will backfire. If an equilibrium is not offered, the human body is thrown in an off-balanced state, and the response will be undesired medical conditions.

Nutrition is arguably amongst the few ways that one can influence their health, because it is a responsibility that nature has assigned to every person, and this can be done by understanding and complying with the laws of nature that reside within the human body.

It is advisable that one should have a healthy daily menu, in order to lead a healthy life. For a healthy daily menu, the meals should be rich in all the nutrients that are needed. The daily diet of a person should have the nutrients values as close as possible to the Recommended Daily Allowance (RDA) . It is true of course, that the RDA values for nutrients differs from a person to another. These values rely on the individual’s weight, height, gender and age as well as taking into consideration the physical activity level.

## Project context

Nowadays there is the tendency to help older adults become more independent during their daily life activities. As aging may come with several health problems the older adult may become clumsy and unable to perform some of these daily life activities. One such activity involves preparing food and procuring the goods required for eating. The incapacity of performing such an activity may have a great impact upon the older adult’s health, as poor eating may quickly result in malnutrition which triggers other serious issues such as contributing to or aggravating chronic and acute diseases, speeding up the development of degenerative diseases, and even death.

Malnutrition amongst seniors has become more and more frequent in the last years. It is estimated that in Europe around 15% of the elderly population is affected by poor nutrition and malnutrition caused by the problems of ageing such as decrease in sensitivity, poor dental health, lack of transportation, physical difficulty, forgetfulness and other issues.

The costs determined by malnutrition are not negligeable. According to studies, United Kingom spends up to £7.4 billion per year. The prevalence of malnutrition is high and an issue is that it is hard to recognize malnutrition, because the effect appear with a delay and tracking the nutrition process of an individual can be too costy.

A nutrition system that would generate daily food menus that are best for a user’s profile would tackle this problem. By using food packages that are offered by the local food vendors, the nutrition system would provide the user with both the possibility to procure daily meals from the catering parties and order the menus are most appropriate according to the user’s medical condition and profile.

The food menu generation for a person must offer a daily menu that best fits the user. Because the metabolism is different from one person to another, the correct recommended nutrient values should be computed, in order for the menu to have ideal values to which it can compare its nutrients to. Food menus must also receive a ‘score’, that will reflect how well the combination of menus is a fit to the user’s profile. That ‘score’ in genetic algorithm terms is called the fitness value, and it is computed by applying a designed fitness function to, in our case, the combination of food menus that represent a day’s meals.

Because of the fact that the search space could grow exponentially with the number of food packages that are provided by local vendors, the time necessary for the search of the optimal solution by using brute-force search methods is very high. As this is more of an artificial intelligence problem, the solution proposed in this thesis for this is to use a heuristic search algorithm inspired by nature processes, more accurately, a swarm-intelligence-based algorithm hybridized with genetic algorithm heuristics.

Due to the drawbacks given by the computation of conventional numerical methods in solving complex optimization problems, researchers may have to rely on meta-heuristic algorithms. Since the explosion of artificial intelligence and growth of interests in this field, many meta-heuristic algorithms have been successfully applied to various engineering optimization problems. For most complicated real-world optimization problems, they have provided better solutions in comparison with conventional numerical methods.

To imitate natural phenomena, most meta-heuristic algorithms combine rules and randomness. These phenomena include the biological evolutionary processes, such as genetic algorithm, evolutionary algorithm and differential evolution, animal behavior, such as particle swarm optimization, tabu search and ant colony algorithm, as well as physical annealing processes, such as simulated annealing.

Deterministic optimization algorithm tend to fail as the problem size increases, as opposed to the bio-inspired stochastic approach that is a computationally efficient alternative to the former. Meta-heuristics are based on the iterative improvement of either a population of solutions or a single solution, and both types of heuristics can be applied in order to obtain a more efficient hybrid algorithm.

On the subject of bio-inspired computation, a vast literature exists for solving an impressive array of problems, and, more recently, studies have reported on the favorable results and success of such methods for solving difficult problems in all key areas of computer-science.

Many projects have been started, ideas have been launched and visions have been made regarding health care systems in the past years because of the rapid technological progress and the market need. And currently this is a field that has yet to be fully explored because the applications of health care systems can be infinite. Limitations that have been stopping development will soon be removed or reduced, thus ideas of applications that were considered to be impossible before can now be put to practice.

## Thesis Structure

The structure of everything hereafter is as follows:

* **Chapter 2** conveys a general overview of the purpose of this thesis together with some sub-goals, reasoning and motivation for choosing the subject and some specification of the underlying project.
* **Chapter 3** covers related work which was done in the same field, a bibliographic study that was done before this thesis was even began. Here the reader will find some medical approaches to this problem and some facts that help the reader obtain a better understanding of the whole domain of the problem.
* **Chapter 4** illustrates the theoretical foundation on which this project was built on. This includes some techniques and algorithms which form the backbone of the system and which ended up being used in its implementation.
* **Chapter 5** presents the implementation details covering design decisions and code-level analysis of the project. Here, the reader can see how the theoretical foundation found in chapter 4 was abstracted and represented into code.
* **Chapter 6** revolves solely around how the system was evaluated and tested, which parameters were involved and how it behaved in different situations. Measurements and before and after tests are also present in this chapter.
* **Chapter 7** consists of a step-by-step guide to installing and using the built system accordingly.
* **Chapter 8** presents a summary of the proposed solution with emphasis on scientific contributions, results and innovation.
* The **Bibliography** is also provided at the end of the paper
* An **Acronyms** sections is available for abbreviations
* The final section is comprised of an **Appendix**

# Project Objectives

## Problem Statement and Motivation

As the number of food providers is continuously increasing, the number of food offers is extremely high making the choice of the most appropriate offer that satisfies both the nutritionist’s diet and older adult’s constraints difficult for the older adult. To overcome this issue, in this thesis a bio-inspired optimization method is proposed that, based on the set of food offers available on the market, provides the food offers required to cover the breakfast, lunch, dinner and snack meals for one day that best satisfy the nutritionist’s diet and older adult’s cost and time constraints.

Studies have shown that the ratio of retirees to workers in Europe will double to 0.54 by 2050. Because of the high rate of mortality and low rate of natality, the ratio between the age groups will greatly incline towards the higher numerosity of the elderly. The growth of the senior population will result a need of medical care services that the current medical care systems simply cannot withstand.

Following the up-to-date evolution of the population, it is easy to conclude from observations that there will be a continuous growth in the need for elderly carers. By using this system for generating food menus for the elderly, such situations can be avoided or treated. The number of human carers can be reduced by the user of automated carers, that are, without doubt, more superior in many ways for certain tasks, if designed properly.

By offering an automated carer for each person, the problem of nutrition amongst the population can be aided and also a great load of the medical health care services will be exempted. The automated carer should generate food menus that are most appropriate according to the user personal profile. This will be done by searching the optimal combination of food packages that follow certain filters of the user.

Because of the fact that setting a personalized diet for each individual is difficult considering that it does take a lot of time resources, which apparently are becoming more and more scarce with the day to day activities, it is clear that the need for an automated care that would generate food menus is high.

By making use of the advanced technological progress and bringing it to the whole health care process, the overall medical system care experience will be greatly improved. This project is will do just that, as small step in the ladder of health care system improvement. By having a system that assures proper nutrition to elder people, the health care becomes more efficient, more available (the „cyber” carer will be present anytime) and it will reduce costs grealty.

This will also stimulate the food vendor market to provide food packages that are healthy and nutritious, as well as offer a centralized system that after gathering all data, will generate food packages also by the user’s budget and patience (if the user wants the delivery time to be as short as possible).

This application can be extended greatly. Special features for patients with certain medical conditions could be added. Special regimes for patients suffering from diabetes, anaemia, certain vitamin or mineral defficiencies, etc could be available by certain recommended daily allowance preset values.

This application can also aid conditions such as overweight or underweight by adjusting the nutrients value according to formulas for losing weight provided by nutriotionists’ research.

## Objectives

The main objective of this project is that given a repository of food packages (i.e. set of food items corresponding to breakfast, lunch, dinner or snack) provided by several food providers, find the optimal combination of food packages for each of the meals of a day such that they meet an older adult’s profile (containing information about the preffered food and allergies), prices and delivery time constraints and recommended diet.

In order to obtain this goal of food menu generation, the following objectives must be achieved:

* Analysis on the personalized food menu generation by enforcing the need of the development of a bio-inspired technique
* Thorough study of existing meta-heuristics that can be applied to the context of this problem, comparison and final decision regarding the chosen method
* Design of an ontology model fit to the medical and nutritional domain
* Design of a database fit to the nutrition domain
* Population with real data of the nutritional database, meaning ingredients with authentic nutritional values and a starting set of real recipes, together with some mockup data for the food packages offered by food vendors
* Design of a hybrid search optimization algorithm inspired from nature, more particularly, by the behaviour of the mating process of cuckoo birds. The algorithm should be subject to analysis, observation of its advantages and disadvantages, in order to improve the overall results
* Design of a prototype of the application with the integrated model and algorithm that will successfully prove the method works
* Evaluation of the classic algorithm and the hybrid version and perform analysis of the obtained results with the variation of the adjustable parameters

## Requirements

The starting point of each application is a set of requirements. These requirements can be either functional or non-functional, one dealing with what the application should be able to do and the others cover constraints and requirements on how these functional specifications should be implemented. The following section presents the two types of requirements

### Functional Requirements

The behavioral and functional aspects of the application are covered by the functional requirements. These requirements are basically the tasks which have to be done in order to get to the final goal. In our case, the main functional requirements would be to develop a bio-inspired technique for generation of recommended food menus for the elderly. The food menus generated should take into account the personal profile of the user, thus the menus being dependent of the following:

* Doctor’s medical prescription, if any
* User’s allergies or unallowed ingredients
* User’s likes and dislikes, i.e. ingredients that the user would prefer the menu to have, and ingredients that the user does not want to appear in the food menus’ recipes
* Food menu cost from local food vendor
* Food menu delivery time duration from local food vendor
* Recommended Daily Allowance for the ideal nutrition values, that is a function of the user’s
  + Age
  + Gender
  + Height
  + Weight
  + Physical Activity Factor (PAF)

The nutrients are divided into micronutrients and macronutrients. For the Recommended Daily Allowance the following nutrients should be taken into account:

* Micronutrients:
  + Calcium
  + Iron
  + Sodium
  + Vitamin A, B, C, D
* Macronutrients
  + Lipids
  + Carbohdrates
  + Proteins

The search algorithm that will be responsible should be able to find a solution that has a high fitness and also integrate diversity of food menu by combining them in an intelligent way.

Thus the product functionality can be divided into **features** as follows:

* **Personalized general food menu recommendation**. The system automatically computes the values of macro- and micro-nutrients that are recommended for a person's daily intake based on his or her profile: age, weight and amount of physical activity. The computations are based on the well-known nutritional formulas Harris-Benedict Equation and Dietary Reference Intake (DRI).
* **Integration of doctor’s prescriptions.** If the doctor prescribes a prescription to a patient, the previously described general recommended values for nutrients are overwritten.
* **Integration of user allergies.** If the user has allergies then the results will not include the recipes that have ingredients that the user is allergic to. The respective ingredients are removed at the beginning from the work data set.
* **Implementation of the Cuckoo Search Optimization algorithm.** This project is a research project that implements the bio-inspired algorithm entitled Cuckoo Search.
* **Algorithm hybridization.** The Cuckoo Search (CS) algorithm is improved by adding different heuristics from other artificial intelligence algorithms like swarm particle optimization or others as the CS can be considered as being part of one. Heuristics are benchmarked to see which improve algorithm performance the most.
* **User profile management.** The user is able to modify his profile data. This is available through a graphical user interface (GUI) that can be manipulated either by the user, or the user carer.
* **Food vendor interface.** This feature should enable food vendors to add food packages with cost and delivery time information

### Non-functional Requirements

System qualities, as they are also called, are the requirements used to evaluate the behavior of a system. There is quite a long list of non-functional requirements but we will only analyze those which are relevant to our system.

* **Efficiency**: Because of the fact that this application will require a search optimization algorithm, it is very important that the application should not perform any tasks that would waste resources. The algorithm must be subject to analysis in order to adjust it’s variable parameters so that the best results are achieved with the minimum of effort.
* **Scalability**: Large amounts of data should not be an issue as the system should be able to operate and perform its tasks with small or large input all the same.
* **Modularity**: Adding more modules should not cause issues nor require great architectural changes. Adding more modules should be a seamless problem with as little implementation impediments as possible.
* **Accuracy**: The system should provide accurate results. Delicate as it may seem, a close to as if it were random result set is of no use because then the algorithm would not output the desired results.
* **Reliability**: Feeding the system with invalid or corrupt data should yield errors not crash it. Error handling should be present in all the layers of the applications.

Of course there could be other, less important non-functional requirements but which will not be mentioned here. Security might be a problem once a crawler is created on top of the application which should provide constant input to it, however, this is not yet the case.

# Bibliographic Research

Several automated nutrition systems have already been developed similar to project’s idea, however, with some differences. Some nutrition systems have been presented in some articles where the main goal is to provide a personalized nutrition menu and there are some that are more oriented towards overweight people or regimes for diabetes people and so on. Nutrition systems that are particularly focused on the elderly are not that many.

In this section several approaches to this problem will be described. Some notions will be presented that are similar or related to the problem tackled in this diploma thesis. Because of the fact that there are 3 main perspectives (medical perspective, food generation algorithm perspective and model perspective – including ontologies) that are related to the problem context, the information from the articles bellow will stick mainly to these.

In article **[1]** a nutrition counseling system is proposed for offering variety in the daily menus. Due to the effects of industrial growth, drastic changes in consuming behavior, standard of living and lifestyle have appeared. People stopped preparing food by themselves and have adopted the well known habit of the more wealthy countries of procuring the daily food menus by eating out regularly. This often means that they cannot control the characteristics of the food they take, such as its taste calories per portion, and other nutritional details. The results of this is appearance of obesity and unbalanced nutrition.

Also, because of the industrial growth, the population percentage with low physical activity level has risen due to the sedentary lifestyle which affects the nutritional needs of the human body. Thus, due the fact that the daily intake of calories remains the same the result is that the population is more and more prone to obesity. Previously, researchers have designed a few expert or intelligent systems in order to deal with this issue. Lee et al. proposed “An intelligent Healthcare Agent for Food Recommendation at Tainan City”, but the quantity of calories daily that people should consumed is unknown. In addition to this, another well known issue that has arisen with this change is that a large amount of people eat the same menus on a daily basis, leading to an overall small variety of nutrients intake.

The solution proposed in this article [1] is clustering food, which is a procedure to identify attribute similarities between foods. If comparable foodstuffs are allotted to the same cluster then one can replace certain foods of similar characteristics in order to avoid generating the same monotonous dish or menu suggestions each time.

The essential nutritional constituents selected in this approach consists of the following:

* Energy - The energy used for the human body to work properly. The amount of energy in a food depends on the amount of calories from such food. The amount of energy required also depends on age, height, weight, gender and activity level
* Carbohydrates - The macronutrients needed the most. Based on “Dietary Reference Intakes” published by the United States Department of Agriculture (USDA), around 50% of calories come from carbohydrates. There are many reasons why carbohydrates are very important for human body and they are as follows:
  + The main source of energy in human’ body is carbohydrates.
  + The central nervous system, the heart, the brain, and the muscles use carbohydrates to function correctly.
* Proteins - With reference to the USDA’s, “Dietary Reference Intakes”, a small amount of calories come from protein, for which, most of people get this kind of nutritional need sufficiently. People require protein because:
  + Growth (In particular, chiefly influential for children, teenagers, and pregnant women)
  + Repairing other cells.
  + Body immune system.
  + Producing mainly enzymes and hormones
  + Being energy when carbohydrate is not available
* Fats - Fats have been portrayed as the main culprit for causing obesity and therefore have a poor reputation. However some fats are essential for survival. According to the USDA, “Dietary Reference Intakes” one-third of calories come from fats. Fats are required for:
  + Growth
  + The most concentrated source of energy by fats
  + Taking care of all cell membranes

The flow diagram for the nutrition system described in this article [1] is depicted in the figure 3-1.

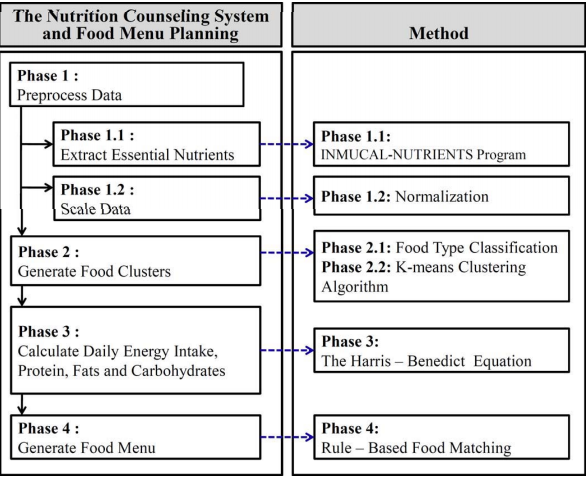


Figure 3-1. Nutrition counseling framework [1]

The first phase consists of extracting the nutritional information from the menus as well as scaling the values in order for them to be evaluated as a whole. The second phase is the food cluster generation step. The food types are classified by certain criteria and second tier clustering by using a K-means clustering algorithm is the performed.

In the third phase, the macronutrients are computed with the help of the well known Harris-Benedict equation.

Some experimental results of this approach that are presented show that all the user’s menus have higher percentage resemblance values in terms of energy content than nutrient composition. In Figure 3-2 an example of the outpul of a resulted daily menu solution is shown.

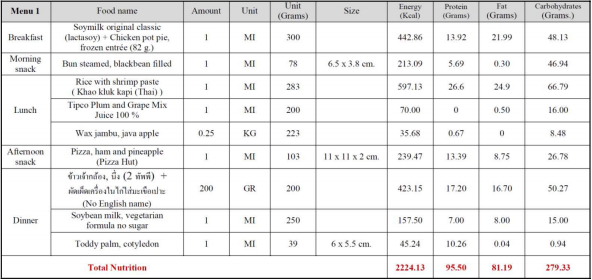


Figure 3-2. Output example of solution [1]

The results from the eight user volunteers suggest that the NCS is capable of generating individual menus for three meals (breakfast/lunch/dinner) and also offering a variety of foodstuff choices. Overall the users were satisfied with the program and its interface with many willing to consider using it again. The NCS can be useful for people who are concerned with their health and more specifically, their dietary habits.

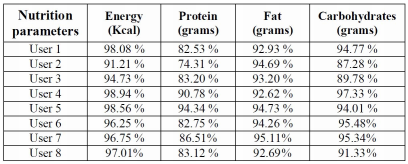


Figure 3-3. Nutrition Counseling Results [1]

In article **[2]** the development and design of a personalized nutrition and food planning system for older people that uses a food-oriented ontology together with an expert system is described.

The world population is rapidly increasing, and due to growing health and medical care this is also the case for the elderly. Old people have limited regenerative abilities and are more prone to disease, syndromes, and sickness than other adults. In addition, their bodies also produce less sex hormone which results in the accumulation of body fat or not eating a full nutritional benefit, which also may cause nutritional based diseases. Among the diseases that older people often suffer from are hypertension, diabetes, osteoarthritis, osteoporosis, heart disease, and enlarged prostate.

In order to solve this problem, in this article [2] an approach to develop a personalized food and nutrition planning system for the elderly on the basis of a food-oriented ontology for the management of knowledge about food and nutrition has been presented. A use of an expert systems for food an nutrition planning for the elderly is made, which takes into account the physical condition and needs of the older people accurately and efficiently. The system is developed for the use as a hospital service to a provide food and nutrition plan for the older patients.

The system framework is shown in Figure 3-4. The framework comprises a (1) User Interface; a (2) Food Planning System; the (3) Personal Health Record; a (4) Knowledge Base; and a (5) Database.

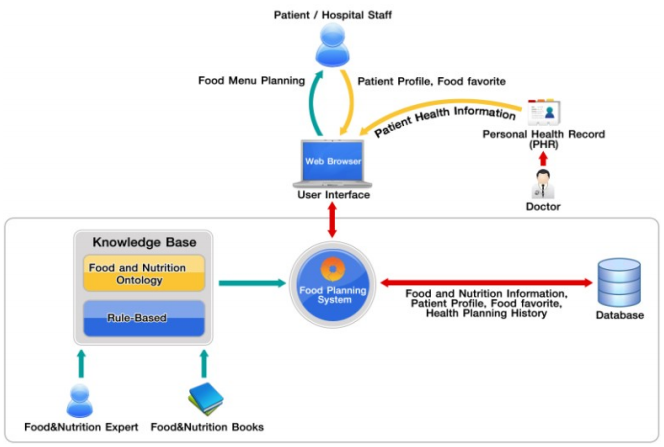


Figure 3-4. Ontology-Drived Personalized Nutrition System Framework [2]

This paper uses the concept ontology from (Snae, & Brueckner, 2008) in Figure 3-4 adapted to suit the aims of this research. Ontologies have become recognized and widely used in state-of-the-art gathering of information and intelligence, information retrieval, knowledge representation, and database management systems (Thomas, 1993). RDF (RDF, 2004), RDF Schema (RDF Schema, 2004) and OWL (OWL, 2004) to describe food and nutrition knowledge have been employed. These languages are proposed by W3C.

Protégé ontology editor (Protégé, 1987) has been used in order to create, expand and edit the ontology. The ontology makes use of food specifics, such as animal-based or plant-based food, flavors, added by nutritional facts, such as vitamins, proteins, fats, carbohydrates, and so on. This information can be used to recommend food and nutrition an appropriate to the physical condition and personalized to the patient likes and dislikes.

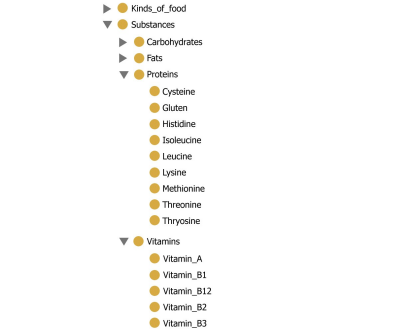


Figure 3-5. Ontology Nutritional System [2]

In article **[3]** an approach to solve the problem of obesity or being overweight is presented. Obesity or being overweight in general often leads to other more severe diseases. With more than half of the population in the western countries being overweight or even obese, systems are developed to help users make healthier food choices by recommending healthier recipes or food items according to the user’s needs or likes and dislikes.

Professional nutritionists utilize a broad selection of tools to establish an individual’s nutritional intake. These include food diaries, 24-hour recalls or questionnaires, with food diaries or 24-hour recalls being preferred giving superior measures. With food diaries the user records all amounts of consumed foods and beverages over a period of time preferably at the time they were consumed, usually no more than 3 or 4 days are recorded. For 24-hour recalls the user has to give a report during a meeting with the nutritionist of every food or beverage consumed over the last 24 hours or the last day.

More recently systems were developed which calculate the amount of food intake with the help of pictures of meals the user has taken. Applications for this are already available, which identify food pictures in the user’s web archives and calculates the dietary balance of meals.

In article **[4]** a review of the computer aided diabetes therapy is presented introducing GIGISim (Glucose-Insulin and Glycemic Index Web Simulator) e-learning tool based on the glucose and insulin plasma levels simulation models and genetic algorithms optimization

Together with the system, recently reported solutions assisting diabetes therapy were summarized and their functionality presented. Artificial intelligence is applied in GIGISim tools to improve patients’ management and health awareness. Interactive, diabetes-dedicated simulators, supported with genetic algorithms (GA) have a great deal of educational potential for patients and their families, and may also offer a means of training for health-care professionals.

In this paper the newly developed, diabetes dedicated, educational tools compared to existing approaches and solutions is presented. Glucose-Insulin and Glycemic Index Web Simulator (GIGISim) serves as a graphical simulator of postprandial glucose profiles of diabetic individuals, and can be accessed via Internet. Results are based on human carbohydrate metabolism model and influenced by user’s diet – meals and food products can be entered into the system and possible glucose and insulin fluxes simulated.

One of the main advantages of using GIGISim is the possibility to create personalized profiles and perform simulations based on adapted model. Information on age, body mass index (BMI) and possible diabetes is used to personalize simulation result by choosing model parameters from predefined sets. More advanced method includes patient’s results of Oral Glucose Tolerance Test as one of the algorithm inputs, adapting dynamically the set of model parameters.

The use of this system has been limited to educational purposes, because model takes into account only a confined number of the factors associated with glucose metabolism, and model parameters are not easily individualized to accurately simulate metabolic processes for all specific diabetes individuals.

# Analysis and Theoretical Foundation

## Ontologies

Our model containing that will be designed by using ontologies. Ontologies are a way to abstract data in a way similar to the real world model by using classes, individuals, data properties and object properties. Ontologies support the representation of different taxonomies and vocabularies. The concept of class used in defining ontologies is similar with the concept of class used in the object oriented programming languages. Classes are templates or blue prints that represent a group of entities that have the same properties. Data from the real world may be classified by using a hierarchy of classes and subclasses, similar to the concepts of object oriented programming where real objects are mapped in a semantic transformation process. Individuals are instances of classes, and can have data properties and object properties. The line between individuals and classes is hard to determine, and sometimes the same entity may be represented in both ways. Data properties are similar with the fields from object oriented languages, and can have different types such as Boolean, integer, string, etc. Object properties represent relations between individuals.

One of the characteristics that differentiate the ontologies from the other representations of the data is the fact that the ontologies are very flexible, and sometimes the same entity can be represented in different manners: as a class, as an individual, as a data property, or as an object property. Ontologies are a dynamic way to represent data, they provide a multitude of different ways for the representation of the same set of data

For building an ontology, in general it is necessary to use a special ontology language, for example OWL (Web Ontology Language). OWL provides ways to represent the elements that constitute an ontology: ontology classes, ontology subclasses, data properties, object properties, and individuals. OWL represents elements from the ontologies in a simple manner. The syntax of OWL is similar with the syntax of XML. The ontologies can be saved in .owl files which can be viewed from different applications, such as Protégé, or they can be opened from different developing environments, by using special tools

There are two main important concepts when defining object properties or data properties:

* Domain of the property - represents the set of classes or data types to which the property refers
* Range of the property - represents the values that the property can take

Also, object properties can have different characteristics such as functional (can have only one value), inverse (to some other property), sub-property (of another property), or equivalent (to some other property). One important aspect that must be taken into consideration is the way the ontologies are built. There are many ways to represent data in the form of ontologies. There are situations in which the same entity can be represented as an individual or as a class. Also there are cases in which a data property can replace the subclass relation.

Data for the project will be provided from different domains such as food domain, diet domain, health domain, diagnostic domain and prescription domain. It is very important to group together the elements that belong to the same domain, while providing a flexible way to deal with any changes that can appear. This fact can be obtained by constructing an ontology which contains other small ontologies. Each of these small ontologies will represent a different domain.

### OWL and SWRL

In [7], data representation possibilities include axioms, including class declaration, individual declaration, class assertion, subclass assertion, property declaration, property assertion, negative property assertion and annotation assertion. SWRL rules are a consequence of the fact that OWL language is not able to express all possible relations. The expressivity of OWL can be extended by adding rules to the ontology. SWRL rules are similar with the rules from Prolog or Datalog languages. Rules have the format premise-conclusion. The premise is represented as a list of statements connected by ‘and’, and the conclusion is represented by a list of statements that become true when the premise is true.

Among the things that can be inferred by using SWRL are the following: an individual is an instance of some specific class, an individual has a certain data property, an individual is related to another individual by a certain object property etc. The rules increase the expressivity of the ontologies. It would take too much time to represent all the data of the ontologies. By writing a small number of rules regarding the classes, individuals, data properties, and object properties of the ontology, the knowledge may be doubled, tripled, or increased with any rate. Inferring information becomes a time consuming operation when the ontology data is too large and the rules are too complex.

Limitations are not exempted from OWL and SWRL. OWL cannot express relations between individuals referenced by properties. There are concepts that cannot be expressed in OWL but which can be expressed in SWRL. A limitation of SWRL is the fact that arbitrary SWRL rules lead to undecidibility. Only the so DL-safe rules are implemented in reasoners. DL, as a fragment of FOL, cannot express fuzzy expressions, non-monotonicity, propositional attitudes, and modal logic including possibility and necessity, epistemic modalities, temporal logic and deontic logic.

Using SWRL rules has some disadvantages. One of the is the fact that the ontology increases after each calling of the reasoner, and thus all the information that was inferred before remains in the ontology. A solution would be to remove the properties that are no longer true from the ontology, but this is a very complicated process. There are situations in which it is impossible to determine which properties have two values, when in fact they need to have only one value. Another disadvantage of using SWRL is the time. It takes too much time to infer new data due to the complexity of the rules. This period of time can be saved by storing more information in the database, but in this case the role of the reasoning rules becomes useless.

### SPARQL

For querying information that is stored in ontologies, as in [5] is exemplified, SPARQL queries are used. A very important aspect of the semantic systems is the ability to query data. It provides some advantages such as including restrictions and other conditions. It is possible to change the order in which the results are retrieved, to limit the number of the results or Chapter 3 14 to change the starting element. SPARQL is similar with sql and has the advantage that it can be used to query an ontology. Complex queries may be written using SPARQL in a syntax easy to understand. This query language also permits some mathematical functions such as sum, count, min, max. For data which is not explicitly defined by the user, one can query data which is inferred from existing data, which is yet another advantage of using SPARQL.

As for query techniques, the methodologies vary, as stated in [6]. The first option is to use SWRL rules, while the second option is to use SPARQL queries. When using SWRL rules, without SPARQL, querying information becomes a complicated process. When the data that is retrieved refers to a single individual, data property or object property, there are no major problems. The situation changes in the case data from multiple individuals must be retrieved, and some operations are performed on this data (sum, count, etc.).

### Ontology Modelling from Medical Perspective

The following model is based on Dorin Moldovan’s diploma thesis and adapted to our specific context. Different ontologies will be presented in detail. The highest level ontology is the nutrition care process ontology. It will be composed from other four ontologies:

* **Nutrition monitoring ontology**
* **Nutrition assessment ontology**
* **Nutrition diseases diagnostic ontology**
* **Nutrition intervention ontology**

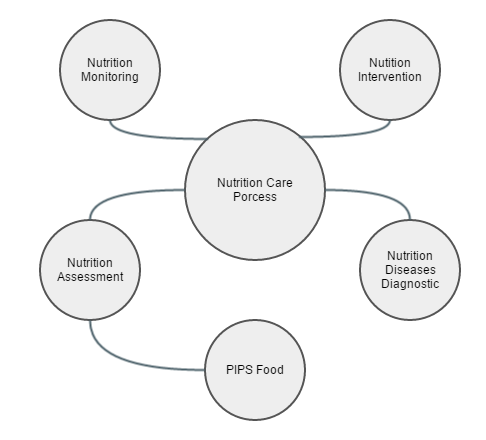


Figure 4-1. Nutrition Care Process Ontology Representation

**Nutrition monitoring** ontology is the first ontology from the four ontologies that compose the nutrition care process ontology. The purpose of this ontology is to represent the information that is relevant for assessing the feeding behavior of an elder. It will store data such as: the older adult daily food intake, the older adult personal data – anthropometric measurements (height, weight change, weight, BMI), biochemical data, health profile (diseases), and other personal data (gender).

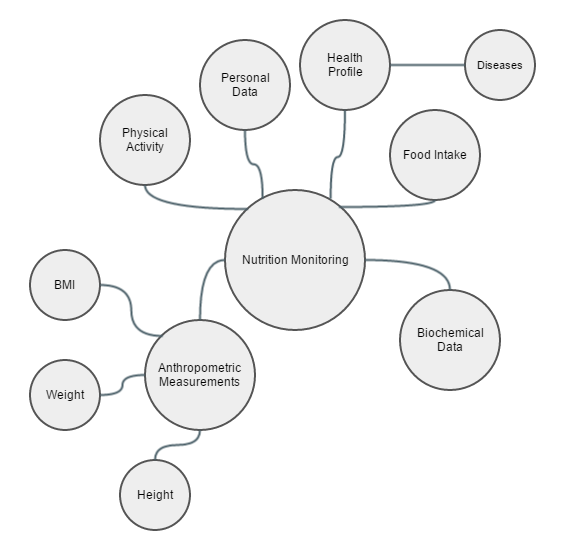


Figure 4-2. Nutrition Monitoring Ontology Representation

**Nutrition assessment** ontology is the second ontology that is contained in the nutrition care process ontology. It will store data about different types of foods and their associated nutritional values. There are two different types of foods. The first type is represented by the combined foods which are based on a recipe. The second type is represented by the basic foods, which are classified according to the PIPS food ontology.

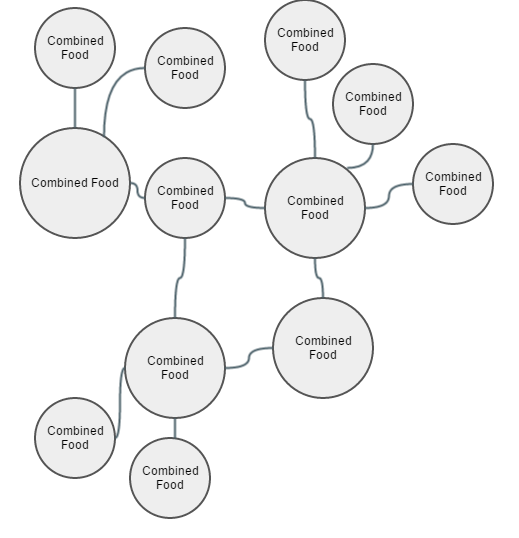


Figure 4-4. Nutrition Assessment Ontology Representation

PIPS Food ontology contains a classification of the foods. There will be different types of foods such as: beverages, egg products, fruits, grain products, meat, milk products, nuts and seeds, oils and fats, sea food, soups and sauces, special nutrition products, sugar products and vegetables. Each of these types will contain different subtypes. This classification is done in order to facilitate the daily work of the elder which consists in choosing which products he consumes during the day. Another important part of this ontology is represented by the food quantities. There will be different types of food quantities such as slice, cup, bowl, dish, pound, piece, can, box, bag, carton, jar and loaf. Each food will have different values for each of these quantity types. Considering the fact that the intention of the application is to be a proof of concept, the values for each of these quantities will be generated as a random number that belongs to an interval (for example a random number between 20 and 250). The reason for choosing random numbers is because many of the foods that will populate the database will consist of different ingredients, each of them having a different density, and the proportion of the ingredients that compose these foods is not known exactly.

Recipes are another part of this ontology. They represent a collection of basic foods in different proportions and each recipe will be associated with a combined food. There will be different types of combined foods: flavored food, regional cuisine food and dish, and each of them will have different subtypes. For example, flavored food will have the following subtypes: sour food, bitter food, spicy food, salted food and sweet food. Regional cuisine food will be of two types: Italian cuisine food and Spanish cuisine food. Finally, there will be three types of dish: desert, main course, and starter dish. Nutrition intake will be evaluated each day and it will contain the amounts of nutritional values the older consumes during the day such as: energy (kcal), fat (g), carbohydrates (g), fatty acids (g), protein (g), potassium (mg), calcium (mg), sodium (mg), vitamin D (ug), alcohol (g), and water (g). These values are very important because malnutrition is prevented by imposing some restrictions on them.

**Nutrition diseases diagnostic** ontology has the main goal to define, classify and represent the nutrition related diseases. There are different types of nutrition diseases such as weight diagnostic (obesity, underweight, overweight and normal weight), malnutrition, imbalance of nutrients, fluid intake diagnostic (excessive fluid intake and suboptimal fluid intake), and energy balance diagnostic (excessive energy intake and suboptimal energy intake).

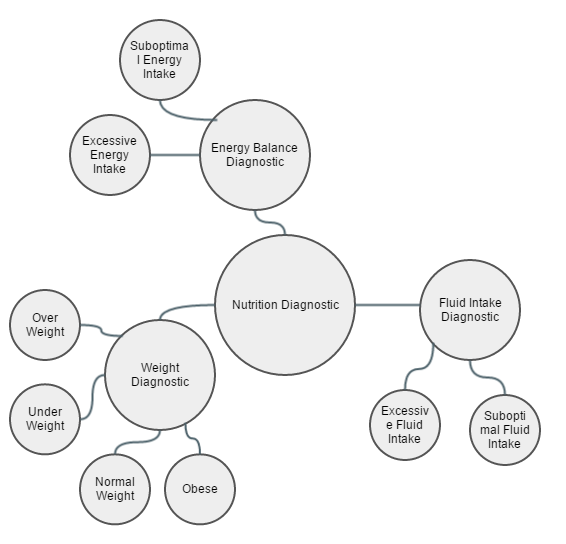


Figure 4-3. Nutrition Diseases Diagnosis Ontology Representation

**Nutrition intervention** ontology is the last ontology from the four ontologies that compose the nutrition care process ontology. Nutrition prescription will define different types of diets such as cardio vascular disease diet and diabetes type 2 diet. Each of these diets will have associated different restrictions that the elder is advised to follow.

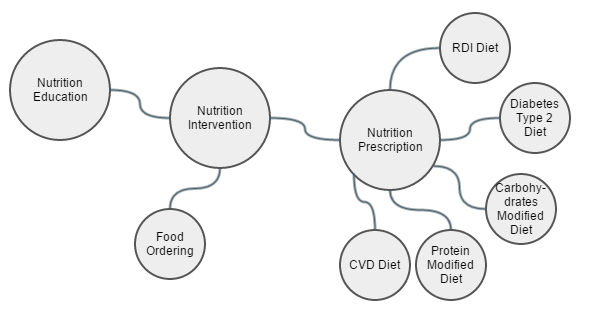


Figure 4-5. Nutrition Intervention Ontology Representation

New data can be inferred from data that already exists by using ontologies, with the help of the reasoner. Inference rules are written in SWRL (Semantic Web Rule Language). Rules are used in order to increase the expressivity of OWL. As described in [7], SWRL rules are similar with the rules written in Prolog or DATALOG languages.

Amongst many others, an objective of the inference rules is to determine nutritional recommendations associated to a specific elder. Each elder has different specific characteristics such as gender, height, weight, which will influence the amount of energy that he should consume during the day. By knowing the amount of energy that the elder should consume during the day, the rest of the nutritional values that he should consume can be determined as different fractions of this value.

## Food Modelling

Collecting data about foods in a structured manner is also one of the objectives. Even though there are a lot of resources on the internet, many of them are not classified very well. The objective is to classify data about foods in a hierarchy so that the elder to be able to select very easily what foods he consumes during the day. The first step is to collect data from the internet. A database for basic food ingredients with the nutritional values of each according to specific measurements must be used. For that, the database provided by **USDA** has been chosen.

### USDA Database

In [8] the USDA is presented. USDA is **National Nutrient Database for Standard Reference** (SR) is the major source of food composition data in the United States. It provides the foundation for most food composition databases in the public and private sectors. As information is updated, new versions of the database are released. This version, Release 27 (SR27), contains data on 8,618 food items and up to 150 food components. It replaces SR26 issued in August 2013.

National Nutrient Data Bank is a repository of information for up to 150 nutrients for about 8,500 foods.  It is made available in the principle database- the [USDA National Nutrient Database for Standard Reference](http://www.ars.usda.gov/Services/docs.htm?docid=8964) (SR).  The data can be accessed online or downloaded for use on A PC.  Programs are available to look up the nutrient content of foods using a Windows PC.

For the health care sector, food composition data are used in dietary therapy of patients in hospitals and community settings, formal and self-directed nutrition education programs for adults and children, and preventive guidance for obstetric, pediatric, and geriatric populations. The data are essential to the calculation of school, hospital, nursing home, and other institutional menus.

USDA is a great source for gathering nutritional information a very large set of food ingredients. The database consists of several sets of data: food descriptions, nutrients, weights and measures, footnotes, and sources of data. The sections below provide details about the information in each.

### USDA Database Structure

Database content is structured as with the following components:

**Food Descriptions** - This includes descriptive information about the food items. A full description (containing the name of the food with relevant characteristics, e.g., raw or cooked, enriched, color) and a short description (containing abbreviations) are provided. Scientific names, common names, manufacturers’ names, amounts of refuse, and refuse descriptions are provided where appropriate. The common name field includes alternative names for a product, e.g., soda or pop, for a carbonated beverage.

**Food Group** - To facilitate data retrieval the food items in SR are organized into food groups. Currently there are 25 food groups, which are listed in the Food Group Description file. Data on ethnic foods are contained in their respective food groups, for example data on plantains, a Latino ethnic food are in food group 9 (Fruit and Fruit Juices), while the Asian foods, miso and natto, are entered in food group 16 (Legumes and Legume Products). Food group 36 (Restaurant Foods) contains foods obtained from casual dining, full service restaurants, Latino restaurants, and Chinese restaurants (not Fast Foods, which are in food group 21). Home prepared items or prepared frozen entrees are included in Food Group 22, Meals, Entrees, and Side dishes. Some food items, such as beverages and rice, though obtained at restaurants are included in their respective food groups.

**Nutrients** - The Nutrient Data file contains mean nutrient values per 100 g of the edible portion of food, along with fields to further describe the mean value.

Weights and Measures - Information is provided on household measures for food items (for example, 1 cup, 1 tablespoon, 1 fruit, 1 leg). Weights are given for edible material without refuse, that is, the weight of an apple without the core or stem, or a chicken leg without the bone, and so forth. The Weight file contains the gram weights and measure descriptions for each food item. This file can be used to calculate nutrient values for food portions from the values provided per 100 g of food. The following formula is used to calculate the nutrient content per household measure:

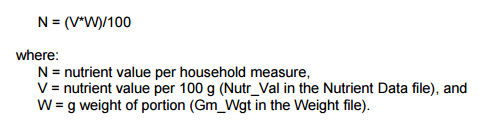


Figure 4-5. Nutrient content per household measure formula [8]

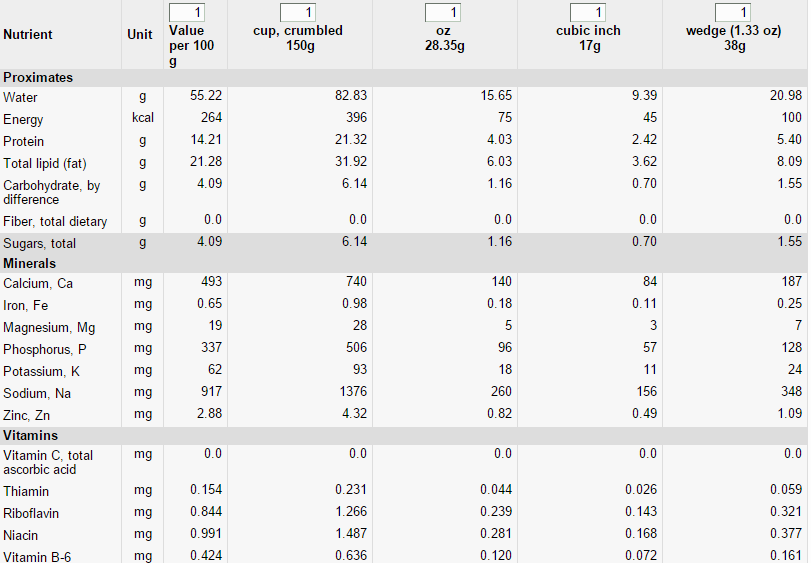


Figure 4-6. Food ingredient nutritional values by measurement for feta cheese[8]

In the Figure 4-6, an example of the nutritional information of a feta cheese food ingredient. It can be seen that USDA contains nutritional values for all nutrients available (not all are shown in the figure) fir several measurement types.

Nutritional information is shown for crumbled cup (~150g), oz (~28.35g), cubic inch (~17g) and wedge(~38g). Other measurements exist for different food ingredients. This feature is very helpful in the case of our problem context. The food packages will consist of dishes that are composed of one or more recipes. These recipes that food ingredients, and as most recipes available nowadays, the measurement unit is not always in grams. For example a recipe could have the following description: 4 skinless, boneless chicken breast halves, 1/4 teaspoon salt, 1 pinch ground black pepper to taste, 1 pinch ground black pepper to taste, 1/2 cup low-sodium chicken broth, 1/4 cup white wine, 2 tablespoons honey and 2 tablespoons whole-grain mustard. It can be seen that the necessary steps for integrating this recipe description into the database model can be quite difficult. If the measurement unit were to be only in grams, at each ingredient transformations should be made which can prove to be quite time consuming unless automatized. USDA however solve this problem by offering nutritional values for other measurement units such as cup, table spoon, small spoon etc.

The USDA nutrient database has the following schema.

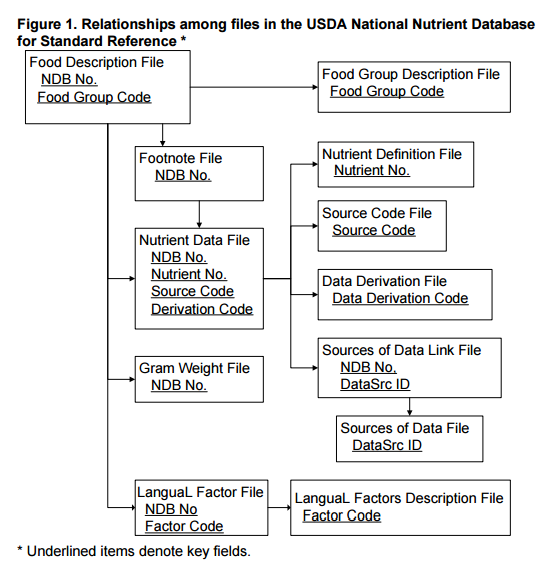


Figure 4-7. USDA database schema [8]

## Metaheuristics

For generating food menu recommendations that would fit best to the user’s profile, medical prescription, allergies, preferences and food provider costs a bio-inspired metaheuristic will be used due to the fact that the search space is very large, because of the large number of food packages, thus this problem will be modelled as a search optimization problem for finding the optimal solution.

A definition of a metaheuristic is that it “is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. Metaheuristics sample a set of solutions which is too large to be completely sampled. Metaheuristics may make few assumptions about the optimization problem being solved, and so they may be usable for a variety of problems.” [10]

According to [11], mathematically speaking, most optimization problems can be written in the generic form

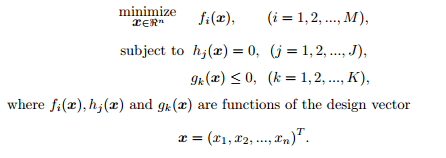


Figure 4-7. Mathematical representation of optimization problems [11]

This of course is an example of a minimization problem, however it can also be formulated, if the objectives are the opposite in case the values is better if it is higher, as a maximization problem.

The mathematical entities are as follows:

* The **x vector** represents the vector of decision variables, and they can be real continuous, discrete or the mixture of these two.
* The space formed by the objective function values is called the solution space and the space spanned by the decision variables is called the design space or search space **Rn**.
* The functions fi(x) where i=1,2,..,M are called the **objective functions**.
* The equalities for hj and inequalities for gk are called **constraints**.

Several criteria are identified by which the optimization can be classified and are presented as follows:

* Number of objectives – the optimization problem can have but one objective (M=1) or more (M>1)
* Number of constraints – the optimization problem can have no constraints (J=K=0) or can have constraints (J+K>0)
* Function – the optimization problem can have its objective function either linear or non-linear
* Shape (landscape) – the objective functions that are not linear can be unimodal (only one global optimal solution) or multimodal(more global optimal solutions)
* Type of adjustable decision variables – the adjustable decision variables xi can be discrete, continuous or mixed
* Determinism – the system can be either deterministic or stochastic

As opposed to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. [10] Many metaheuristics implement some form of stochastic optimization, so that the solution found is dependent on the set of random variables generated. By searching over a large set of feasible solutions, metaheuristics can often find good solutions with less computational effort than algorithms, iterative methods, or simple heuristics. As such they are useful approaches for optimization problems.

A way of characterizing the type of search strategy. [10] One type of search strategy is an improvement on simple local search algorithms; Metaheuristics of this type include simulated annealing, tabu search, iterated local search variable neighborhood search and GRASP. The other type of search strategy has a learning component to the search; metaheuristics of this type include ant colony optimization, evolutionary computing and genetic algorithms.

Another way of classifying is single solution vs population-based searches. [10] Single solution approaches focus on modifying and improving a single candidate solution; single solution metaheuristics include simulated annealing, iterated local search, variable neighborhood search and guided local search. [12] Population-based approaches maintain and improve multiple candidate solutions, often using population characteristics to guide the search; population based metaheuristics include evolutionary computation, genetic algorithms and particle swarm optimization.

Swarm Intelligence is another category of metaheuristics which is collective behavior of decentralized, self-organized agents in a population or swarm.

In addition to the serial algorithms above, there are **hybrid** and **parallel metaheuristics.** [12] A **hybrid metaheuristic** is one which combines a metaheuristic with other optimization approaches, such as algorithms from mathematical programming, constraint programming and machine learning. Both components of a hybrid metaheuristic may run concurrently and exchange information to guide the search. A parallel metaheuristic is one which uses the techniques of parallel programming to run multiple metaheuristic searches in parallel; these may range from simple distributed schemes to concurrent search runs that interact to improve the overall solution.

The use of metaheuristics is for combinatorial optimization in which an optimal solution is sought over a discrete search space. An example problem is the travelling salesman problem where the search space of the problem increases, which makes an exhaustive search for the optimal solution infeasible. [11] Additionally, multidimensional combinatorial problems, including most design problems in engineering such as form-finding and behavior-finding, suffer from the curse of dimensionality, which also makes them infeasible for exhaustive search or analytical methods. Popular metaheuristics for combinatorial problems include simulated annealing, genetic algorithms, scatter search and tabu search.

For solving an optimization problem, efficient search or optimization algorithms are needed. There are many optimization algorithms which can be classified in many ways, depending on the focus and characteristics.

Optimization can be classified into gradient-based algorithms and derivative-free or gradient-free algorithms. [10]Gradient-based algorithms such as **hill-climbing** use derivative information, and they are often very efficient. Derivative-free algorithms do not use any derivative information but the values of the function itself. Some functions may have discontinuities or it may be expensive to calculate derivatives accurately, and thus derivative-free algorithms.

[10]In another perspective, optimization algorithms can be classified into trajectory-based and population-based. A trajectory-based algorithm typically uses a single agent or one solution at a time, which will trace out a path as the iterations continue. Hill-climbing is trajectory-based, and it links the starting point with the final point via a piecewise zigzag path. Another important example is simulated annealing which is a widely used metaheuristic algorithm.

[10]If an algorithm works in a mechanical deterministic manner without any random nature, it is called deterministic. For such an algorithm, it will reach the same final solution if we start with the same initial point. Hill-climbing and downhill simplex are good examples of deterministic algorithms. On the other hand, if there is some randomness in the algorithm, the algorithm will usually reach a different point every time the algorithm is executed, even though the same initial point is used.

A basis for algorithm classification can also be search capabilities. In this case, algorithms can be divided into local and global search algorithms. Local search algorithms typically converge towards a local optimum, not necessarily (often not) the global optimum, and such an algorithm is often deterministic and has no ability to escape from local optima. Simple hill-climbing is such an example. On the other hand, for global optimization, local search algorithms are not suitable, and global search algorithms should be used. Modern metaheuristic algorithms in most cases tend to be suitable for global optimization, though not always successful or efficient. A simple strategy such as hill-climbing with random restarts can turn a local search algorithm into an algorithm with global search capability. In essence, randomization is an efficient component for global search algorithms.

Algorithms may not exactly fit into each category. It can be a so-called mixed type or hybrid, which uses some combination of deterministic components with randomness, or combines one algorithm with another so as to design more efficient algorithms.

## Hill-Climbing

Hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

For finding local optimum (a solution that cannot be improved by considering a neighboring configuration) but it is not necessarily guaranteed to find the best possible solution (the global optimum) out of all possible solutions (the search space) hill climbing is an appropriate choice.  In convex problems, hill-climbing *is* optimal. Examples of algorithms that solve convex problems by hill-climbing include the simplex algorithm for linear programming and binary search. [13]

Local optima’s only characteristic that can be are guaranteed can be cured by using restarts (repeated local search), or more complex schemes based on iterations, like iterated local search, on memory, like reactive search optimization and tabu search, or memory-less stochastic modifications, like simulated annealing.

The relative simplicity of the algorithm makes it a popular first choice amongst optimizing algorithms. It is used widely in artificial intelligence, for reaching a goal state from a starting node. Choice of next node and starting node can be varied to give a list of related algorithms. Although more advanced algorithms such as simulated annealing or tabu search may give better results, in some situations hill climbing works just as well. Hill climbing can often produce a better result than other algorithms when the amount of time available to perform a search is limited, such as with real-time systems. It is an anytime algorithm: it can return a valid solution even if it's interrupted at any time before it ends.

The variants of hill climbing will be presented:

* In **simple hill climbing**, the first closer node is chosen, whereas in **steepest ascent hill climbing** all successors are compared and the closest to the solution is chosen. Both forms fail if there is no closer node, which may happen if there are local maxima in the search space which are not solutions. Steepest ascent hill climbing is similar to best-first search, which tries all possible extensions of the current path instead of only one.
* **Stochastic hill climbing** does not examine all neighbors before deciding how to move. Rather, it selects a neighbor at random, and decides (based on the amount of improvement in that neighbor) whether to move to that neighbor or to examine another. Coordinate descent does a line search along one coordinate direction at the current point in each iteration. Some versions of coordinate descent randomly pick a different coordinate direction each iteration.
* **Random-restart hill climbing** is a meta-algorithm built on top of the hill climbing algorithm. It is also known as **Shotgun hill climbing**. It iteratively does hill-climbing, each time with a random initial condition x0. The best xm is kept: if a new run of hill climbing produces a better x_m than the stored state, it replaces the stored state. Random-restart hill climbing is a surprisingly effective algorithm in many cases. It turns out that it is often better to spend CPU time exploring the space, than carefully optimizing from an initial condition.

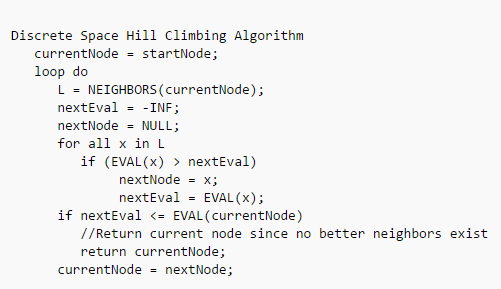


Figure 4-8. Hill Climbing Pseudocode [13]

# Hybrid Technique for Generating Food Menu Recommendations using the Cuckoo Search Algorithm

This chapter models the problem of generating healthy menu recommendations for older adults as an optimization problem and proposes a hybrid Cuckoo Search Optimization method for solving this problem. The method hybridizes the state of the art Cuckoo Search meta-heuristic by injecting strategies inspired from Genetic Algorithms, Hill Climbing and Tabu Search into the steps that generate new solutions of the optimization problem. Experiments have been conducted on several hybridization configurations to identify the most appropriate hybridization that leads to the healthy menu recommendation that best satisfies the older adult’s diet recommended by the nutritionist, its culinary preferences and time and price constraints.

The proposed optimization method hybridizes the state of the art Cuckoo Search Optimization meta-heuristic with strategies borrowed from Genetic Algorithms, Path Relinking, Hill Climbing, Simulated Annealing, and Tabu Search to identify the most appropriate hybridization configuration that provides the best result in a short time. The optimization method is further integrated in the ordering system that we propose. In what follows, the paper will mainly focus on presenting the optimization method.

## Cuckoo Breeding Behavior

In [14], Cuckoo Search algorithms is presented based on the breeding behavior of cuckoo birds. Cuckoo are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the Guira cuckoos lay their eggs in communal nests, though they may remove others’ eggs to increase the hatching probability of their own eggs (Payne et al 2005). Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species. If a host bird discovers the eggs are not its own, it will either throw these alien eggs away or simply abandons its nest and builds a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in color and pattern of the eggs of a few chosen host species. This reduces the probability of their eggs being abandoned and thus increases their breeding factor. Parasitic cuckoos often choose a nest where the host bird just laid its own eggs.

## Cuckoo Search

Cuckoo Search idealized the cuckoo breeding behavior therefore various optimization problems can adapt this approach. It has proven to be better than other metaheuristic algorithms in applications.

Each egg in a nest represents a food package that represents a day’s meal such as breakfast, lunch, dinner, snack1 or snack2. A cuckoo egg represents a new food package. The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solutions in the nests. Each nest will be composed of exactly five eggs, which represent all the 5 meals of a day, thus making the set of eggs equivalent to the solution of our problem. The solution of our problem is in fact the recommended set of a day’s meals.

For a clearer view, the algorithm’s representation of components is presented in the Figure 5-1.

|  |  |  |
| --- | --- | --- |
| **Biological Concepts [1]** | **CS Concepts [1]** | **Concepts from our Optimization Problem** |
| Egg from a nest | Food Package | A food package represents a day’s meal. A food package can be of type breakfast, lunch, dinner, snack1 or snack2. |
| Egg from a cuckoo | Food Package | The cuckoo’s eggs simply represents a new food package |
| Nest | Solution | The nest is the place holder for the eggs. Each nest contains exactly five eggs(food packages) which represent the 5 meals from a day. Thus a nest represents the placeholder for the solution that is composed of 5 eggs. |
| Best Nest | Solution | The nest with the best solution, i.e. the best fitness value amongst all the other nests |
| Cuckoo | New Solution Generator based on crossover | A cuckoo will generate 5 eggs that it will place in a nest, in an attempt to improve the nest’s fitness value |
| Breeding | Crossover/heuristic functions applied to generate new solutions | Crossover strategies applied between a combination of food packages and the current optimal combination of food packages |
| Egg replacement in a nest by a cuckoo | Recplacing an egg from the nest with the cuckoo’s egg. | Some eggs of the cuckoo, or maybel all, will be different from the |
| Building a new nest | Generate Random Solution / Improved Solution Based on Old One | Hill Climbing, Tabu Search-based strategies for improving a combination of food packages |

Table 5-1. Algorithm representation 1

Cuckoo Search is based on three idealized rules:

1. Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest;
2. The best nests with high quality of eggs will carry over to the next generation;
3. The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability pa ∈ (0,1). Discovering operate on some set of worst nests, and discovered solutions dumped from farther calculations.

### Cuckoo Search Solution Representation

In this case, a solution of the optimization problem is formally represented as follows:



where *fPb*, *fPs1*, *fPl*, *fPs2* and *fPd* are the food packages selected for breakfast, the first snack, lunch, the second snack, and dinner respectively. Each food package can have a starter, main course and/or desert dish.

The solution is a meal for an entire day. In principle, it contains breakfast, lunch, dinner and two snacks meals which are the five main components of a solution.

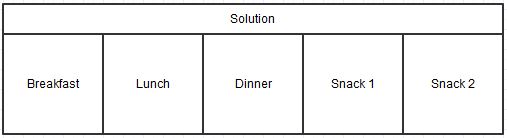


Figure 5-2. Solution representation 1

A food package is formally represented as:



where *fIi* is a food item (e.g. noodle soup) and *m* is the number of food items part of the food package, each food item being classified as either a starter, main course, or a dessert. Additionally, each food item has a recipe associated as well as a set of values corresponding to the nutrients that are being considered.

Each of these can have a starter, main course and/or desert dish. The following illustration shows the relationships between the meals and the dishes.

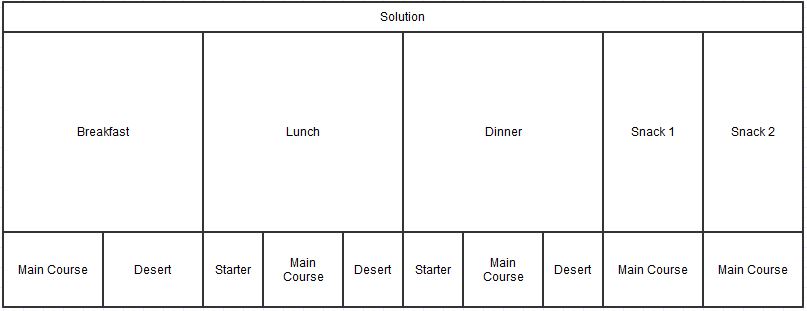


Figure 5-3. Solution representation with dishes

### Cuckoo Search Fitness Function

This section contains work made by my collegue Cristian Prigoana, however the fitness function must be included because it is absolutely relevant to the algorithm context.

The fitness function is an evaluation function that rates each state or solution. Usually, it returns a higher value for better states, and of course, lower for worse states.

For the diet of a person, the fitness function models the quality of the menu, in terms of following factors:

1. Nutrients
   * Kilocalories
   * Macro-nutrients: carbohydrates, proteins, fats,
   * Micro-nutrients: vitamin A, B, C, D, calcium, iron, sodium
2. Deviation of the quantities of the previously mentioned items from the desired ones;
3. Doctor's prescription;
4. Patient's preferences;
5. Cost and delivery time of the food package.

As the doctor or the patient wishes to control the meals on different levels, a fitness function for every level is computed. The evaluation of a solution, which in our case is a meal for a day is done on three different levels and then combined to form the final mark.

The first is the nutrients' and kilocalories' level and deals with evaluating the meals from these points of view. If the number of calories and the quantities of nutrients that they contain are close to the ideal (desired) ones, then this function will have a great value. The ideal values are given are computed based on the user’s profile (gender, age, physical activity etc.) and can be overwritten by the doctor.

The second level deals with food items. It takes into consideration inclusion or exclusion of items to meet the requirements or preferences of the doctor or patient. If some undesired food item exist in a food package, then it is marked lower.

Next, the third level evaluates a solution based on the cost and delivery time of the food package by the food provider.

Finally, the three functions will be combined, in order to give the final mark to the solution. In order to properly define all the mentioned functions, the following aspects are explained first:

* Ideal value
* Weight
* Error function

|  |  |
| --- | --- |
| **Level** | **Evaluation by** |
| **1** | Deviation of kilocalories, macro and micro-nutrients from the desired values |
| **2** | Existence/absence of required/restricted or (un)desired food items |
| **3** | Cost and delivery time |

Table 5-4. Fitness function levels

Because the fitness value of a solution depends on more than one criterion, i.e. on different levels (see **Error! Reference source not found.** level), the fitness function template will be applied for each criterion in particular; there are 3 criteria (levels) hence applied 3 times and then combined. The template function consists of a weighted average between all the components of a level and are applied in this chapter. Let *c* be a component from the set *C* of components, *w(c)* the weight of component *c*, real*(c)* the quantity of *c*, *ideal(c)* the ideal quantity of *c*, *E(q(c), ideal(c))* the error function which penalizes any deviation of *q(c)* from the ideal value of *c* and *X* the set of solutions. The fitness function is **maximized,** i.e. the ideal or optimal value is 1, and 0 is the worst.

#### Top level fitness function

The top level fitness function is the one that gives the final mark to a solution by combining the previously presented smaller-level fitness functions. The fitness function template is here also applied, therefore, it is just a weighted average between level 1, 2 and 3 fitness functions.



Where *fitness1* evaluates the quality of a solution from the nutritionist’s recommended diet perspective, *fitness2* evaluates the quality of a solution from the older adult’s culinary preferences, and *fitness3* evaluates the quality of a solution from the older adult’s cost and delivery time constraints. The definitions of all three fitness function components, i.e. *fitness1*, *fitness2* and *fitness3* rely on the following aspects:

#### Fitness function level 1: nutrients

The level 1 fitness function takes a solution as input and provides a value between 0 and 1 to evaluate it from the nutrients’ point of view. It assures that each nutrient comes in the appropriate quantity. The previously mentioned set *C* of components is in this case the set of nutrients, therefore if *N* is the set of nutrients, then:



where: *w*1(*n*) represents the value of the weight associated to the nutrient *n*, *N* is the set of nutrients associated to the food packages part of the solution *sol*, *ideal1*(*n*) represents the ideal value of the nutrient *n*, *real*(*n*) represents the actual value of the nutrient *n* in the current solution *sol* being evaluated, and *E*1 is the error function defined for the fitness component *fitness1 .* The functions w, ideal and E1 are described more thoroughly in my colleague’s, Cristian Prigoana, diploma thesis.

#### Fitness function level 2: ingredients

The level 2 fitness function takes a solution as input and provides a value between 0 and 1 to evaluate it from the ingredients’ point of view. It checks if each ingredient in the *like* *list* is present in the meal and each ingredient in the *dislike list* is not present in the meal (see **Error! Reference source not found.**). In this case, the set *C* of components is the set *F* of food items present in the solution meal union the *specified list*



where: *w*2(*i*) represents the value of the weight associated to the ingredient *i*, *real*(*i*) represents the ingredient *i*, *ideal2*(*i*) indicates whether the ingredient *i* is among the ones preferred by the older adults or not, and *E*2 is the error function defined for the fitness component *fitness2*.

#### Fitness function level 3: cost and delivery time

The top level fitness function is the one that gives the final mark to a solution by combining the previously presented smaller-level fitness functions. The fitness function template is here also applied, therefore, it is just a weighted average between level 1, 2 and 3 fitness functions.



where *real*(*cost*, *sol*) represents the cost of the food menu recommendation *sol*, *ideal*(*cost*, *sol*) represents the maximum cost desired by the older adult, *real*(*time*, *sol*) represents the delivery time of the food menu recommendation *sol*, *ideal*(*time*, *sol*) represents the maximum delivery time accepted by the older adult, and *E*3 is the error function defined for the fitness component *fitness3* as follows:



### The Crossover based creation of new solution

The Crossover-based broods creation strategy applies the crossover operation between the queen and the drone at randomly selected crossover points. A crossover point may be one of the main components of the solution: breakfast, lunch, dinner, snack 1 or snack 2. Thus, a new solution (brood) is created by randomly selecting components between the queen and the drone. The process is repeated multiple times to provide a set of broods.

proceuure Crossover-Based-Broods-Creation-Strategy (solution1, solution2): returns New Solution

newSolution = new solution;

Breakfast newSolBr = newRandomBoolean() == true ? solution

newSolution.setBreakfast(

foreach meal in {Breakfast, Lunch, Dinner, Snack1, Snack2}

if( newRandomBoolean() == true)

newSolution.setMeal(solution1.meal)

else

newSolution.setMeal(solution2.meal)

return newSolution

There are 5 possible crossover points. Each meal can be considered a crossover candidate between the two solutions. The number of crossover points are chosen randomly, as are the selected crossover points.

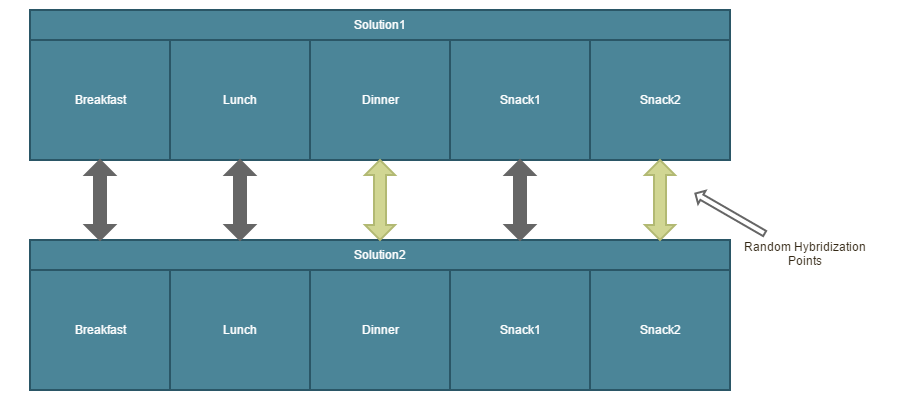


Figure 5-5. Solution Crossover with Random Selection Points

(golden arrows crossover points)

### Hill Climbing - hybridization point in CS

The Hill Climbing heuristic searches for better solutions in the neighborhood of the current best solution. If one is found, it replaces the current best solution. The solutions that are worse than the current one are ignored, as in every greedy approach. This step is repeated for a certain number of times or until a local maximum is found.

Procedure hillClimbing(solution): returns a solution

local: timeStep=0, localMaximumFound=false,Solution = brood

local: limit, neighborhoodSize

while(timeStep < timeLimit and localMaximumFound == false)

timeStep += 1

localMaximumFound = true

N = getHeighborhood(bestSolution)

for each solution in N

fitness(solution) > fitness(bestSolution)

bestSolution = solution

localMaximumFound = false

return bestSolution

## Cuckoo Search Hybrid Versions

The Cuckoo Search Optimization Algorithm (see CS Algorithm) identifies the optimal combination of food packages for the meals of a day by processing a set of semantic descriptions of food packages provided by different providers. The algorithm takes as input the following parameters: (*i*) *Rep* – repository of semantic descriptions of food packages, (*ii*) *nestNumber* – the size of the initial population, (*iii*) *maxIteraions* – maximum number of iterations, (*iv*) pa–probability factor that the host bird discovers the cuckoo eggs

The algorithm consists of an initialization and an iterative stage. In the initialization stage the initial population of individuals is randomly generated, from which the best solution will be extracted as the best solution in the population, while the rest of the solutions are labeled as nest solutions that will further attempt to improve the best solution

**CS Algorithm**

**Inputs:** *nestNumber, maxIterations, pa, Rep*

**Output:** solopt

**Begin**

*nestPopulation =* ***Random\_Generate(****Rep, nestNumber)*

Solbest = **Extract\_Best\_Solution(***nestPopulation)*

*i=0*

**while(***i<maxIterations****) do***

**for** *j=0*  **to**  *nestPopulation*

*cuckoo[j] =* **Cuckoo\_Generation\_Strategy(***nest[j]***)**

**if** *cuckoo[j].fitnessValue > nest[j].fitnessValue*

*nest[j] = cuckoo[j]*

**end for**

**for** *j=0*  **to**  *nestPopulation*

**if** *randomDouble > pa // randomDouble is a random number float 0 and 1*

*nest[j] =* **NewNest\_Generation\_Strategy*(****nest[j])*

**end for**

Solbest = **Extract\_Best\_Solution(***nestPopulation)*

**end while**

**return** Solbest

**End**

**NOTE:** The shadowed lines are hybridization points for improvement of results

In the Cuckoo Search algorithm two hybridization points have been inserted. By applying different new solution generation heuristics the results are improved. As a result of combining the hybridization components with the core component presented 4 hybrid versions have been developed.

|  |  |  |
| --- | --- | --- |
| **Cuckoo Generation Strategy** | **New Nest Generation Strategy** | **Acronym** |
| Random Levy flight solution generation | Random Levy flight solution generation | VersionRLRL |
| Uniform Crossover between current nest and best solution | Uniform Crossover between current nest and best solution | VersionUCUC |
| Uniform Crossover between current nest and best solution | Hill Climbing heuristic applied on old nest | VersionUCHC |
| Tabu Search heuristic applied on current nest | Hill Climbing heuristic applied on old nest | VersionTSHC |

Table 5-1. Hybrid versions of Cuckoo Search

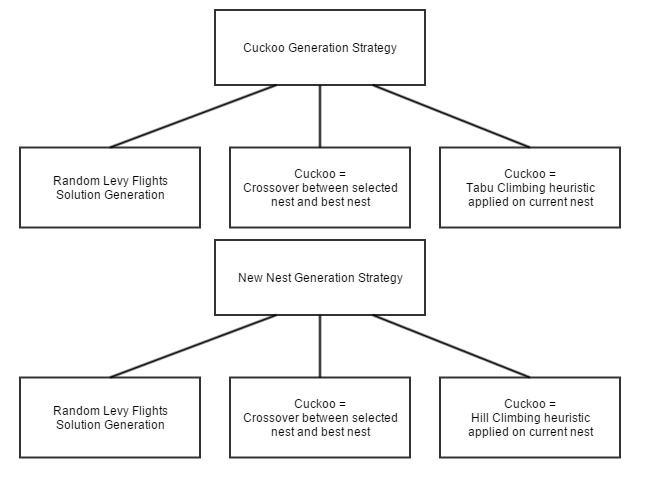


Figure 5-6. Hybridization Points - Strategies

The Cuckoo Search flow diagram is presented in Figure 5-7.

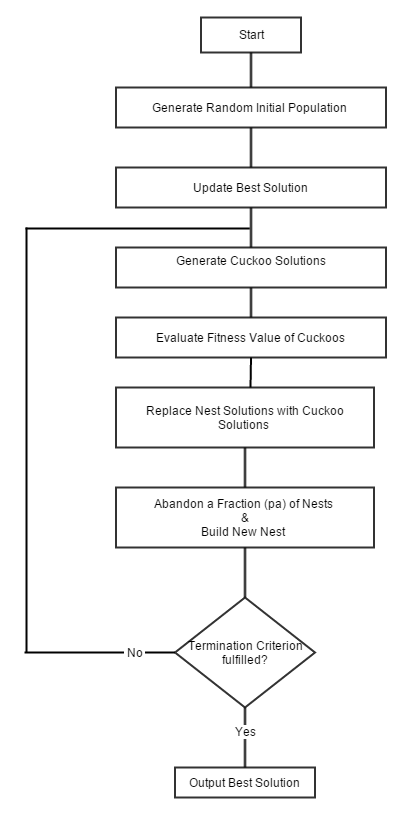


Figure 5-7. Cuckoo Search algorithm flow diagram

# Detailed Design and Implementation

This chapter described the application prototype that was developed for testing the hybrid technique. This permits the visualisation of the recommended food menus generated and also offers a module for exporting results into files for enableing business logic, to view charts and statistics about the results.

## Design of Experimental Prototype

Analysis and design is made for this application and the output is several types of UML diagrams for the description of the system

### Conceptual Architecture

To validate and harness the hybrid Cuckoo Search optimization technique an experimental prototype has been designed.

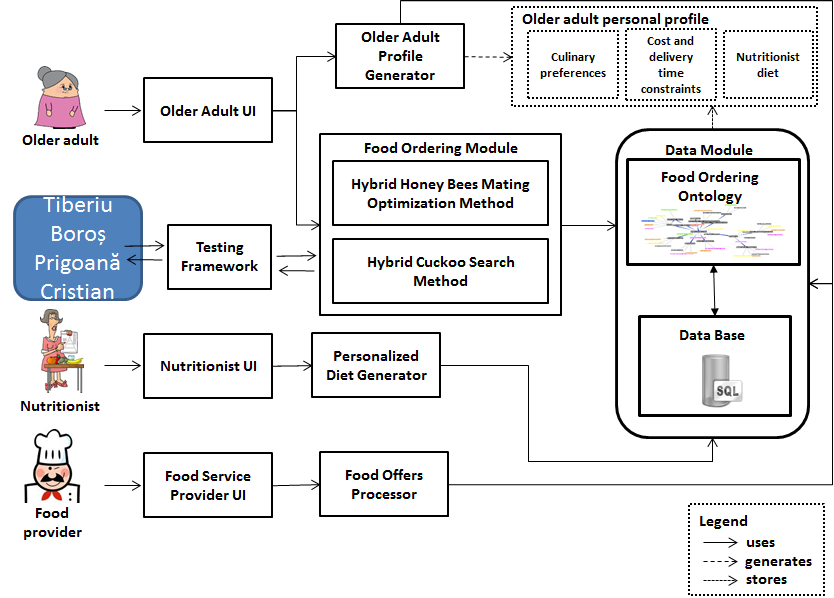


Figure 6-1. High level architecture of DIET4Elders system

As it can be seen in the system architecture, the entities involved with the system comprise of the older adult, the nutritionist, the food provider and the configuration admin. For this experimental prototype, the nutritionsts’ entity and the food providers’ entities have been stubbed. There is no interface in the system for the food provider and nutritionist because the food packages have been generated with random food provider data.

The hybrid algorithms developed by me and my collegue Cristian Prigoana, Cuckoo Search and Honey Bee Mating Optimization, are placed in the Food Ordering Module. This module is connected to the ontology database layer and the MySQL database

### Database Model

The MySQL database model has been developed in order to harness all the information supplied by the USDA National Nutrient Database. As described in Chapter 5, the database provides information about basic food ingredients and their nutritional information per measument. Many types of measurements are available and they lie in the *weight* table. Each recipe contains one or more basic food ingredients. This is modelled in the *recipe\_food\_relation* table, which contains a foreign key to the recipeId and the nutrient identification number as well as the measurement type and quantity.

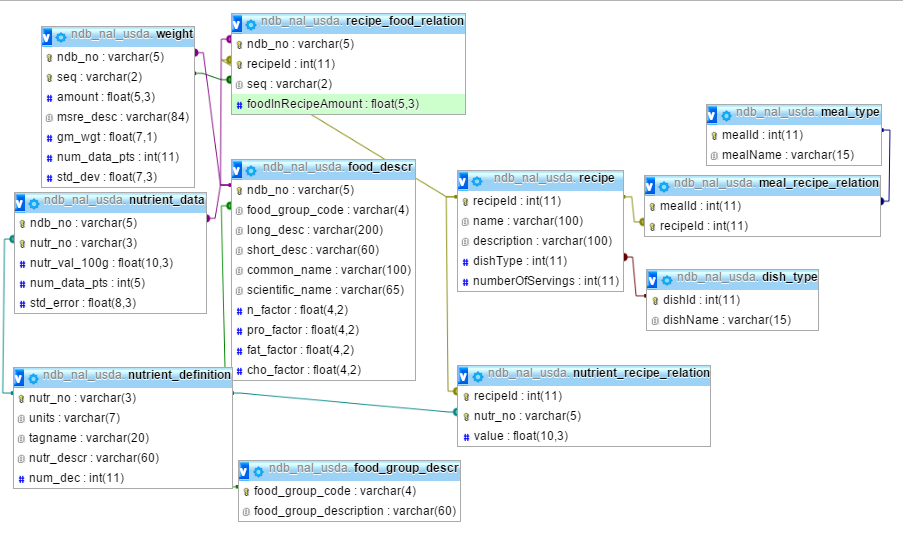


Figure 6-2. Database model containing food ingredient nutrients with data gathered from USDA

The second part of the database models the following entities:

* Dish – this entity, as its name already suggests, is a dish that has a certain recipe, a dish type (starter, main course or dessert) and a meal type the dish can belong to; a dish can belong to more than one meal type; the meal types are breakfast, lunch, dinner and snack
* Meal Variant – a meal variant contains one, two or three dishes depending on the meal type;
  + Snack – contains only one dish
  + Breakfast – contains two dishes: main course and dessert
  + Lunch – contains three dishes: starter, main course and dessert
  + Dinner – contains two dishes: main course and dessert
* Menu – contains all the five meals of the day: breakfast, lunch, dinner and two snacks
* Food Provider – local food vendor that provide the food packages
* Geographical Area – geographical area of the food provider

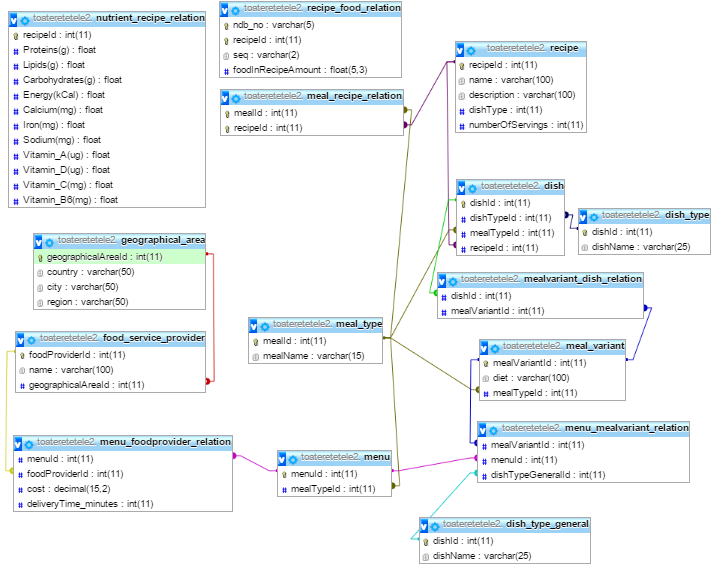


Figure 6-3. Database model continuation. This part models the food providers, menus, meals, dishes, recipes with nutrition data from the former figure 6-3

### Ontology Classes

Each ontology is written in a separate .owl file, by using Protégé. There will be six .owl files, and their names will be presented below:

• food.owl

• nutritionassessment.owl

• nutritioncareprocess.owl

• nutritiondiagnostic.owl

• nutritionintervention.owl

• nutritionmonitoring.owl

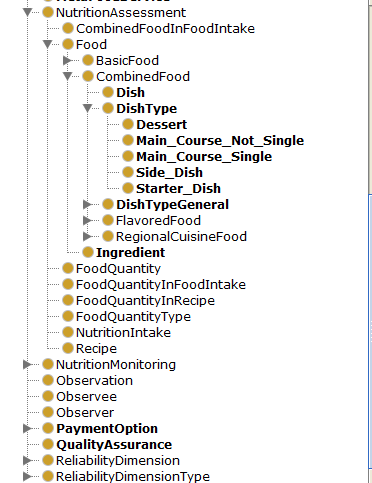


Figure 6-4. Ontology model part1

The hierarchy of the ontologies is presented below:

• **Nutrition care process ontology.** This ontology is composed of four ontologies: nutrition assessment, nutrition diagnostic, nutrition intervention and nutrition monitoring

• **Nutrition assessment ontology.** This ontology holds information about food recipes, dishes, dish types and basic food ingredients.

• **PIPS food ontology.** This ontology contains a hierarchy of foods categorized by type (egg products, fruits, grain products, etc )

• **Nutrition intervention ontology.** This ontology contains data about the nutrition prescription, nutrition education and food ordering.

• **Nutrition monitoring ontology.** This ontology represents the care process ontology and contains information specific for an elder such as: anthropometric measurements, biochemical data, food intake, health profile, mediteranean diet adherence, p[ersonal data and physical activity

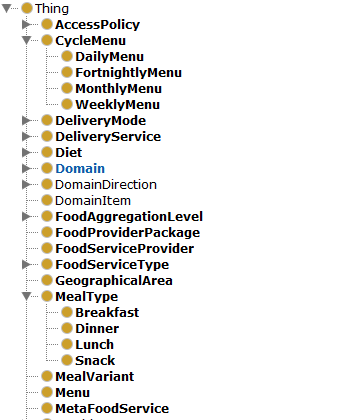


Figure 6-5. Ontology model part 2 1

### Use Case Model

Currently this prototype does not support many interactions because of the purpose of was to harness the hybrid algorithms by building a testing framework.

The use case model of this system contains a single use case: generate recommended food menu. This prototype does not contain authentication, because the user profile has been stubbed, and can be configurable from a configuration panel. The actor of this use case is a regular user. There are no **preconditions** for this use case. The following steps are identified.

**Main Success Scenario (Basic Flow)**

1. The user starts the system
2. The user runs the seleced algorithm to generate recommended food menus

**Extensions (Alternate Flows)**

\*a At any time the user can modify configuration settings (NOTE: at system startup an initial setup of the configuration settings already exists because of some stubs, but can be modified)

The configuration setting actions could be the following

1. The user adds/removes food ingredients that he/she would want in the food menu recommentation or leave empty
2. The user adds/removes food ingredients that he/she would not want in the food menu recommentation or leave empty
3. The user can modify his/her profile (weight, height, age, gender, physical activity factor)
4. The user can modify algorithm choice (Cuckoo Search or Honey Bee Mating Optimization)
5. The user can modifiy algorithm adjustable variables and heuristics adjustable variables
6. The user modifies the configuration file

### Activity Diagram

The flow of this prototype is more clear with the aid of an activity diagram, from the start point to the end point. The activity diagram is presented in Figure 6-7.

In the first part the after the system is started the food packages are loaded either by extracting them from the ontology database either by extracting them from the local cache files if they exist and are up-to-date.

Afterwards the GUI is displayed. The user can edit the configuration setup. The configuration settings for the user profile, the selected algorithm and the adjustable variables for each algorithm and also the adjustable variables for each heuristic is available and open for custom editing.

The user has the possibility to:

* run the algorithm with the current setup or
* run the algorithm with the configurations from the configuration file and exporting the results to CSV files for evaluating the results and chart visuals

And at last, the system will output the results either on GUI if the first option was selected, either in CSV files.

The user can repeat these steps as many times as desired.

### Deployment Diagram

The deployment diagram will describe the system from the physical perspective.

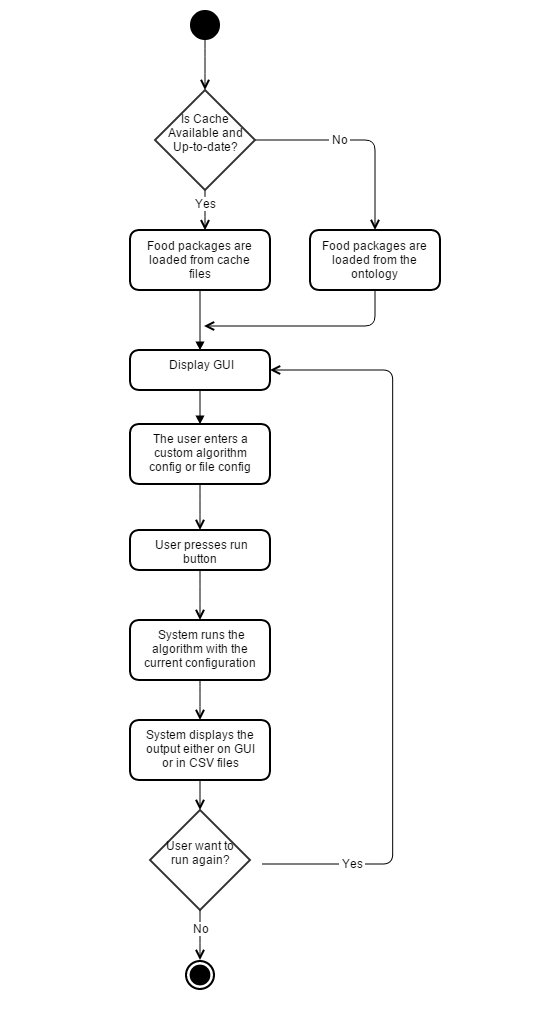


Figure 6-6. Activity diagram of prototype

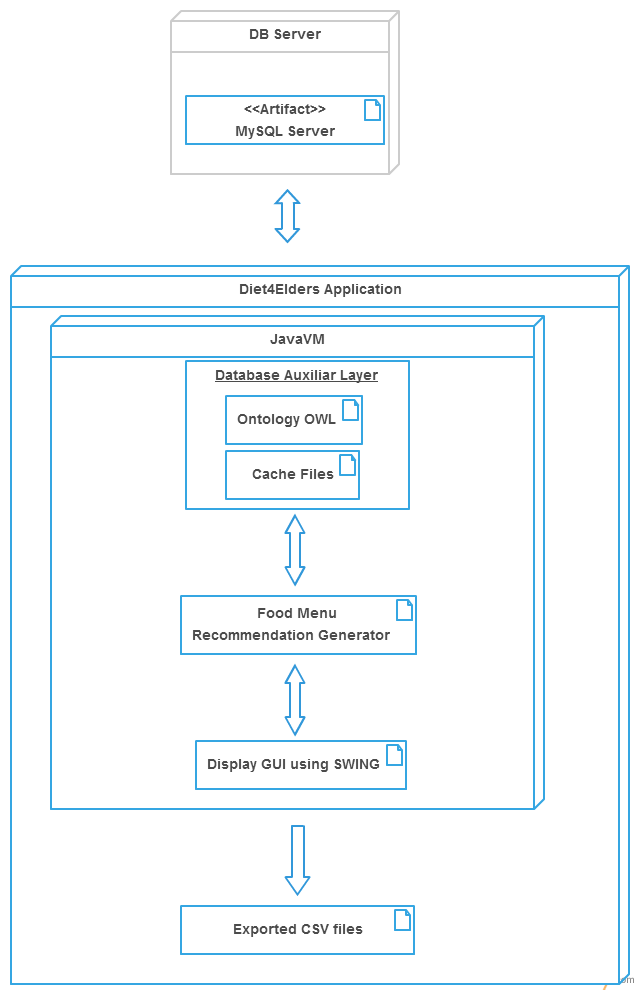


Figure 6-7. Deployment Diagram

### Modules Architecture

A more detailed architecture of the Testing Framework modules designed in this license thesis is presented as follows:

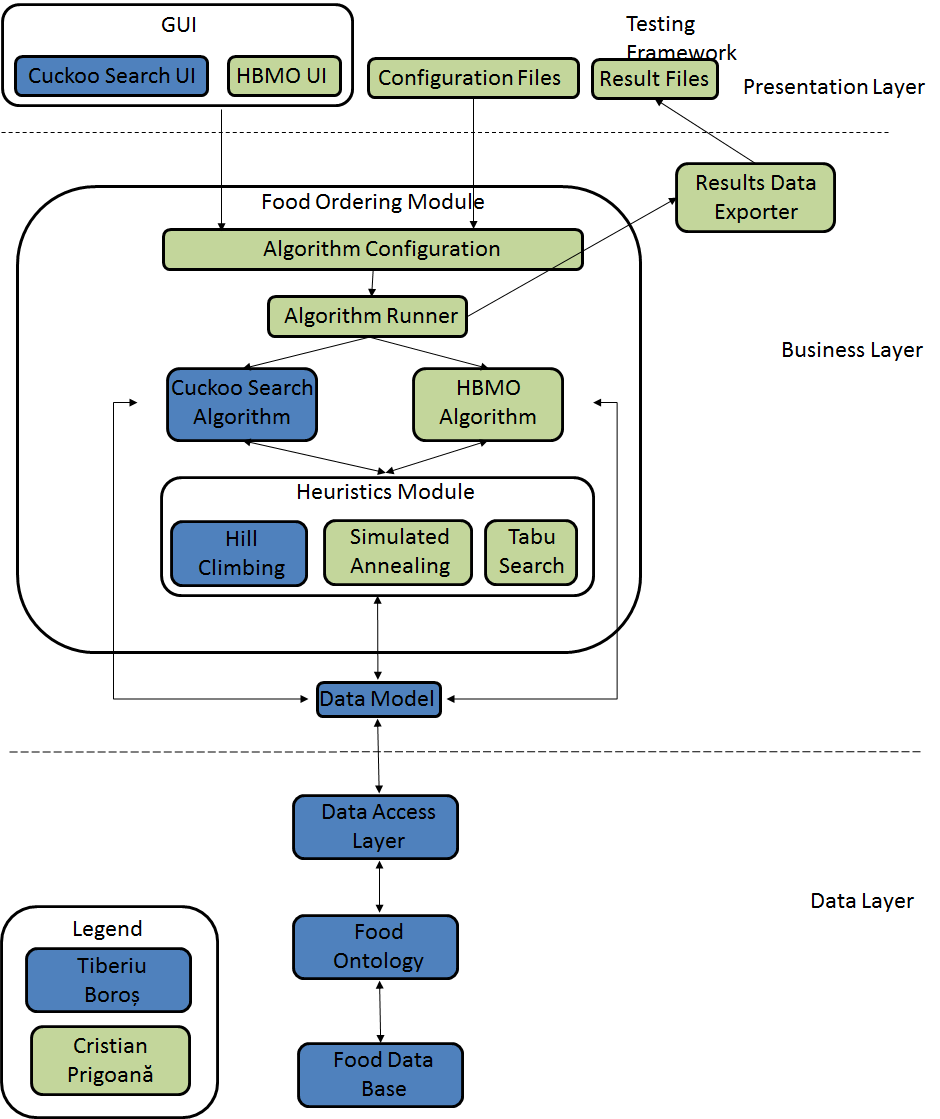


Figure 6-8. Testing Framework Conceptual Architecture

The Testing Framework is composed of the following modules and resources:

1. **Data Layer (DAL)**

* **Food Database** - that is populated from by data gathered from the USDA National Nutrient Database
* **Food Ontology** – the ontology model representing the the entire model of the system that is used; the ontology is partially populated by the Food Database
* **Data Access Layer (DAL) –** this module provides simplified access to data stored in the ontology (it actually gathers data from the Food Database through the ontology by D2RQ mapping)

1. **Business Layer**

* **Data Model –** this module contains the representation of the model implemented for the use and integration in the Cuckoo Search and Honey Bee Mating Optimization algorithm
* **Food Ordering Module -**  this module contains both the Cuckoo Search algorithm and the Honey Bee Mating Algorithm as well as heuristics that are applied to hybridize the algorithms; the algorithm runner is a harness for running the algorithm and injecting a configuration setup with adjustable parameters

1. **Presentation Layer**

* **Results Data Exporter –** this module is for exporting results to files in order to offer presentation of statistics and charts for comparison and evaluation
* **Graphical User Interface** **(GUI)** - for the Cuckoo Search algorithm and for the Honey Bee Mating Optimization designed by my collegue Cristian Prigoana; the GUI defers by the different adjustable parameters that are in the configuration panels;

### Class Diagrams

In this section the main diagrams of most relevant classes will be presented. The following section will included:

* Class diagram of the Brood Improver Package
* Class diagram of the Model Package
* Class diagram of the Algorithm package

#### Brood Improver Package

The Brood Improver package class diagram is presented in Figure 6-9. As seen in the diagram there are three main solution improver heuristics: *SimulatedAnnealingBroodImprover, TabuSearchBroodImprover* and  *HillClimbingBroodImprover.* These heuristics implement the *BroodImproverAlgorithm* class which has only one functionality and that is to apply „improve” to a solution, and this will basically take the solution from the parameter and apply the certain heuristic process implemented in the heuristics and obtain a better overall result.

Configurations are available for each heuristic in order to adjust the parameter for the purpose of calibrating and improving the heuristic. These heuristics can run in parallel with the help of the *Partition* class and the number of threads can also be adjusted. The class already has an implicit value howeever this must be adjusted according to the system’s specifications. At the end of the parallel run, the solutions are joined and certain solutions are selected.

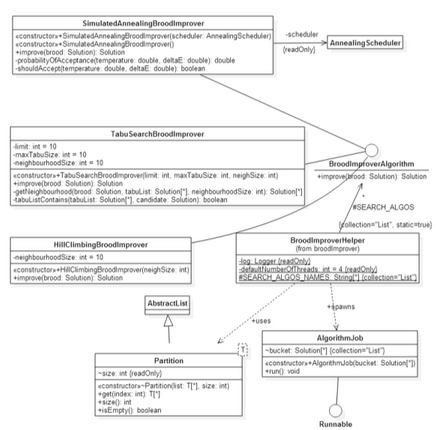


Figure 6-9. Class diagram of Brood Improver Package

#### Model Package

The model package is inspired from the model that has been designed for the MySQL database and the ontology.

The *FoodNutrients* class contains all the nutrient information that is used in this context, and each class of the model inherit this class because the nutritional values are computed from the construction of each model bottom up.

The entities are hierarchically constructed, as already explained in previous subchapters.

The *Recipe* class is contained by the *Dish* class. This mapping is one to one, thus each dish will have one recipe. The *Dish* class used by the *MealVariant* class. This class represents a meal of the day and has one, two or three dishes depending on the meal type.

*Menu* class contains 5 *MealVariant* classes and it stands for the day’s meals which are breakfast, lunch, dinner and two snacks.

#### Algorithm Package

The Algorithm Package can be seen in Figure 6-11. This package hold the two algorithm developed in this project. It can be seen that there is a *MainAlgorithm* class for enforcing OOP principles and each algorithm uses *BroodImprover* as well as extensive use of the model for performing a run.

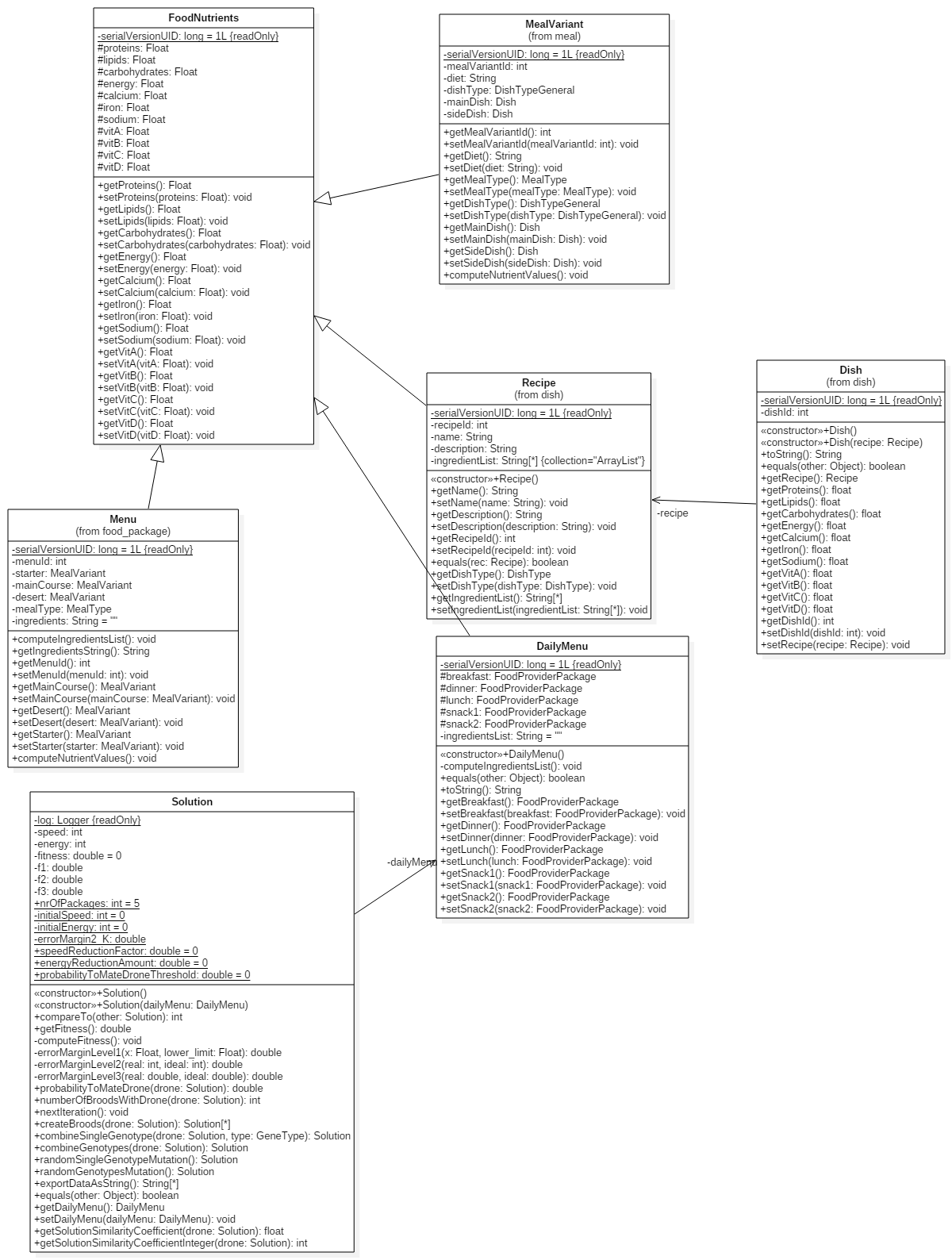


Figure 6-10. Diagram of the model of the system

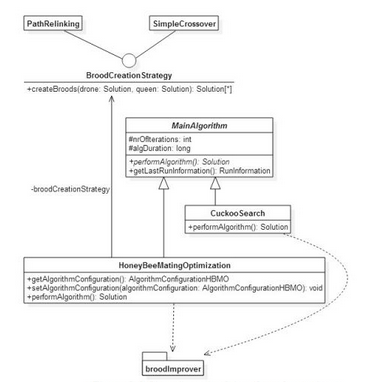


Figure 6-11. Class diagram of Algorithm package

## Implementation Details

### Tools and Technologies

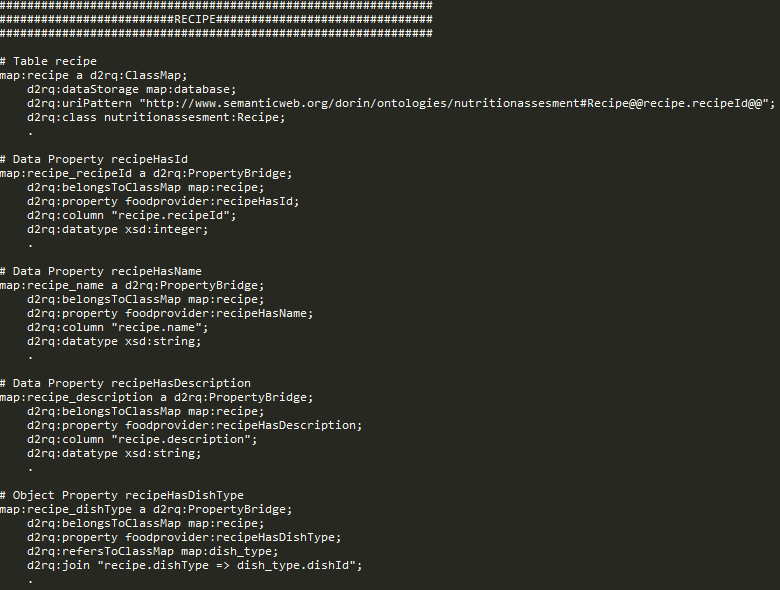
For the implementation of this application the following tools and technologies have been used:

* Protégé – is a tool that contains a large set of modelling structures that help the development, visualization and processing of ontologies. It is a free, open source ontology editor and knowledge acquisition system.
* MySQL – is a relational database management system (RDBMS) and I have used this together with the MySQL Workbench which is the graphic tool that MySQL community offers
* Eclipse – is an integrated development environment (IDE) that contains a base workspace and an extensible plug-in system for customizing the environment
* D2RQ – the D2RQ platform is a system for accessing relational databases as virtual, read-only RDF graphs. I have used this to map the MySQL database with the ontology
* StarUML –is a tool used for creating UML diagrams; by the help of this different types of diagrams can be accomplished
* Gliffy – is an online tool for creating UML diagrams; by the help of this different types of diagrams can be accomplished
* GitHub – is a web-based Git repository hosting service, which offers distributed revision control and source code management

### Code Sequences

#### D2RQ Ontology mapping

The file foodprovider2.ttl from the local path: ‘src/main/resources/ontology’ contains the mapping of all the database entities to ontology entities. Example of script snippets are presented bellow.



Figre 6-12. D2RQ mapping snippet of recipe entity

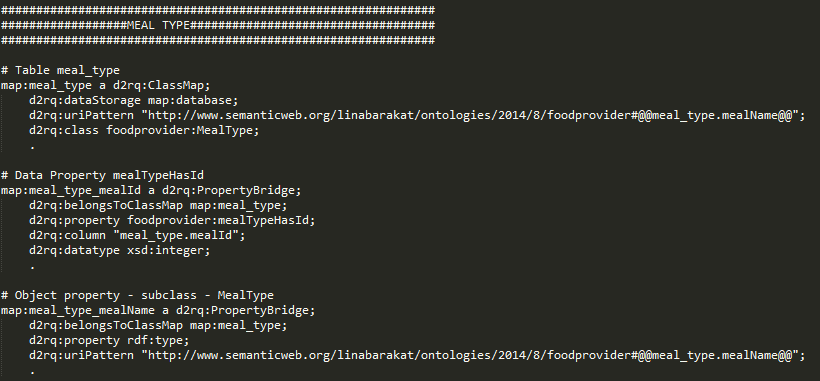


Figure 6-13. D2RQ mapping snippet of Meal \_Type entity

#### Solution Crossover Method

As described in the Analysis and Theoretical Foundation Chapter, the crossover between two solutions is done at random points, and the results solution may have genomes from bots „parents” in a random manner.

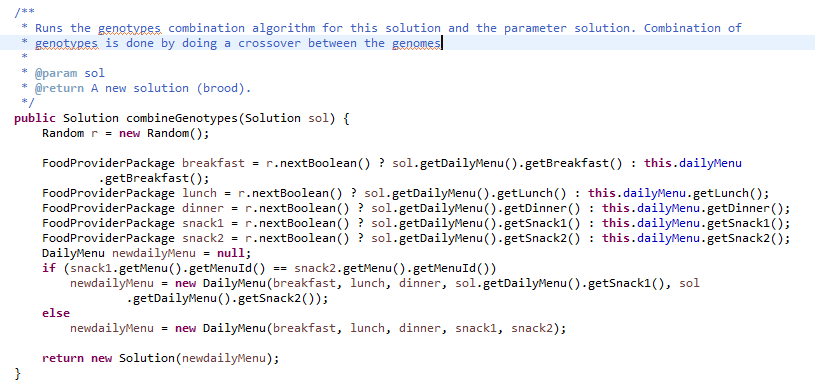


Figure 6-14. Crossover Method of a solution with another

#### Improving the newly generated solutions when a nest is discovered

If a nest is discovered, then generate a new nest by using a hill climbing heuristic. The newly generated solution will supposedly always be better than the previous solution so that the results keep improving.

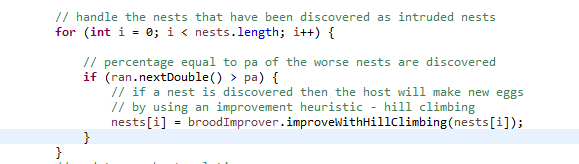


Figure 6-15. Improving newly generated nests by using hill climbing

#### SPARQL query for gathering data

For querying the ontology SPARQL queries have been used. In the Figure 6-13 a SPARQL query can be seen, used for retrieving all the recipes from the ontology.

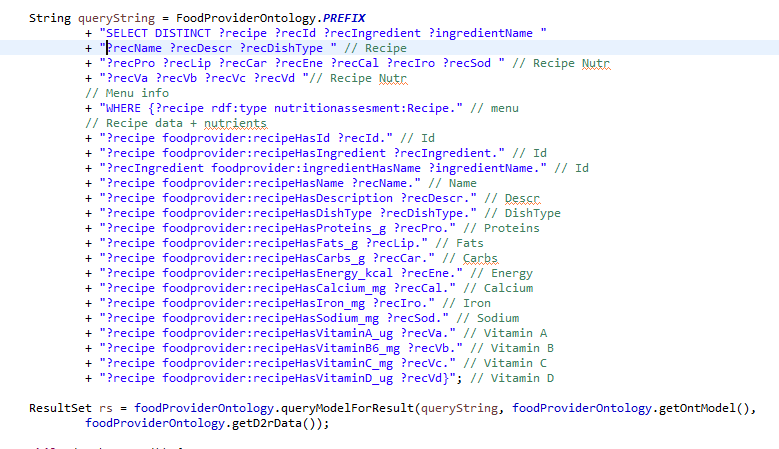


Figure 6-16s. SPARQL query for retrieving recipes from the ontology

# Testing and Validation

The technique that have been developed throughout the whole process have been tested on different scenarios that are presented in this section. Thus, an analysis can be made for comparison of the proposed techniques. Also, an analysis between Cuckoo Search hybrid technique and Honey Bee Mating Optimization hybrid technique developed by my colleague Cristian Prigoana will be made.

The porpose of this chapter is to verify:

* Result Validation
* Best setup of adjustable parameters – tuning the algorithm

## Test Scenarios

The system on which these tests have been made has the following specification:

* Processor: Intel® Core™ i7 2670QM/2630QM Processor
* Memory: 8096MB DDR3
* OS: Microsoft Windows 8 64bit

The user profile is presented in the table 7-2 along with the ideal quantities for nutrients (table 7-1).

|  |  |  |
| --- | --- | --- |
| Nutrient | Value | UM |
| Energy | 2287 | KCal |
| Carbohydrates | 314 | G |
| Proteins | 128 | G |
| Lipids | [114, 200] | G |
| Calcium | 1200 | Mg |
| Sodium | 1500 | Mg |
| Iron | 8 | Mg |
| Vitamin A | 900 | µg |
| Vitamin B | 1.3 | Mg |
| Vitamin C | 90 | Mg |
| Vitamin D | 20 | µg |

Table 7-1. User profile ideal nutrition values

|  |  |
| --- | --- |
| User Parameter | Value |
| Age | 70 years |
| Gender | Male |
| Height | 78 kg |
| Weight | 172 cm |
| PAF | 1.5 |
| Likes | Lettuce, apricot |
| Dislikes | Pepper |
| Preferred delivery time | 40 minutes |
| Preferred cost | 65 RON |

Table 7-2. User profile used for testing 1

## Validation

In this section, the focus is on the values of the nutrients that appear in the meal and the food items that appear in the meal as well as the cost and delivery time. Theoretically, they should be as close as possible to the ideal values that are mentioned in Table 6-1 and

Table 6-2. The system has run the algorithm 10 times with the default values and its output has the following characteristics:

* Fitness value and duration – the average fitness value and the average duration are found in Table 6-3. A value of 0.866 for the fitness value is very acceptable from the authors’ point of view. It roughly means that 86% of the parameters have real values close to the ideal ones. Also, the system returned the menu recommendation in just 330 milliseconds on verage, a value that is even comparable to the polynomial algorithms.

|  |  |
| --- | --- |
| Parameter | Real |
| Average Fitness | 0.8197 |
| Average Duration | 1080 |

Table 7-3. Results of run

* Nutrients – the values of the nutrients found in the recommended food menu are expressed in Table 6-4 in the Real values column and the ideal values next to them so that it is easy to compare the values and to check whether they are acceptable or not. The Mark column represents a mark given by the author for estimating how accurate the algorithm was for each of the parameter. Out of 11 parameters, there are 6 green, 3 yellow and just 2 red parameters.
  + Green means high accuracy – may deviate up to 25%
  + Yellow means mediocre accuracy – may deviate up to 60%
  + Red mean low accuracy – the deviation is above 60%
  + Deviation is computed as following: |𝑖𝑑𝑒𝑎𝑙−𝑟𝑒𝑎𝑙| / 𝑟𝑒𝑎𝑙 ∗100

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Parameter | Real values | Ideal values | | Derivation(%) | Mark | UM |
| Energy | 2020 | 2287 | 11.78 | | GREEN |  |
| Carbohydrates | 192 | 314 | 19.15 | | GREEN |  |
| Proteins | 82.74 | 128 | 35.36 | | YELLOW |  |
| Lipids | 109.2 | [114, 200] | 4.22 | | GREEN |  |
| Calcium | 498.48 | 1200 | 68.23 | | RED |  |
| Sodium | 2361.95 | 1500 | 57 | | YELLOW |  |
| Iron | 10.3 | 8 | 28.75 | | YELLOW |  |
| Vitamin A | 1237.92 | 900 | 37.44 | | YELLOW |  |
| Vitamin B | 1.89 | 1.3 | 45.38 | | YELLOW |  |
| Vitamin C | 88.29 | 90 | 1.19 | | GREEN |  |
| Vitamin D | 2.11 | 20 | 89 | | RED |  |

Table 7-4. Results nutrients-wise

* Food items – the recommended food menu is composed out fo the food items described in table 7-5

|  |  |  |  |
| --- | --- | --- | --- |
| Meal | Starter | Main Course | Dessert |
| Breakfast | N/A | Special Armadillo Eggs | Cinnamon roll |
| Lunch | Ark Soup | Aromatic Green Casserole | Apricot Pear Tart |
| Dinner | Amish-Style Chicken and Corn Soup | Aromatic and Crispy Duck | Lemon bars |
| Snack1 | N/A | Apricot Shortbread Sandwiches | N/A |
| Snack2 | N/A | Amaranth, Fig and Arugula Salad W/sesame Dres | N/A |

Table 7-6. Results meals

* Cost and delivery time of the solution shown in the Table 7-6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Parameter | Breakfast | Lunch | Dinner | Snack1 | Snack2 | UM |
| Cost | 10 | 15 | 20 | 8 | 7 | RON |
| Delivery Time | 35 | 24 | 45 | 67 | 60 | minutes |

Table 7-7. Results cost and delivery time wise

## Best Configuration Setup

This section has the purpose of presenting the results with the variations of the adjustable parameters from the Cuckoo Search Optimization algorithm. There are four versions of the Cuckoo Search which were described in Chapter 5 in the Cuckoo Search Versions subchapter and are named as the following:

* VersionRLRL - hybridization points both with random nest generation; this is the classic version of the Cuckoo Search algorithm
* VersionUCUC – hybridization points with guided new solution generation by the best solution
* VersionUCHC – hybridization points with guided new solution generation by the best solution for the cuckoo solutions and hill climbing heuristic improvement for the nest intrusion discovery by the host bird
* VersionTSHC – hybridizations points with tabu search heuristic improvement for the cuckoo solutions and hill climbing heuristic improvement for the nest intrusion discovery by the host bird

### PA Variation

In this section, results are shown with the comparison of the four algorithms when a PA variation is applied. PA can have values from 0 to 1 and is varied by the values 0.2, 0.4, 0.6, 0.8 and 0.9. The result charts have on the X axis the PA value and on the Y axis in the first chart is the fitness values of the results and in the second chart is the duration in milliseconds.

Figure 7-1. Fitness value with PA variation

Figure 7-2. Duration value with PA variation

In conclusion, for this problem context the PA parameter does not improve or drop the fitness value, however the duration of the search drops considerably if the PA is higher for the VersionUCHC and the VersionTSHC because at each nest discovery the solution is improved, however for VersionRLRL and VersionUCUC the duration remains rather constant.

Thus the PA parameter should be chosen to be maximum for the decrease in the duration of the search

**BEST PA: 1**

### NestNumber Variation

In this section, results are shown with the comparison of the four algorithms when a NestNumber variation is applied. NestNumber can have any integer value greater than 0 and for the test scenario it is varied by the values 5, 10, 15, 20, 25, 30, 50. The result charts have on the X axis the NestNumber value and on the Y axis in the first chart is the fitness values of the results and in the second chart is the duration in milliseconds.

Figure 7-3. Fitness value with number of nests variation

Figure 7-4. Duration values with number of nests variation

In conclusion it can be seen in Figure 7-3 that the higher the nest number is the better the fitness value is for all the algorithm version, however the rate of increase in the fitness value drops for nest number higher than 20, but nevertheless, the fitness value improves.

For the first three algorithms VersionRLRL, VersionUCUC and VersionUCHC the duration is significantly better than VersionTSHC when the nest number is varied. It can be seen in that in Figure 7-4 that duration increases greatly with the increase of the number of nests, thus this is a matter of performance/results matter.

**BEST NestNumber: -** debatable, the higher the nestnumber the better the the fitness value the worser the duration; for an equilibrium the range 15-25 nests seems appropriate

### MaxIterations Variation

In this section, results are shown with the comparison of the four algorithms when a MaxIterations variation is applied. MaxIterations can have any integer value greater than 0 and for the test scenario it is varied by the values 50. 100. 250, 500, 750, 100, 1500. The result charts have on the X axis the MaxIterations value and on the Y axis in the first chart is the fitness values of the results and in the second chart is the duration in milliseconds.

Figure 7-5. Fitness values with MaxIterations variation

In conclusion it can be seen that the fitness value increases with the increase of the number of iterations for all algorithms, especially for the classic one.

The duration value also increases in a constant manner, however the duration increase rate is higher for the fourth algorithm and the fitness value does not increase substantially

**Best MaxIterations:** - for the fourth algorithm the fitness does not go up that much after 500 iteration, but the duration grows way more; thus for VersionTSHC the best maxIterations would be about 500, in that area; this goes for Version UCHC as well, however for the other two algorithm versions (VersionRLRL and VersionUCUC) the maxIterations value does not increase the duration as much as it does for the other two so 1000-1500 iterations would be good

Figure 7-6. Durations values with MaxIterations variations

## Cuckoo Search Hybrid Versions Comparison

In this section the four algorithms versions VersionRLRL, VersionUCUC, VersionUCHC and VersionTBHC will be compared. These algorithms have been developed iteratively, bringing improvement from a version to another. This will be shown in the following charts in Figure 7-7 where the average fitness values of several runs of the versions are compared to one another with small variations of the variables to see wether the results are linear across the variables’ variation.

Figure 7-7. Fitness Value with adjustable variables variation

\

Figure 7-8. Duration Values with adjustable variables variation

# User’s manual

## Installation manual

The software application is written in the Java programming which fortunately allows easy installation. Once the Java Virtual Machine is installed on the system, which is the most difficult part, the Java Archive of the application is run by the Java Virtual Machine. The source code is found on the GitHub service. The Eclipse development environment is highly recommended also. The steps to install the applications are the following:

1. Install the Java Virtual Machine

* Go to https://java.com/en/download/ website
* Press the download button to download the installer
* Use the installer wizard to install the Java Virtual Machine 2. Install Eclipse from https://eclipse.org/downloads/

3. Create a new project from GitHub code

* File->Import->Projects from Git->Clone URI
* Type https://github.com/tiberiu-b/diet4elders.git to the URI text field
* The Host and Repository Path fields as well as the Protocol drop-down selection box are automatically populated
* Do not type anything in the User and Password fields if you do not want get authenticated because the repository is public
* Click Next and select only the masterHybridCuckooSearch branch
* Click Next and select the desired location
* Click Next and select the desired wizard for new project creation and continue with the standard procedure from now on.
* Build the project

## User guide

The user can introduce the parameters of the algorithm in two ways, either through the graphical user interface or the configuration files. For the latter one, however, the algorithm is triggered by a button in GUI.

The GUI presented in Figure 8-1 is the configuration panel allows custom configuration of a single run of the algorithm without heuristics, because the checkboxes are not selected. In fact, by default the algorithm runs 10 times in order to compute the average. The output is presented in a selectable text field for further analysis. The basic information is printed, like the average fitness, average duration, the meals, but they can be easily customized from the code. In Figure 7-2 the configuration that contains the parameters of the heuristics improving the solutions is shown. The “Run” button in the left will trigger the algorithms.

Next if an INI configuration file is used, like Figure 5-2, the “Run INI config” button must be pressed. In both cases, the application writes the results in CuckooSearch\_data.csv

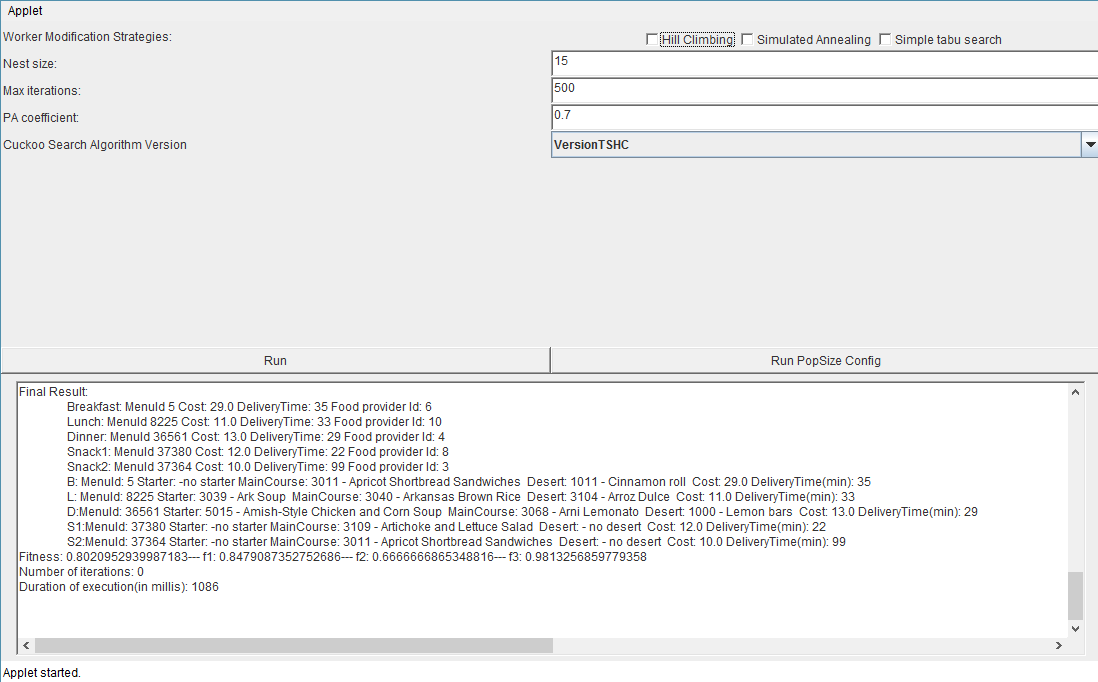


Figure 8-1. GUI of Cuckoo Search testing framework

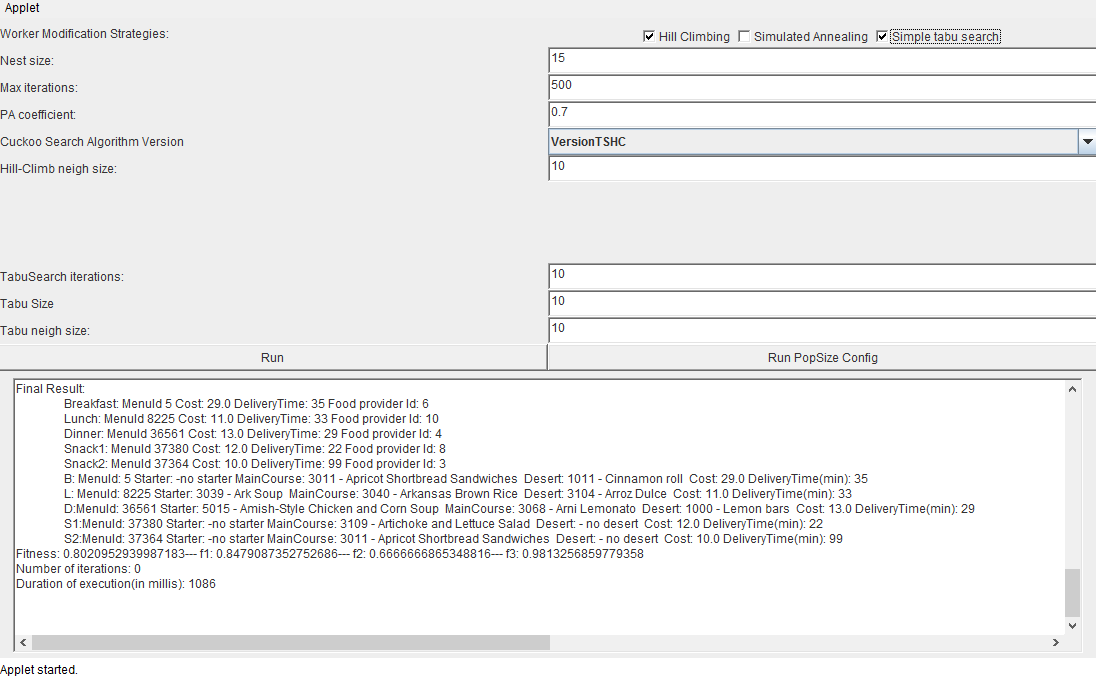


Figure 8-2. GUI of Cuckoo Search testing framework with heuristic adjustable parameters

# Conclusions

In this chapter, to conclude the work that has been done in this license thesis, the contributions and achievements that this thesis has brought are presented, along with an analysis of the results a future development possibilities for this application.

## Contributions and Achievements

In this thesis a bio-inspired hybrid optimization algorithm has been proposed as a solution to the problem of generating food menu recommendations in a personalized manner, as well as a testing framework for analysis of the results.

The application that has been built on the basis of this thesis has reached its main goal by the use of hybrid Cuckoo Search optimization algorithm that has been developed in this thesis. This was possible by following the list of the objectives that have been presented in chapter 2.

Due to the difficulties that have risen with the growth of the senior population which has led to a major need of health care services, this project is an innovative contribution to the solutions that have appeared with the purpose of aiding this situation. The overall medical system care experience would be greatly improved by the use this project. This project is small step in the ladder of health care system improvement. By having a system that assures proper nutrition to elder people, the health care becomes more efficient, more available (the „cyber” carer will be present anytime) and it will reduce costs.

Another important contribution that this thesis brings is a hybrid version of the existing Cuckoo Search algorithm that is an improvement. This has been done by applying the certain key-points named hybridization points heuristics inspired from genetic algorithms. The heuristics that have been proposed in this thesis are Hill Climbing and Tabu Search. The overall results are improved in terms of computation time, and fitness value.

## Critical Analysis of Results

The hybrid Cuckoo Search algorithm described in this license thesis is better than the classic version of the Cuckoo Search as shown in the seventh chapter where the results have been compared. Thus, three additional hybrid versions have been developed iteratively and compared. The improvement has shown to be significant. However, as shown in Figure 7-4, the results are limited not only by the performance of the algorithm by also by the solution domain. Because of the fact that to perfectly prescribe a food menu to a user, by the exact ideal values of this profile is firstly there must exist food menus that have ideal nutrition values. In the solutions achieved some nutrients have bad nutrient values and this is because there is a large number of nutrients and the overall score is composed, thus some may be very good and some bad.

## Future Development

Further development to this application can be made in many ways. This is also because of the fact that the current application is only a prototype to harness the developed algorithms and not intended for market with users that do not have knowledge about the flow of the application.

Future development can also be done on the performance of this application. Especially to the part of the data manipulation and system caching. The retrieval of food packages from the database can be optimized by improving the data access layer.

The user interface is very basic and does not provide a pleasant user experience by comparison with the interface standards that are at present. The current look may give off a flair of a cheap software product which does not reflect the amount of theoretical work that has been done behind. Another remark on the user interface is that the current application does not offer interfaces for the nutritionist that would modify recommended nutritional values, nor the local food vendors that would basically have CRUD access to the food packages that are added by the vendor therefore these are on the list of future development tasks.

Currently the solutions generated are a day’s meals, however it can be extended due to the solid ontology model design to theoretically unlimited number of days. Scaling the meal to multiple days would involve other problems such as offering a diversity in the food menu generation from a day to another and accessing the nutritional values not only for a day but for a set of days, thus improving the monitoring process of the senior.

Also, features that would offer preset nutritional recommendations and prescription for certain diseases and medical conditions could be added. Preset values for senior that want to lose weight, or gain weight can be added, so the food menu recommendations will provide some extra/less carbs or energy etc.

# Bibliography

[1] Y. Usthasopha, S. Phimoltares, N. Cooharojananone, Nutrition Counseling System and Food Menu Planning, International Conference on Information Science and Applications, 2010.

[2] S. Sivilai, C. Snae, M. Brückner, Ontology-Driven Personalized Food and Nutrition Planning System for the Elderly, The 2nd International Conference in Business Management and Information Sciences, 2012.

[3] S. Mika, Challenges for Nutrition Recommender Systems, Context Aware Intelligent Assistance, 2011.

[4] J. Bulka, A. Izworski, J. Koleszynska, J. Lis, I. Wochlik, Automatic meal planning using artificial intelligence algorithms in computer aided diabetes therapy, Proceedings of the 4th International Conference on Autonomous Robots and Agents, 2009.

[5] Nuno Alexandre de Jesus Lopez, A SPARQL Query Engine over Web Ontologies using Contextual Logic Programming, Master Degree in Computer Science, Universidade de Evora, October, 2007

[6] Algirdar Sukys, Lina Nemuraite, Bronius Paradauskas, Edvinas Sinkevicius, SBVR Based Representation of SPARQL Queries and SWRL Rules for Analyzing Semantic Relations, Bustech 2011: The First International Conference on Business Intelligence and Technology

[7] OWL 2 and SWRL tutorial, <http://dior.ics.muni.cz/~makub/owl/>

[8] US Department of Agriculture, Agricultural Research Service, Nutrient Data Laboratory. USDA National Nutrient Database for Standard Reference, Release 27 (revised). Version Current:  May 2015.  Internet: <http://www.ars.usda.gov/ba/bhnrc/ndl>

[9] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A survey on metaheuristics for stochastic combinatorial optimization. 8, 2 (June 2009), 239-287. DOI=10.1007/s11047-008-9098-4 <http://dx.doi.org/10.1007/s11047-008-9098-4>

[10] Blum, C.; Roli, A. (2003). "Metaheuristics in combinatorial optimization: Overview and conceptual comparison" **35** (3). ACM Computing Surveys. pp. 268–308.

[11] X. S. Yang, Engineering Optimization - an Introduction with Metaheuristic Applications, Cap. 2, 18, p. 15-19, 233-239, 2010.

[12] Talbi, E-G. (2009).Metaheuristics: from design to implementation. Wiley. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-470-27858-7](https://en.wikipedia.org/wiki/Special:BookSources/0-470-27858-7).

[13] [Skiena, Steven](https://en.wikipedia.org/wiki/Steven_Skiena) (2010). *The Algorithm Design Manual* (2nd ed.). [Springer Science+Business Media](https://en.wikipedia.org/wiki/Springer_Science%2BBusiness_Media). [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [1-849-96720-2](https://en.wikipedia.org/wiki/Special:BookSources/1-849-96720-2).

[14] Yang, X.-S., and Deb, S. (2010), Engineering Optimisation by Cuckoo Search. This revised version included a demo version of the cuckoo search as a supplement; Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No.4, 330-343 (2010)

# Appendix 1

**Cuckoo Search VersionTSHC`**

**private** Solution csVersionTSHC () {

// Adjustable parameters

**int** nestNumber = algorithmConfiguration.getNestSize();

**int** maxIterations = algorithmConfiguration.getMaxIterations();

**double** pa = algorithmConfiguration.getPa();

// local parameters

BroodImproverHelper broodImprover = **new** BroodImproverHelper();

**int** t = 0;

Random ran = **new** Random();

Solution bestSolution;

**int**[] nestSimilarity = **new** **int**[nestNumber];

// Generate initial solutions

Solution[] nests = **new** Solution[nestNumber];

solGenerator.generateRandomSolutions(nestNumber).toArray(nests);

// Best solution is last, because it originates from an ordered structure

bestSolution = nests[nestNumber - 1];

// create a number of cuckoos equal to the nest size

Solution[] cuckoos = **new** Solution[nestNumber];

**while** (t++ < maxIterations) {

// generate new cuckoos

**for** (**int** i = 0; i < cuckoos.length; i++) {

// new cuckoo makes eggs similar to nest eggs but fitness value

// will be improved by use of hill climbing heuristic

cuckoos[i] = broodImprover.improveWithTabuSearch(nests[i]);

// Compute similarity coefficient between cuckoo and nest

nestSimilarity[i] = cuckoos[i].getSolutionSimilarityCoefficientInteger(nests[i]);

}

// replace all the nests that have a lower fitness value

**for** (**int** i = 0; i < nests.length; i++) {

**if** (nests[i].getFitness() < cuckoos[i].getFitness()) {

nests[i] = cuckoos[i];

}

}

// handle the nests that have been discovered as intruded nests

**for** (**int** i = 0; i < nests.length; i++) {

// percentage equal to pa of the worse nests are discovered

**if** (ran.nextDouble() > pa) {

// if a nest is discovered then the host will make new eggs

// by using an improvement heuristic - hill climbing

nests[i] = broodImprover.improveWithHillClimbing(nests[i]);

}

}

// update new best solution

**for** (Solution sol : nests)

**if** (bestSolution.getFitness() < sol.getFitness())

bestSolution = sol;

}

algDuration = System.*currentTimeMillis*() - start;

**return** bestSolution;

}

**Cuckoo Search VersionUCHC**

**private** Solution csVersionUCHC () {

// Adjustable parameters

**int** nestNumber = algorithmConfiguration.getNestSize();

**int** maxIterations = algorithmConfiguration.getMaxIterations();

**double** pa = algorithmConfiguration.getPa();

// local parameters

BroodImproverHelper broodImprover = **new** BroodImproverHelper();

**int** t = 0;

Random ran = **new** Random();

Solution bestSolution;

**int**[] nestSimilarity = **new** **int**[nestNumber];

// Generate initial solutions

Solution[] nests = **new** Solution[nestNumber];

solGenerator.generateRandomSolutions(nestNumber).toArray(nests);

// Best solution is last, because it originates from an ordered structure

bestSolution = nests[nestNumber - 1];

// create a number of cuckoos equal to the nest size

Solution[] cuckoos = **new** Solution[nestNumber];

**while** (t++ < maxIterations) {

// Compute mean fitness value of all nests for statistics

// generate new cuckoos

**for** (**int** i = 0; i < cuckoos.length; i++) {

// new cuckoo makes eggs similar to nest eggs but fitness value

// will be improved by use of hill climbing heuristic

cuckoos[i] = broodImprover.improveWithHillClimbing(nests[i]);

// Compute similarity coefficient between cuckoo and nest

nestSimilarity[i] = cuckoos[i].getSolutionSimilarityCoefficientInteger(nests[i]);

}

// replace all the nests that have a lower fitness value

**for** (**int** i = 0; i < nests.length; i++) {

**if** (nests[i].getFitness() < cuckoos[i].getFitness()) {

nests[i] = cuckoos[i];

}

}

// handle the nests that have been discovered as intruded nests

**for** (**int** i = 0; i < nests.length; i++) {

// percentage equal to pa of the worse nests are discovered

**if** (ran.nextDouble() > pa) {

// if a nest is discovered then the host will make new eggs

// by using an improvement heuristic - hill climbing

nests[i] = broodImprover.improveWithHillClimbing(nests[i]);

}

}

// update new best solution

**for** (Solution sol : nests)

**if** (bestSolution.getFitness() < sol.getFitness())

bestSolution = sol;

}

algDuration = System.*currentTimeMillis*() - start;

**return** bestSolution;  
}

**Hill Climbing Brood Improver**

**public** **class** HillClimbingBroodImprover **implements** BroodImproverAlgorithm {

**private** **int** neighbourhoodSize = 10;

**public** HillClimbingBroodImprover(**int** neighSize) {

neighbourhoodSize = neighSize;

}

@Override

**public** Solution improve(Solution brood) {

**final** **int** limit = 100;

**int** timeStep = 0;

**boolean** localMaximum = **false**;

**while**(timeStep < limit && !localMaximum){

++timeStep;

localMaximum = **true**;

**for**(**int** i = neighbourhoodSize; i > 0; i--){

Solution newBrood = brood.randomSingleGenotypeMutation();

**if**(newBrood.getFitness() > brood.getFitness()){

brood = newBrood;

localMaximum = **false**;

}

}

}

**return** brood;

}

}

**D2RQ Mapping**

**# D2RQ Namespace**

**@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .**

**# Namespace of the ontology**

**@prefix nutritionassessment: <http://www.semanticweb.org/dorin/ontologies/nutritionassessment#> .**

**@prefix nutritionmonitoring: <http://www.semanticweb.org/dorin/ontologies/nutritionmonitoring#> .**

**@prefix nutritiondiagnostic: <http://www.semanticweb.org/dorin/ontologies/nutritiondiagnostic#> .**

**@prefix food: <http://www.semanticweb.org/dorin/ontologies/food#> .**

**@prefix foodprovider: <http://www.semanticweb.org/linabarakat/ontologies/2014/8/foodprovider#> .**

**@prefix nutritionassesment: <http://www.semanticweb.org/dorin/ontologies/nutritionassesment#> .**

**# Namespace of the mapping file; does not appear in mapped data**

**@prefix map: <#> .**

**# Other namespaces**

**@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .**

**@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .**

**@prefix jdbc: <http://d2rq.org/terms/jdbc/> .**

**@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .**

**map:database a d2rq:Database;**

**d2rq:jdbcDSN "jdbc:mysql://localhost/toateretetele";**

**d2rq:jdbcDriver "com.mysql.jdbc.Driver";**

**d2rq:username "root";**

**d2rq:password "root";**

**jdbc:autoReconnect "true";**

**jdbc:zeroDateTimeBehavior "convertToNull";**

**.**

**##############################################################**

**##################MEAL TYPE###################################**

**##############################################################**

**# Table meal\_type**

**map:meal\_type a d2rq:ClassMap;**

**d2rq:dataStorage map:database;**

**d2rq:uriPattern "http://www.semanticweb.org/linabarakat/ontologies/2014/8/foodprovider#@@meal\_type.mealName@@";**

**d2rq:class foodprovider:MealType;**

**.**

**# Data Property mealTypeHasId**

**map:meal\_type\_mealId a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:meal\_type;**

**d2rq:property foodprovider:mealTypeHasId;**

**d2rq:column "meal\_type.mealId";**

**d2rq:datatype xsd:integer;**

**.**

**# Object property - subclass - MealType**

**map:meal\_type\_mealName a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:meal\_type;**

**d2rq:property rdf:type;**

**d2rq:uriPattern "http://www.semanticweb.org/linabarakat/ontologies/2014/8/foodprovider#@@meal\_type.mealName@@";**

**.**

**##############################################################**

**######################DISH TYPE###############################**

**##############################################################**

**# Table dish\_type**

**map:dish\_type a d2rq:ClassMap;**

**d2rq:dataStorage map:database;**

**d2rq:uriSqlExpression "CONCAT('http://www.semanticweb.org/linabarakat/ontologies/2014/8/foodprovider#',REPLACE(dish\_type.dishName,' ','\_'))";**

**d2rq:class foodprovider:DishType;**

**.**

**# Data Property hasDishTypeId**

**map:dish\_type\_dishId a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:dish\_type;**

**d2rq:property foodprovider:dishTypeHasId;**

**d2rq:column "dish\_type.dishId";**

**d2rq:datatype xsd:integer;**

**.**

**# Object property - subclass - DishType**

**map:dish\_type\_dishName a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:dish\_type;**

**d2rq:property rdf:type;**

**d2rq:uriSqlExpression "CONCAT('http://www.semanticweb.org/linabarakat/ontologies/2014/8/foodprovider#',REPLACE(dish\_type.dishName,' ','\_'))";**

**.**

**##############################################################**

**#########################RECIPE###############################**

**##############################################################**

**# Table recipe**

**map:recipe a d2rq:ClassMap;**

**d2rq:dataStorage map:database;**

**d2rq:uriPattern "http://www.semanticweb.org/dorin/ontologies/nutritionassesment#Recipe@@recipe.recipeId@@";**

**d2rq:class nutritionassesment:Recipe;**

**.**

**# Data Property recipeHasId**

**map:recipe\_recipeId a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:recipe;**

**d2rq:property foodprovider:recipeHasId;**

**d2rq:column "recipe.recipeId";**

**d2rq:datatype xsd:integer;**

**.**

**# Data Property recipeHasName**

**map:recipe\_name a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:recipe;**

**d2rq:property foodprovider:recipeHasName;**

**d2rq:column "recipe.name";**

**d2rq:datatype xsd:string;**

**.**

**# Data Property recipeHasDescription**

**map:recipe\_description a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:recipe;**

**d2rq:property foodprovider:recipeHasDescription;**

**d2rq:column "recipe.description";**

**d2rq:datatype xsd:string;**

**.**

**# Object Property recipeHasDishType**

**map:recipe\_dishType a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:recipe;**

**d2rq:property foodprovider:recipeHasDishType;**

**d2rq:refersToClassMap map:dish\_type;**

**d2rq:join "recipe.dishType => dish\_type.dishId";**

**.**

**# Data Property recipeHasNumberOfServings**

**map:recipe\_numberOfServings a d2rq:PropertyBridge;**

**d2rq:belongsToClassMap map:recipe;**

**d2rq:property foodprovider:recipeHasNumberOfServings;**

**d2rq:column "recipe.numberOfServings";**

**d2rq:datatype xsd:integer;**

**.**