A picture containing text, clipart

Description automatically generated

**FCAI – Cairo University**

**Meal (diet) Selection Project**

*using Genetic Algorithm*

**Team members:**

|  |  |
| --- | --- |
| Name | ID |
| Ghassan Elgendy | 20220239 |
| Abdelrahman Ashraf | 20220189 |
| Mohamed Osama Khaled Mohamed | 20220477 |
| Ibrahim Medhat | 20221003 |

 **Table of Contents**

1. **Introduction**  
   1.1 Project Motivation  
   1.2 Problem Statement  
   1.3 Objectives  
   1.4 Report Structure
2. **Literature Review**  
   2.1 Overview of Diet Optimization Problems  
   2.2 Genetic Algorithm in Diet Planning  
   2.3 Review of Selected Research Papers  
   2.4 Key Takeaways and Research Gap
3. **Mathematical Formulation**  
   3.1 Decision Variables  
   3.2 Objective Function (e.g., cost, calories, or deviation from ideal macros)  
   3.3 Constraints (e.g., nutritional, preferences, cost, quantity)
4. **Implementation of Genetic Algorithm**  
   4.1 Data Preprocessing  
   4.2 Encoding Scheme  
   4.3 Fitness Function  
   4.4 Selection, Crossover, Mutation  
   4.5 Constraint Handling Technique  
   4.6 Algorithm Flowchart and Logic  
   4.7 Algorithm Parameters and Tuning
5. **Experiments and Results**  
   5.1 Test Case 1: Small Sample (5–10 foods)  
   5.2 Test Case 2: Medium Sample (50+ foods)  
   5.3 Test Case 3: Full Dataset (100+ foods)  
   5.4 Discussion of Results  
   5.5 Limitations and Edge Cases
6. **Fuzzification of a Constraint**  
   6.1 Selected Parameter for Fuzzification (e.g., daily protein intake)  
   6.2 Fuzzification and Membership Functions  
   6.3 Defuzzification Method  
   6.4 Impact on the Optimization Results
7. **Conclusion**  
   7.1 Summary of Findings
8. **References**

**1. Introduction**

**1.1 Project Motivation**

Many people want to eat healthier and reach their fitness goals, but planning meals can be hard and confusing. This project aims to make meal planning easier for everyone.

**1.2 Problem Statement**

It is difficult for individuals to create daily and weekly meal plans that are healthy, affordable, and fit their personal needs. There is a need for a tool that can do this automatically.

**1.3 Objectives**

* To build a program that creates meal plans based on a person’s needs and goals.
* To make sure the plans are healthy, varied, and not too expensive.
* To help users know what to eat and what to buy each week.

**3.2 Objective Function**

The objective is to find a meal plan that:

* Minimizes the total cost,
* Minimizes the deviation from nutritional targets,
* Minimizes impractical meal features (like too many small portions or lack of diversity).

**Mathematically:**

Fitness(x,Gen)=−Cost(x)−[PW(Gen)⋅∑kPenaltyk(x)+SmallPortionPenalty(x)]−DiversityPenalty(x)Fitness(**x**,Gen)=−Cost(**x**)−[*PW*(Gen)⋅*k*∑​Penalty*k*​(**x**)+SmallPortionPenalty(**x**)]−DiversityPenalty(**x**)

Where:

* Cost(x)=∑i=1nci⋅xi100Cost(**x**)=∑*i*=1*n*​*ci*​⋅100*xi*​​

(ci*ci*​ = cost per 100g of food i*i*)

* Penaltyk(x)Penalty*k*​(**x**) = penalty for deviation from nutrient k*k*'s target
* PW(Gen)*PW*(Gen) = penalty weight, increases with generation
* SmallPortionPenalty(x)SmallPortionPenalty(**x**) = penalty for small portions
* DiversityPenalty(x)DiversityPenalty(**x**) = penalty for lack of food group diversity

**(In code: fitness = -actual\_total\_cost - (penalty\_weight \* total\_nutrient\_penalty + small\_portions\_penalty) - food\_group\_diversity\_penalty)**

**3.3 Constraints**

1. **Portion bounds:**

0≤xi≤300∀i0≤*xi*​≤300∀*i*

(In code: enforced directly when generating and mutating solutions.)

1. **Nutritional constraints:**

For each nutrient k*k* (e.g., calories, protein, fats, etc.):

Lk≤∑i=1nNk,i⋅xi100≤Uk*Lk*​≤*i*=1∑*n*​*Nk*,*i*​⋅100*xi*​​≤*Uk*​

Where Nk,i*Nk*,*i*​ is the amount of nutrient k*k* per 100g of food i*i*, and Lk,Uk*Lk*​,*Uk*​ are lower and upper bounds (e.g., 90–130% of target).(In code: handled as soft constraints via penalty terms.)

1. **Other practical constraints:**

* **Small portions:** Penalize if too many foods have 0<xi<200<*xi*​<20.
* **Diversity:** Penalize if too many foods from the same group are included.

(In code: handled as penalty terms in the objective function.)

**4. Implementation of Genetic Algorithm**

**4.1 Data Preprocessing**

* **Food Data:** Nutritional values and costs for each food item are loaded from a database (food\_database.py).
* **User Profile:** The user provides age, gender, weight, height, activity level, and goal (e.g., lose, maintain, gain weight).
* **Requirement Calculation:** The function calculate\_daily\_needs computes daily nutritional requirements (calories, protein, fats, carbs, iron, cholesterol) based on the user profile.

**4.2 Encoding Scheme**

* **Chromosome Representation:** Each solution (individual) is a vector of real numbers, where each gene xi*xi*​ represents the portion (in grams) of food item i*i* in the meal plan.
* **Bounds:** Each gene is constrained to 0≤xi≤3000≤*xi*​≤300.

**4.3 Fitness Function**

* **Penalty-Based Evaluation:**

The fitness function evaluates each meal plan by combining the total cost, deviation from nutritional targets, and practical considerations (such as small portions and food diversity).**Penalties** are applied for any violation of nutritional or practical constraints. The more a solution deviates from the requirements, the higher the penalty, which reduces its fitness.

* **Implementation:**

See the calculate\_fitness function. The fitness is higher for solutions that are low-cost, nutritionally adequate, and practical, and lower for those that violate constraints.

**4.4 Selection, Crossover, Mutation**

* **Selection:** Tournament selection is used to choose individuals for reproduction, favoring those with higher fitness.
* **Crossover:** Simulated Binary Crossover (SBX) combines genes from two parents to produce offspring.
* **Mutation:** Gaussian mutation adds random noise to genes, introducing diversity.

**4.5 Constraint Handling Technique**

* **Portion Bounds:** Enforced directly by clamping gene values after crossover and mutation.
* **Nutritional and Practical Constraints:**

Handled as **soft constraints** using penalty terms in the fitness function. If a solution violates a constraint (e.g., nutrient out of range, too many small portions, lack of diversity), a penalty is added, making it less likely to be selected.

**4.6 Algorithm Flowchart and Logic**

**Algorithm Steps:**

1. **Initialize Population:** Generate a random population of meal plans.
2. **Evaluate Fitness:** Calculate the fitness (with penalties) for each individual.
3. **Selection:** Select individuals for reproduction using tournament selection.
4. **Crossover:** Apply crossover to create new offspring.
5. **Mutation:** Mutate offspring to introduce variation.
6. **Elitism:** Preserve the best individuals from the previous generation.
7. **Repeat:** Iterate steps 2–6 for a set number of generations.
8. **Output:** Return the best meal plan found.

**4.7 Algorithm Parameters and Tuning**

* **Population Size:** 1500 individuals (default).
* **Number of Generations:** 20 (default).
* **Elite Size:** 10 (number of top individuals preserved each generation).
* **Crossover Rate:** 0.8 (probability of crossover).
* **Mutation Rate:** Starts at 0.2 and decreases over generations.
* **Tournament Size:** 250.
* **Portion Bounds:** 0–300 grams per food item.
* **Penalty Weights:** Set in the fitness function for each type of penalty (e.g., 100 for calorie overage, 20 for protein underage, etc.).

Parameters can be tuned to improve convergence or solution quality.

**6. Fuzzification of a Constraint**

**6.1 Selected Parameter for Fuzzification**

* **Parameter:** The selected parameter for fuzzification is **food diversity**, specifically the number of different foods included in a daily meal plan.
* **Rationale:** Instead of requiring a strict, fixed number of different foods per day, a fuzzy approach allows for more flexibility and realistic meal plans.

**6.2 Fuzzification and Membership Functions**

* **Fuzzification:**

The crisp value (target number of different foods per day) is converted into a fuzzy set using a **triangular membership function**.

* **Membership Function:**

For a given diversity d*d*, the membership function is defined as:

μ(d)={0if d≤a or d≥cd−ab−aif a<d<b1if d=bc−dc−bif b<d<c*μ*(*d*)=⎩⎨⎧​0*b*−*ad*−*a*​1*c*−*bc*−*d*​​if *d*≤*a* or *d*≥*c*if *a*<*d*<*b*if *d*=*b*if *b*<*d*<*c*​

where a*a* is the lower bound, b*b* is the target, and c*c* is the upper bound for diversity.

* **Implementation:**

See the functions triangular\_fuzzy\_membership and calculate\_fuzzy\_diversity\_fitness in the code.

**6.3 Defuzzification Method**

* **Defuzzification:**

In this context, defuzzification is not used to produce a crisp output, but rather the **fuzzy membership value itself is used as the fitness** for the diversity constraint. The genetic algorithm tries to maximize this membership value, favoring solutions with diversity close to the target.

**6.4 Impact on the Optimization Results**

* **Flexibility:**

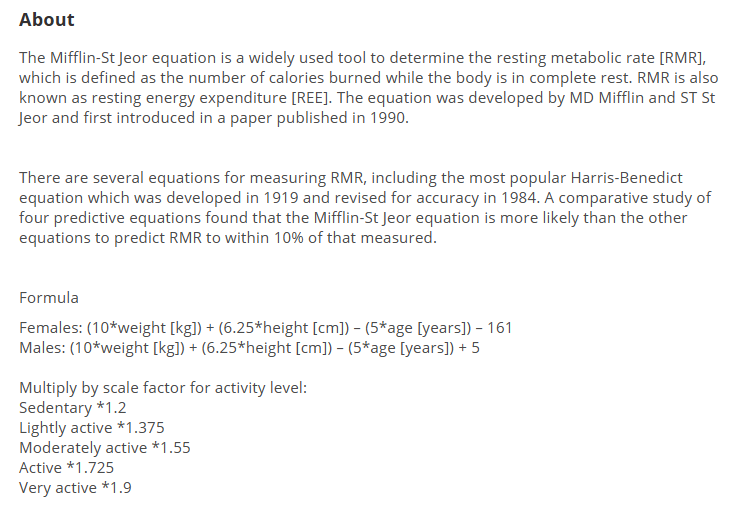
Fuzzification allows the algorithm to prefer meal plans with diversity near the target, but not strictly require an exact number. This leads to more practical and varied solutions.

* **Optimization:**

The use of fuzzy membership as a fitness value smooths the search landscape, making it easier for the genetic algorithm to find good solutions that balance diversity with other objectives.

* **User Experience:**

Users receive meal plans that are diverse, but not rigidly fixed, improving satisfaction and adherence.

[Mifflin-St Jeor Equation](https://reference.medscape.com/calculator/846/mifflin-st-jeor-equation)

-

A screenshot of a text

AI-generated content may be incorrect.[Macronutrient Calculator](https://www.tgfitness.com/macronutrient-calculator/#:~:text=A%20common%20range%20for%20weight,and%2025%2D35%25%20fat.)