Optimizing Dietary Plans Using Evolutionary Algorithms

Iqra Azfar
Department of Computer Science
Habib University
Karachi, Pakistan
ia07614@st.habib.edu.pk

Rabia Shahab
Department of Computer Science
Habib University
Karachi, Pakistan
rs07528@st.habib.edu.pk

Javeria Azfar
Department of ECE
Habib University
Karachi, Pakistan
ja07622@st.habib.edu.pk

***Abstract*—This paper proposes an application of Computational Intelligence (CI) techniques to optimize dietary plans tailored to meet the nutritional needs of athletes engaged in various sports disciplines. Utilizing Evolutionary Algorithms (EAs), we aim to generate personalized dietary plans by selecting foods from a comprehensive database while considering factors such as age, gender, sport, and preferred foods. The outcome is an efficient and effective system capable of providing users with optimized dietary plans for three meals a day, accompanied by a breakdown of nutrient intake proportions. The system's functionality will empower individuals to make informed food choices, ensuring they meet their specific nutritional goals while adhering to dietary restrictions and health requirements.**

***Index Terms*—Evolutionary Algorithm, Computational Intelligence, Diet Plan Optimization**

I. INTRODUCTION

Diet Plan Optimization for athletes focuses on generating a diet plan that meets their nutritional requirements. For this project, we used genetic algorithm to optimize diet plans for athletes based on their age, gender, the sport they play and the foods they prefer. We used two extensive datasets containing information about various food items and the recommended dietary intake for athletes belonging to different sports disciplines. The anticipated outcome of this project is an efficient and effective system for generating personalized dietary plans tailored to meet specific nutritional goals and constraints for individuals belonging to different demographic groups. Users will be provided with optimized dietary plans along with a breakdown of nutrients for each meal, facilitating informed food choices. A proper breakdown of their 3 meal plan for the day will be given with all the ingredients they are required to eat at each time, and graphs to show the amount of each nutrient they would be consuming eg. pie charts to show nutrient intake proportions.

A. Motivation and related work

Adequate nutrition plays a crucial role in bolstering overall health. A well-rounded dietary plan not only enhances physical well-being but also serves to uplift mental health. Moreover, proper nutrition not only improves immediate health outcomes but also carries long-term implications, especially for athletes, extending well beyond their competitive careers.

Over the course of research for this project we encountered this paper [1] which employs a Genetic Algorithm to optimize food portion sizes based on the Guidelines to Balanced Nutrition, providing recommendations for healthy food consumption patterns and potentially informing national food consumption policies. While this approach successfully addresses the optimization of diet based on a standard guideline, we decided to focus on the dietary needs of athletes. By integrating Computational Intelligence techniques, specifically Evolutionary Algorithms (EAs), we aim to develop a system that considers various factors, including age, gender, sport type, and food preferences, to generate personalized dietary plans. Our motivation stems from the desire to provide athletes with nutritionally balanced dietary solutions tailored to their specific needs.

Syahputra [2] also investigated how a genetic algorithm optimized weekly diet schedules for diabetes patients. They took into account each patient's specific caloric and nutritional needs based on factors like age, weight, activity level, and allergies. The algorithm utilized the Harris-Benedict equation to calculate caloric requirements and tailored meals accordingly. The solution aimed to balance required nutrition and improve patient adherence to suitable diets. Overall, their approach demonstrated the potential for personalizing diabetic meal planning using genetic algorithms.

Moreover, to broaden our research in order to attain the most optimized results, we decided to take a multi-objective inspired Genetic Algorithm(MOGA) inspired approach [3] where weights were assigned to each nutritional value so it could focus on optimizing that the most. The analysis and comparisons of both techniques will be shown below.

II. TECHNICAL BACKGROUND

Genetic algorithms belong to the category of optimization algorithms, designed to identify the most optimal solution to a given computational problem by either maximizing or minimizing a specific function. They fall under the broader field of evolutionary computation, which mimics biological processes like reproduction and natural selection to identify the most ”fit” solutions. Similar to biological evolution, genetic algorithms involve elements of randomness, but they also provide control over the degree of randomness in the optimization process. Within each generation, genetic algorithms apply three fundamental genetic operators—selection, crossover, and mutation—to each chromosome, each with certain probabilities.

A. Procedure

1) EA Implementation using Bounded Knapsack: The Knapsack Problem and the Evolutionary Algorithm involve selecting items to optimize a task. In the Knapsack Problem, items are chosen based on value and weight to maximize value without exceeding a weight limit. Conversely, the Evolutionary Algorithm selects foods based on nutritional content and preferences to meet dietary requirements, both subject to constraints: a weight limit in the former and nutritional needs in the latter.

Genetic Algorithm follows the following general procedure:

1) **Initialization:** We start by creating a random population of n chromosomes, which represent potential solutions to the problem.
2) **Fitness Evaluation:** Assess the fitness (suitability) of each chromosome in the population based on a predefined fitness function.
3) **Generating New Population:** Repeat the following steps until a new population is formed:
   - **Selection:** Choose two parent chromosomes from the population based on their fitness, favoring those with higher fitness.
   - **Crossover:** With a certain probability, combine genetic material from the selected parents to create offspring (children). If no crossover occurs, the offspring remain identical to the parents.
   - **Mutation:** Introduce changes to the offspring's genetic material at random positions, with a specified probability for each locus.
   - **Incorporation:** Add the newly created offspring to the new population.
4) **Replacement:** Use the newly generated population to continue running the algorithm.
5) **Termination Check:** If the termination condition is met, stop the algorithm and return the best solution found in the current population.
6) **Iteration:** If the termination condition is not met, repeat the process from step 2.

2) MOGA Inspired Implementation: A multi-objective inspired Genetic Algorithm (GA) was implemented, wherein weights were assigned to each set of parameters during fitness computation. These parameters primarily focused on nutritional values, including fats, carbohydrates, and proteins. By assigning weights to these parameters, the algorithm aimed to balance multiple objectives simultaneously, allowing for the optimization of dietary plans that prioritize specific nutritional goals.

Incorporating the concept of Pareto points, the algorithm also evaluated the entire population of solutions. Each solution was assessed based on its Pareto optimality, a state where no single objective can be improved without degrading some of the other objectives. This approach helped in identifying a set of potential solutions that represent the trade-offs among conflicting objectives and are equally good.

The algorithm then normalized these solutions according to the assigned weights. This normalization process ensured that each objective was given its due importance while calculating the fitness. This approach enabled the algorithm to generate solutions that strike an optimal balance between different nutritional requirements, catering to diverse dietary needs and preferences. It also ensured a fair representation of the solution space, providing a diverse set of optimal solutions for different dietary needs and preferences

1) **Simplicity and Directness of the Penalty System**:
   - The first fitness function penalizes deviations from nutritional targets straightforwardly.
   - It employs a clear penalty mechanism where exceeding nutrient thresholds incurs severe penalties, effectively reducing the fitness score to zero.
   - Additionally, the penalty calculation is linear, directly reducing scores based on the percentage deviation from target values. This simplicity facilitates easy understanding and prediction of the fitness function's behavior.
2) **Complexity and Overhead in R-Method Implementation**:
   - The R-method, inspired by Pareto optimization, introduces complexity through Pareto dominance calculations, ranking, and weighting.
   - Unlike the straightforward penalty system, the R-method involves multi-step processes such as calculating deviations, ranking them, and applying weighted scores, which can complicate fitness evaluation.
   - The assignment of weights in the R-method can be subjective and may require trial and error or expert knowledge. This could lead to bias in the results, depending on how the weights are assigned.
   - While the R-method can help find a set of good solutions, there is no guarantee that these solutions are the absolute best. There may exist other solutions that are better in terms of one or more objectives, but these might not be identified by the R-method.
3) **Interpretability and Practical Application**:
   - The direct penalty system offers high interpretability, allowing diet planners or nutritionists to easily understand why certain diets are favored based on clear numeric thresholds and penalties.
   - Conversely, the R-method, while offering a more nuanced view, may require more effort to interpret and justify the rankings and selections it produces.
4) **Reliability and Robustness**:
   - The robustness of the penalty system lies in its simplicity and immediate feedback given by the penalties,

ensuring clear selection pressure towards target values.
- However, the reliance on Pareto scores and composite scores in the R-method can sometimes lead to less stable selections if not well-tuned or if the population lacks diversity.

III. Problem Description

A) **Nutritional Goals**
- **Targets:** These parameters represent the specific nutritional targets for protein, carbohydrates, and fats that the optimized dietary plans aim to achieve.
  Usage: The genetic algorithm evaluates the fitness of each dietary plan based on its ability to meet these target nutritional goals. Deviations from the targets contribute to the fitness score, guiding the search towards plans that better align with the athlete's nutritional requirements.

B) **Preference and All Foods**
- **Preference:** These variables store information about the preferred ingredients selected by the user and the complete dataset of available food items, respectively.
  Usage: User preferences guide the selection of ingredients considered for dietary plans, ensuring that the generated plans include preferred foods. All Foods dataset provides the necessary nutritional information for all available food items.

IV. Problem Formulation

A. Chromosome Structure

- **Length:** The length of the chromosome corresponds to the total number of available food items or ingredients that can be included in the diet plan.
- **Genes:** Each gene in the chromosome represents the quantity or units of a specific food item to include in the diet. The value of each gene indicates the number of units of the corresponding food item to include.
- **Encoding:** The chromosome is typically encoded as a list or array of integers, where each integer represents the quantity of a specific food item. The index of each integer corresponds to the position of the food item in the list of available food items or ingredients.
  Example: For example, if the chromosome has a length of 10 and the value of the genes at indices 0, 3, and 7 are 2, 1, and 0 respectively, it means:
  - Include 2 units of the food item at index 0.
  - Include 1 unit of the food item at index 3.
  - Exclude the food item at index 7 from the diet plan
- **Visualization:** Chromosome: [2, 0, 0, 1, 0, 0, 0, 0, 1, 0]
- **Interpretation:** The above chromosome indicates a diet plan that includes:
  - 2 units of the food item at index 0.
  - 1 unit of the food item at index 3.
  - 1 unit of the food item at index 8.

  The remaining food items are not included in the diet plan (quantities are set to 0).

B. Fitness Function

The fitness function is initialized with three main variables chromosome_protein, chromosome_fat, and chromosome_carbs to track the total protein, fat and carbohydrate content of the chromosome. The function evaluates the quality of a given dietary plan represented by a chromosome. It begins by summing up the total protein, fat, and carbohydrate content of the chromosome by iterating through each ingredient and calculating its nutritional contribution. These values are stored in variables 'chromosome_protein', 'chromosome_fat', and 'chromosome_carbs'. The function initializes a perfect score of 100 and defines thresholds with a 10% flexibility over the target values for each nutrient. If any nutrient value exceeds its respective threshold significantly, the function returns a score of 0, indicating that the chromosome is not a valid solution. Otherwise, it calculates a score based on how close each nutrient value is to its target, deducting points for deviations with a maximum deduction of 25 points per nutrient. The function ensures that the calculated score does not go negative. The function encourages chromosomes that closely meet the target nutritional requirements while penalizing deviations, guiding the genetic algorithm towards finding optimal dietary plans.

C. Crossover

For this problem we focused on striking a balance between exploration and exploitation to avoid premature convergence to sub-optimal solutions and ensure a thorough search of the solution space. We tested four different crossover methods: SBX, Two-Point, Uniform, and a custom New Crossover. The SBX method, simulating binary crossover, and the New Crossover method performed the best. They both introduced variations in offspring chromosomes, crucial for exploration, while also preserving good solutions from the parents, promoting exploitation. In contrast, the Two-Point and Uniform methods had limitations in promoting exploration due to their probabilistic nature, which sometimes led to offspring inheriting sub-optimal traits. Our choice of crossover method significantly impacted the algorithm's performance, with the SBX and New Crossover methods striking the ideal balance between exploration and exploitation, enabling effective exploration of the solution space while maintaining solution quality.

*1) **New Crossover**:* The New Crossover method randomly selects a crossover point in parent chromosomes and combines genes from one parent with the other after this point. This approach facilitates high exploration levels by generating offspring with varied values. However, it carries the risk of losing good solutions if the added genes result in inferior offspring. The method balances exploration and exploitation effectively through the randomness of the crossover point selection and gene addition, ensuring a diverse exploration of the solution space while preserving solution quality.

*2) **SBX Crossover (Simulated Binary Crossover**:* The SBX method emulates one-point crossover in binary-encoded genetic algorithms, producing two offsprings for two parent solutions. It introduces randomness by generating a random number for each corresponding gene pair in the parents. If the number is below 0.5, a larger 'beta' value is computed, resulting in offspring closer to the parents (exploitation). Conversely, if the number exceeds 0.5, a smaller 'beta' value is calculated, leading to offspring farther from the parents (exploration). The parameter 'eta_c' controls the distribution of offspring around the parents, with a larger 'eta_c' favoring exploitation and a smaller one promoting exploration. Additionally, the method ensures offspring remain within problem bounds.

D. Mutation

During mutation indices in a chromosome are randomly selected based on a predefined mutation rate. If chosen for mutation, a random integer within the allowable range for that index is generated. This process introduces diversity into the population, aiding exploration of the solution space. The mutation rate parameter determines the frequency of mutation, influencing the balance between exploration and exploitation. By altering specific indices, mutation can potentially improve or degrade the fitness of solutions, impacting the search for optimal solutions.

E. Parent and Survivor Selection

*1) **Parent Selection**:* The population is initialized randomly of potential solutions, or chromosomes, taking in consideration the user's preferences and dietary requirements. For parent selection, two approaches are implemented. The **Truncation Selection** sorts the population based on each chromosome's fitness, which indicates how well it satisfies the dietary requirements. It then selects the top-performing individuals as parents for the next generation. **Fitness Proportional Selection** selects individuals from the population with a probability proportional to their fitness. Chromosomes with higher fitness values have a greater chance of being selected as parents. These selection mechanisms ensure that the genetic algorithm explores promising areas of the solution space by favoring solutions that perform well, ultimately guiding the population towards optimal or near-optimal solutions over successive generations.

V. EXPERIMENTS AND RESULTS

*1) **Bounded Knapsack Problem:**:* The Bounded KnapSack problem was run for 5000 generations. This was the optimal amount of generations to yield the best results. A lower number of generations led to premature convergence and a higher number took significant amount of time to show results. The result of this top 10 best (most optimal) diet plans. All of these diet plans are optimal hence the user get to choose the plan that suits them the best.

Fig. 1 shows an optimal solution as it the algorithm achieves the target set. in Fig. 2 we see an almost uniform distribution of nutrients hence the devised diet plan is not only providing optimal results it is also maintaining a good balance between all the basic nutrients.

On the flip side here we have Fig. 3, which clearly shows that the meal plan suggested by the algorithm fails to achieve the set targets and as see in Fig. 4 we also are not provided with a good distribution amongst the nutrients. This problem was tackled through a trial and error method using different parent and child selection schemes, crossover combinations and testing out the algorithm with varying number of generations.

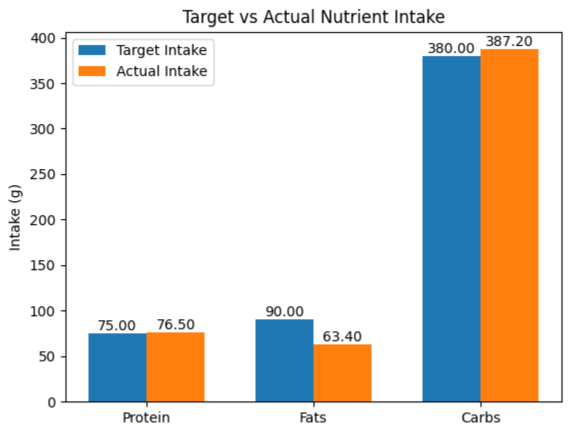

Fig. 1. Good Knapsack Optimization results

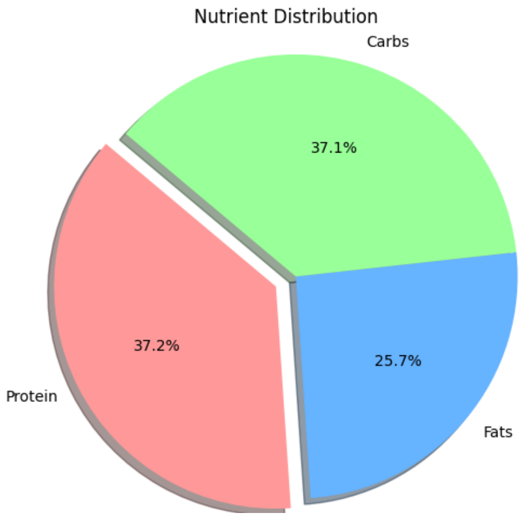

Fig. 2. Good Knapsack Distribution of Nutrients

*2) **MOGO using R-method**:* We ran the MOGO based implementation for 5000 generations. The R-method, is a sophisticated approach designed for multi-objective optimization problems, aiming to find a set of Pareto-optimal solutions that balance several conflicting objectives. The method integrated concepts such as Pareto dominance to provide a comprehensive optimization framework and utilize the relative weights assigned to each objective, to find the optimal solution. As observed in Fig. 5 in comparison to the bounded Knapsack problem MOGO runs less optimally as it does not reach its target for Fats and Carbs by a huge margin, however it still does give us a good enough nutrient distribution as seen in Fig. 6.

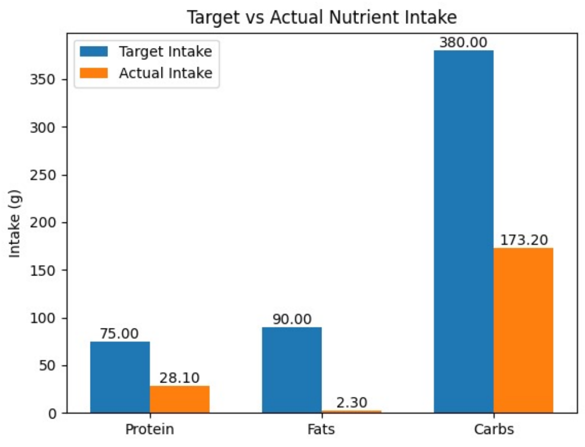

Fig. 3. Bad Knapsack Optimization Results

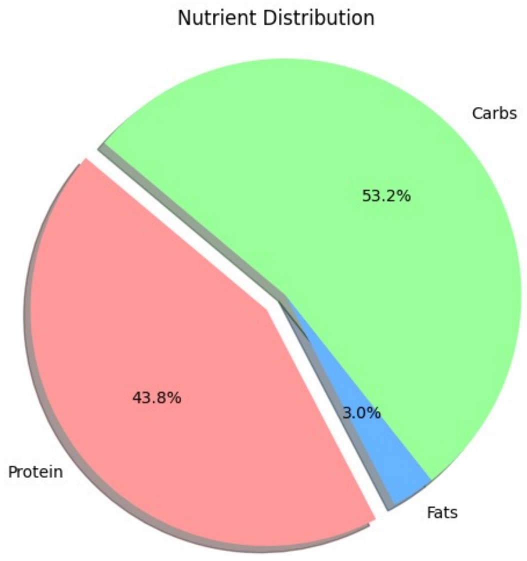

Fig. 4. Bad Knapsack Distribution of Nutrients

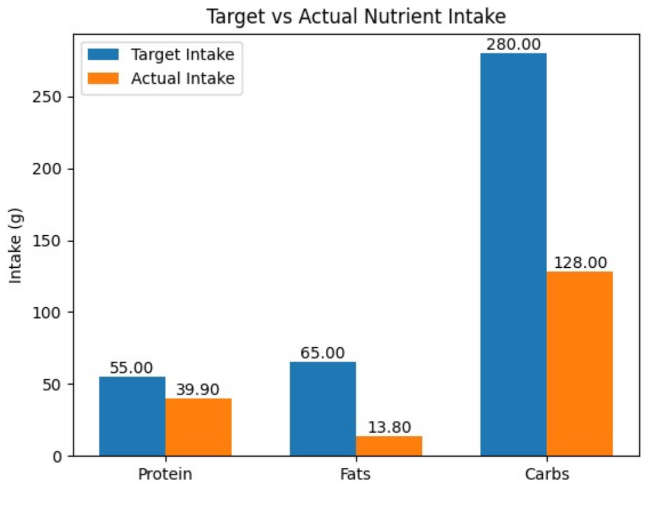

Fig. 5. MOGO Optimization results

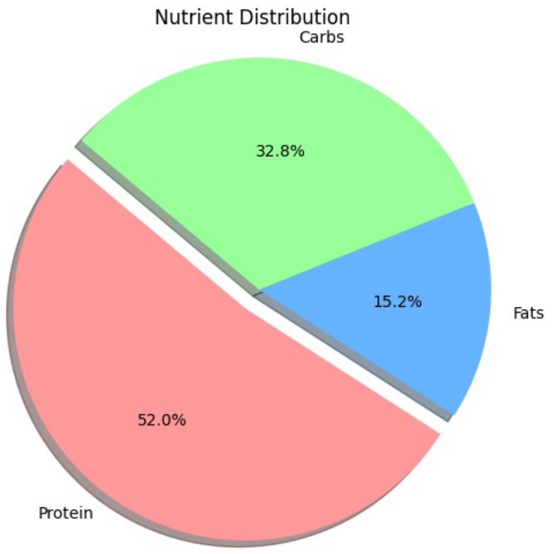

Fig. 6. MOGO Distribution of Nutrients

VI. Challenges

The main challenge during this project was the R-Method implementation. The complexity of the approach using the R-method and its fitness evaluation in the context of a diet optimization problem presents several challenges and limitations, which might make it less suitable than more straightforward, direct methods. The fitness function calculates absolute deviations for each nutrient (protein, fat, carbs) from their targets. This part is straightforward but becomes complex when integrated with other components of the R-method. Furthermore, when a chromosome exceeds the target by more than 10 units for any nutrient, all its contributions are zeroed out. This harsh penalty might simplify handling in some scenarios but could lead to losing potentially good genetic material due to minor overages.

Pareto Dominance: Pareto dominance calculations in diet optimization can be overly complex and might overlook solutions that slightly deviate from one objective but excel in another. Normalizing ranks could reduce the pressure toward better solutions, while weighted sum methods might obscure individual contributions. Direct optimization methods, focusing on how closely a diet meets nutritional targets, are often more practical and easier to interpret for dietitians and end-users.

Using the Pareto dominance technique also slowed down our algorithm quite a bit in comparison to the bounded knapsack implementation.

VII. Conclusion

This paper introduces an innovative application of Computational Intelligence techniques, specifically evolutionary algorithms (EA) to optimize diet plans tailored to meet the unique nutritional needs of athletes across various sports disciplines. We considered factors such as age, gender, sport type and food preferences to generate personalized diet plans to meet specific nutritional needs. Through experiments, we showed that our approach effectively optimizes dietary plans, ensuring they align closely with target nutritional requirements. While we explored a sophisticated multi-objective approach, it posed challenges in complexity and computational overhead. Overall, our research highlights the potential of computational intelligence in revolutionizing personalized nutrition planning for athletes, with implications for improving overall health and performance.

VIII. FUTURE WORK

The multi objective approach leaves us with a huge opportunity to expand this project to include other factors such as cost and specific health issues such as diabetes or cholesterol. The current implementation optimizes meal plans per day, however we can expand our chromosome to produce meal plans for the the entire week. We can also add a collaborative decision support system that involves nutritionists, coaches and other athletes in the dietary planning process which can help us form more holistic and well-informed recommendations.

IMPORTANT LINKS

1) The GitHub project repository for the project can be found at: https://github.com/rabia-s/CS-451-CI-Project-Diet-Optimization-Plan.
2) The references to the athlete information for the project can be found at: https://docs.google.com/document/d/10QWXjx1jTbD2D71qFtSVuxtueis51k1SzIicEIHSMO8/edit?usp=sharing
3) https://github.com/anshulwadhawan/GA
4) https://github.com/imaqn/genetic-algorithm
5) https://github.com/mikemayuare/stiglers-diet-ga
6) https://github.com/scottwait/DietProblem

REFERENCES

[1] Adriyendi, Y. Melia (2021). Optimization using genetic algorithm in food composition. *International Journal of Computing and Digital Systems*, **10**(1), 1019–1029. doi:10.12785/ijcds/100191.

[2] Syahputra, M. F., Felicia, V., Rahmat, R. F., and Budiarto, R. (2017). Scheduling Diet for Diabetes Mellitus Patients using Genetic Algorithm. Journal of Physics: Conference Series, 801, 012033.

[3] RV, Rao., & RJ, Lakshmi. (2021). Ranking of pareto-optimal solutions and selecting the best solution in multi- and many-objective optimization problems using R-method. *sciencedirect.com*.