Algoritmo Genético Aplicado ao Problema da Dieta

Carlos Pedroso[1], **Giovanni R. da Silva**[1], **Roberta S. Tomigian**[1]

[1]Departamento de Informática
Universidade Federal do Paraná (UFPR) – Curitiba – PR – Brazil

{capjunior,grsilva,rst17}@inf.ufpr.br

Abstract. *A nutritional diet is an important factor for human development, playing a significant role in achieving better health. However, all this dietary elaboration presents problems related to food choice, amount of ingested calories, time, and mainly optimization to obtain a balanced diet. Thus, developing a diet involves huge calculations and analysis of food compositions and nutritional elements. One way to optimize this problem is through the application of Genetic Algorithms (GA) inspired by biological models capable of solving problems quickly and accurately. Thus, the present work applies GA to the diet problem for mono and multiobjective. The results obtained by simulation demonstrate that the solution was able to solve the problem quickly and accurately, proving the efficiency of GA.*

Resumo. *A dieta nutricional é um fator importante para o desenvolvimento humano, tendo papel significativo na obtenção da melhor saúde. Entretanto, toda essa elaboração dietética apresenta problemas relacionados à escolha de alimentos, quantidade de calorias ingeridas, tempo e principalmente otimização para obtenção de uma dieta balanceada. Assim, desenvolver uma dieta envolve cálculos e análises enormes de composições de alimentos e elementos nutritivos. Um forma de otimizar este problema é através da aplicação de Algoritmos Genéticos (GA) inspirados em modelos biológicos capazes de resolver problemas com rapidez e precisão. Assim, o presente trabalhos aplica GA ao problema da dieta para mono e multiobjetivos. Os resultados obtidos por simulação demonstram que a solução foi capaz de resolver o problema com rapidez e precisão, comprovando a eficiência do GA.*

1. Introdução

A dieta nutricional é um fator importante para o desenvolvimento humano, tendo papel significativo na obtenção da melhor saúde. Combinada com outros fatores como atividades físicas, descanso e momentos de felicidade, uma dieta adequada consegue transformar vidas [Steptoe and Wardle 2004]. Programações dietéticas científicas e racionais podem não só melhorar significativamente a qualidade de vida, mas também pode determinar se um indivíduo irá ou não desenvolver doenças como câncer, doenças cardiovasculares, obesidade ou diabetes no futuro. Design tradicional de métodos de receitas dietéticas incluem método experimental, método de troca de alimentos e método de cálculo rápido de peso [Lv 2009, Fu and Wang 2007]. Entretanto, toda essa elaboração dietética apresenta problemas relacionados a escolha de alimentos, quantidade de calorias ingeridas, tempo e principalmente otimização para obtenção de uma dieta balanceada. Assim, desenvolver uma dieta envolve cálculos e análises enormes de composições de alimentos e elementos

nutritivos. Estes métodos simplificados tradicionais são incapazes de atender aos requisitos de precisão e velocidade requeridos. Assim, o desenvolvimento de algoritmos e modelos que introduzam teorias e métodos no campo da computação inteligente focado neste problema são necessários.

Ao longo dos anos diversos pesquisadores exploraram o problema da dieta [Harrop and Marlatt 2010], um deles George Joseph Stigler, buscou resolver problema da dieta através da otimização, no final da década de 30. Aplicando a dieta ideal para tentar satisfazer a preocupação do exército americano, que procurava a maneira mais econômica de alimentar as suas tropas, garantindo em simultâneo, certos requisitos nutricionais. Além disso, diversos métodos, modelos matemáticos foram aplicados na tentativa de resolução e otimização do problema, sejam focados na minimização ou maximização da dieta. Entretanto, o aprimoramento da computação e desenvolvimento de algoritmos precisos e eficientes como os genéticos, novos estudos acerca do problema têm alcançado resultados importantes.

Os Algoritmos Genéticos (GA) são uma família de modelos computacionais inspirados na evolução sendo proposto por John Henry Holland em 1975. Em um uso mais amplo do termo, um algoritmo genético é qualquer modelo baseado em população que usa operadores de seleção e recombinação para gerar novos pontos de amostra em um espaço de pesquisa. Os GA têm sido empregados para resolução de diversos problemas na literatura, sejam de otimização, balanceamento, teoria dos grafos ou problemas mais específicos como caixeiro viajante ou o problema da mochila. Assim, neste trabalho abordamos o problema de otimizar uma dieta semanal através da aplicação de um algoritmo genético que indica os alimentos e quantidade que devem ser consumidas conforme o número base de calorias definidas pelo usuário.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta os fundamentos para entender os GAs. A Seção 3 apresenta o problema, modelagem e algoritmo proposto. A Seção 4 detalha a avaliação e os resultados obtidos. Por fim, a Seção 5 apresenta a conclusão e direcionamentos futuros.

2. Fundamentos

Esta seção tem por objetivo introduzir o conceito de Algoritmos Genéticos (GAs) na Subseção 2.1, que será utilizado na proposta de solução do problema da dieta. A Subseção 2.2 discorre sobre a abordagem multiobjetivo empregada na proposta. Por fim, a Subseção 2.3 apresenta o estado da arte de GAs em resolução de problemas.

2.1. Algoritmos Genéticos (GAs)

De acordo com [Whitley 1994], o Algoritmo Genético (GA) é uma família de modelos computacionais inspirados na evolução Darwiniana sendo proposto por John Henry Holland em 1975. Por isso, o princípio da natureza onde o GA se inspira é a seleção que privilegia os indivíduos mais aptos com maior longevidade, e portanto, com maior probabilidade de reprodução. Indivíduos com mais descendentes tem maior probabilidade de perpetuar sua genética nas próximas gerações.

Em um uso mais amplo do termo, um GA é qualquer modelo baseado em população que usa operadores de seleção e recombinação para gerar novos pontos de amostra em um espaço de pesquisa. Na computação, esses algoritmos são utilizados para

buscar a melhor solução para determinado problema, através da evolução de populações de soluções codificadas através de cromossomos artificiais. Com base em [Mirjalili 2019] e explicadas nas subseções a seguir, as etapas do GA são: população inicial, seleção, recombinação e mutação.

População inicial: o algoritmo inicia com uma população aleatória, que pode ser gerada a partir de uma distribuição aleatória Gaussiana, para aumentar a diversidade. Essa população é composta por várias soluções, que representam cromossomos de indivíduos. Cada cromossomo tem um conjunto de variáveis, que simula os genes. O principal objetivo nessa etapa é espalhar as soluções pelo espaço de busca uniformemente para aumentar a diversidade da população e melhorar a probabilidade de encontrar regiões promissoras.

Seleção: na natureza, os indivíduos mais aptos têm uma oportunidade maior de obter comida e acasalar. Isso faz com que seus genes contribuam mais na produção da próxima geração da mesma espécie. Inspirando-se nessa ideia, o GA emprega uma roleta para atribuir probabilidades a indivíduos e selecioná-los para criar a próxima geração proporcional aos seus valores de avaliação (*fitness*).

Recombinação: após selecionar os indivíduos através de um operador de seleção, eles devem ser recombinados para criar a nova geração. Na natureza, os cromossomos nos genes de um homem e de uma mulher são combinados para produzir um novo cromossomo. Isso é simulado pela combinação de duas soluções (pais) selecionadas pela roleta para produzir duas novas soluções (filhos) no GA.

Mutação: a mutação é o último operador evolutivo, onde um ou vários genes são alterados após a criação de soluções dos filhos. A taxa de mutação é definida como baixa em GA porque altas taxas de mutação convertem GA em uma pesquisa aleatória primitiva. O operador de mutação mantém a diversidade da população introduzindo outro nível de aleatoriedade. Na verdade, esse operador impede que as soluções se tornem semelhantes e aumenta a probabilidade de evitar soluções locais no GA.

2.2. Multiobjetivo

Nos Algoritmos Genéticos Multiobjetivos, a cada geração, ou iteração, tem-se um conjunto de indivíduos, ou soluções-pais. Para gerar uma nova população (soluções-filho), os operadores genéticos (tais como recombinação e mutação) são aplicados sobre as soluções-pai. Os métodos de otimização multiobjetivo têm dois objetivos principais: minimizar a distância entre a frente não dominada e a frente pareto ótimo e encontrar um conjunto de soluções diversificadas [Barbosa et al. 2010]. Em contrapartida, os algoritmos genéticos multiobjetivos buscam solucionar dois problemas: o procedimento de avaliação e seleção dos algoritmos para garantir uma busca eficiente para o conjunto pareto ótimo e uma forma de manter a diversidade da população evitando a convergência prematura. Além disso, a literatura apresenta uma diversidade de problemas que podem ser resolvidos por GA multiobjetivo, sejam problemas de computação ou mesmo desafios triviais do dia-a-dia. Soluções multiobjetivo atuam no equilíbrio e proporção dos resultados. Alguns dos problemas que podem ser resolvidos por GA multiobjetivos estão listados na subseção a seguir.

2.3. Aplicações em Problemas de Otimização

Ao longo dos anos os Algoritmos Genéticos (GAs) têm sido aplicados na resolução de diversos problemas em diversas áreas, focando sempre na otimização da resolução do problema. Dentre as aplicações, destaca-se o problema da inversão sísmica. Este problema é extremamente suscetível à aplicação de GAs, pois sua função objetivo é extremamente irregular, sendo altamente não linear, possuindo muitos mínimos e máximos locais e podendo apresentar descontinuidades. Outra aplicação dos GAs é para avaliação de músicas (sequências de acordes) tocadas em arquivos MIDI. No caso, os indivíduos da população foram definidos em grupos de quatro vozes (soprano, contralto, tenor e baixo) ou coros. Cada um é avaliado segundo três critérios: melodia, harmonia e oitavas. Problemas em telecomunicações, onde uma série de soluções promissoras e situações reais, são solucionados utilizando a aplicação de GA. Na medicina, por exemplo, os GAs foram utilizados para auxiliar na elaboração de uma escala de trabalho dos médicos plantonistas neonatologistas da maternidade. O objetivo pretendido foi o de auxiliar na solução da escala de trabalho dos médicos, em como diminuir o esforço e o desgaste humanos para a confecção do plantão. Além dessas aplicações citadas, existe uma infinidade de aplicações que se beneficiam dos GAs, seja no setor financeiro, planejamento urbano, saúde, esportes e múltiplas subáreas da computação na sua totalidade.

3. Proposta

Esta seção inclui a definição do problema tratado, a otimização de um plano de refeições, na Subseção 3.1, assim como a modelagem deste problema para a utilização em algoritmos genéticos na Subseção 3.2. A Subseção 3.3 apresenta a descrição da solução proposta com o algoritmo implementado. Por fim, a Subseção 3.4 detalha um exemplo do problema tratado e do funcionamento da solução.

3.1. Definição do Problema

Uma pessoa que dispõe de vários produtos para se alimentar precisa escolher uma combinação de alguns deles para suprir sua necessidade calórica, considerando os macronutrientes, como as proteínas, os carboidratos e a gordura.

3.2. Modelagem do Problema

Com base em [Korstanje 2020], ao otimizar um plano de refeições, há muitas coisas a serem consideradas. Este trabalho aborda um plano de refeições semanais para uma pessoa baseado em quantas calorias serão consumidas e as porcentagens dessas calorias em proteínas, carboidratos e gordura. Além desses valores, o programa de otimização também precisa de uma lista predefinida de produtos que eles podem escolher, com informações de calorias totais e quantidades em gramas de proteínas, gordura e carboidratos, conforme mostra a Tabela 1.

Tabela 1. Definição das variáveis de cada produto

Índex	Produto	Calorias	Proteínas (g)	Gordura (g)	Carboidratos (g)
$id \in \mathbb{N}$	$string$	$cal \in \mathbb{N}$	$prot \in \mathbb{N}$	$gor \in \mathbb{N}$	$carb \in \mathbb{N}$

A quantidade de calorias total como meta a serem consumidas no período precisa ser definida, assim como as porcentagens de proteínas, gorduras e carboidratos da dieta. O Departamento de Agricultura dos Estados Unidos (USDA, do inglês *United States Department of Agriculture*) indica que cada grama de carboidrato tem 4 calorias, cada grama de proteína também tem 4 calorias e cada grama de gordura tem 9 calorias [USDA 2021]. As porcentagens definidas como meta de cada macronutriente permitem calcular o valor de calorias em relação ao total e em conjunto com o valor da quantidade de calorias por grama do respectivo macronutriente, é possível calcular a quantidade de gramas desejadas de proteínas, carboidratos e gordura.

$$C = \{g_1, \ \dots, g_n\} \mid n, g_i \in \mathbb{N} \tag{1}$$

Os cromossomos dos indivíduos do algoritmo genético são representados por um conjunto de tamanho igual ao número de produtos disponíveis (n), onde cada posição do conjunto representa a quantidade escolhida do produto correspondente ao índex da Tabela 1, como detalha a Equação 1. Cada indivíduo representa uma lista de compras a ser otimizada no processo evolutivo.

3.3. Descrição do Algoritmo

O objetivo do programa de otimização é encontrar uma lista de produtos para o período, cujas calorias e macronutrientes sejam o mais próximo possível dos valores da meta, utilizando um algoritmo genético. Primeiramente, uma população inicial grande é definida de maneira randômica, onde cada indivíduo representa uma lista de compras, conforme descrito na Subseção 3.2. Então, inicia-se o processo evolutivo selecionando os melhores indivíduos ao calcular o quão longe eles estão das quantidades desejadas de calorias. Em seguida, esses melhores candidatos são recombinados entre eles para fornecer novas listas de compras que são combinações aleatórias dos pais com mutação também aleatória. A estratégia de cruzamento adotada é a *two points* e a estratégia de mutação é a *flip bit*. A estratégia de seleção elege o melhor indivíduo em um torneio de 3 escolhidos aleatoriamente. O Algoritmo 1 mostra a evolução em que, caso atinja o objetivo ou após um número definido de gerações, o melhor indivíduo da população final é selecionado, garantindo uma das melhores listas de compras possíveis.

Algoritmo 1 Evolução da dieta no decorrer das gerações

```
1: Pop ← GenerateInitialPopulation()
2: while GENERATIONS do
3:     Offspring ← SelectNextGeneration(Pop)
4:     Offspring ← Crossover(Offspring)
5:     Offspring ← Mutate(Offspring)
6:     Offspring ← Evaluate(Offspring)
7:     Pop ← Offspring
8: Best ← SelectBest(Pop)
```

A implementação ainda considera duas abordagens, sendo a primeira uma otimização apenas da quantidade de calorias, e a segunda multiobjetivo que considera

tanto o valor total de calorias quanto a proporção dos macronutrientes. O processo evolutivo é o mesmo e a única distinção entre as abordagens é a função de avaliação (*fitness*). A primeira versão, mostrada no Algoritmo 2, recupera a quantidade total de proteínas, gordura e carboidratos ao somar os produtos da lista de compras (indivíduo), e depois calcula a quantidade de calorias com os valores definidos pela [USDA 2021]. A abordagem multiobjetivo avalia e busca otimizar as calorias totais e a proporção de proteínas, gordura e carboidratos, conforme apresentado no Algoritmo 3.

Algoritmo 2 Função de avaliação para otimizar as calorias da lista de compras

```
1: procedure EVALUATE(Individual)
2:     TotalProt ← CalcProt(Individual) × 4
3:     TotalFat ← CalcFat(Individual) × 9
4:     TotalCarb ← CalcCarb(Individual) × 4
5:     CalCalc ← TotalProt + TotalFat + TotalCarb
6:     return abs(CalCalc − CalGoal)
```

Algoritmo 3 Função de avaliação multiobjetivo para otimizar as calorias, as proteínas, a gordura e os carboidratos da lista de compras

```
1: procedure EVALUATE(Individual)
2:     TotalProt ← CalcProt(Individual) × 4
3:     TotalFat ← CalcFat(Individual) × 9
4:     TotalCarb ← CalcCarb(Individual) × 4
5:     CalCalc ← TotalProt + TotalFat + TotalCarb
6:     return CalcError(CalCalc, CalGoal,
7:                      TotalProt, ProtGoal,
8:                      TotalFat, FatGoal,
9:                      TotalCarb, CarbGoal)
```

3.4. Exemplos

O exemplo abaixo visa simplificar a atuação do algoritmo genético na escolha dos alimentos para uma dieta com base na quantidade de calorias que se deseja ingerir por dia. Desta forma, se uma pessoa deseja consumir 3 mil calorias por dia totalizando 21 mil calorias por semana com uma porcentagem de proteína de 30%, porcentagem de carboidrato de 50% e com uma porcentagem de gordura de 20%. Os produtos disponíveis estão descritos na Tabela 2. A coluna **(Produto)** define os produtos selecionados para consumo. A coluna de **(Calorias)** apresenta o valor de calorias de cada alimento para a quantidade definida. A coluna de proteína (Proteína g) quantifica os valores de proteína adquiridos para cada alimento. A coluna de gordura **(Gordura g)** define o valor de gordura obtido para cada alimento e a coluna de carboidrato **(Carboidratos g)** quantifica o valor de carboidrato obtido para cada alimento consumido. Dessa forma, a dieta se torna fácil e prática, visto que todos os cálculos e definições de consumo ficam sumarizados para o usuário escolher com base em números e objetivos.

Tabela 2. Lista dos produtos escolhidos

Índex	Produto	Calorias	Proteínas (g)	Gordura (g)	Carboidratos (g)
1	Banana 1u	89	1	0	23
2	Mandarim 1u	40	1	0	10
3	Ananás	50	1	0	13
4	Uva 100g	76	1	0	17
5	Chocolate 1 barra	230	1	13	25
6	Queijo duro 100g	350	28	26	2
7	Queijo macio 100g	374	18	33	1
8	Pesto 100g	303	3	30	4
9	Húmus 100g	306	7	25	11
10	Pasta de berinjela 100g	228	1	20	8
11	Shake proteico	160	30	3	5
12	Hambúrguer vegetariano 1	220	21	12	3
13	Hambúrguer vegetariano 2	165	16	9	2
14	Ovo cozido	155	13	11	1
15	Ovo assado	196	14	15	1
16	Meio pão de baguete	97	3	1	17
17	Pão quadrado 1 fatia	97	3	1	17
18	Pizza de queijo 1u	903	36	47	81
19	Pizza vegetariana 1u	766	26	35	85
20	Leite de soja 200ml	115	8	4	11
21	Achocolatado de soja 250ml	160	7	6	2

O objetivo do programa é otimizar e encontrar uma lista de produtos para um período, cujas calorias e macronutrientes do total do período sejam o mais próximo possível dos valores da meta definida pelo usuário. Desta forma, atendendo todas as necessidades alimentares e energéticas com base em uma escolha de itens otimizada.

Tabela 3. Lista dos produtos escolhidos

Índex	Produto	Mono objetivo	Multiobjetivo
1	Banana 1u	5	5
2	Mandarim 1u	3	8
3	Ananás 100g	1	0
4	Uva 100g	6	2
5	Chocolate 1 barra	0	4
6	Queijo duro 100g	1	2
7	Queijo macio 100g	0	0
8	Pesto 100g	7	9
9	Húmus 100g	3	0
10	Pasta de berinjela 100g	5	1
11	Shake proteico	7	2
12	Hambúrguer vegetariano 1	1	3
13	Hambúrguer vegetariano 2	5	7
14	Ovo cozido	4	1
15	Ovo assado	7	6
16	Meio pão de baguete	9	7
17	Pão quadrado 1 fatia	0	5
18	Pizza de queijo 1u	1	1
19	Pizza Vegetariana 1u	3	6
20	Leite de soja 200ml	9	2
21	Achocolatado de soja 250ml	8	5

A Tabela 3 sumariza a lista de produtos escolhidos e sua quantidade depois da execução do algoritmo genético para o multi e mono objetivo. Observa-se que as quantidades variam conforme a categoria de objetivo desejado pelo usuário, dessa forma, ele consegue otimizar a compra e obter maior precisão na quantidade de alimentos necessários em sua dieta.

4. Avaliação

O algoritmo genético foi implementado e testado no sistema operacional *Windows 10 (v21H1, b19043.1110, x64)*, processador *Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz* com 16.0 GB (15.8 GB usable) de memória. Ele foi desenvolvido na linguagem *Python 3.9.5* [Foundation 2021a], utilizando as bibliotecas:

- Pandas, que fez a manipulação e análise dos dados. [McKinney et al. 2010]
- *DEAP* (*Distributed Evolutionary Algorithms in Python*), é um framework que implementa algoritmos evolutivos [Fortin et al. 2012]
- Numpy, realiza cálculos matemáticos [Harris et al. 2020]
- Time, fornece funções relacionadas ao tempo, utilizado para medir o tempo de execução [Foundation 2021b]
- Matplotlib, plota gráficos [Hunter 2007]

Para obter resultados com mais precisão, calculou-se a média de 35 execuções do mesmo algoritmo em cada cenários de 50, 100 e 500 gerações, conforme apresentado a Figura 1. Além disso, para todos os testes foi considerado o cenário que tinha como objetivos de porcentagem de macronutrientes como pode ser observado na Figura 2.

Figura 1. Metodologia de execução

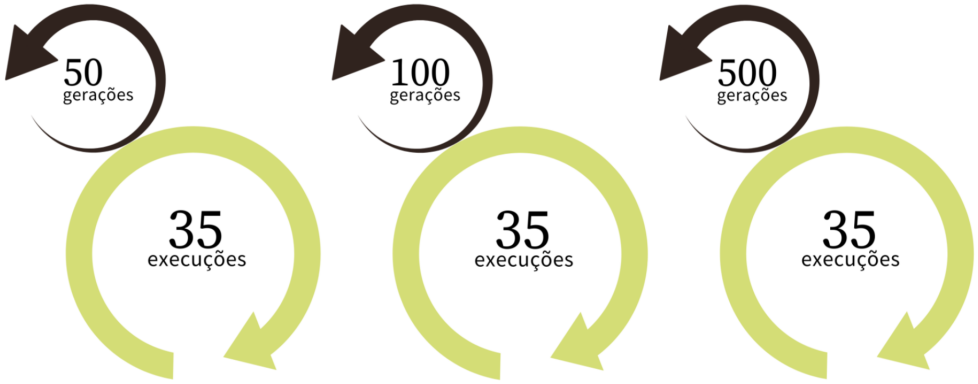
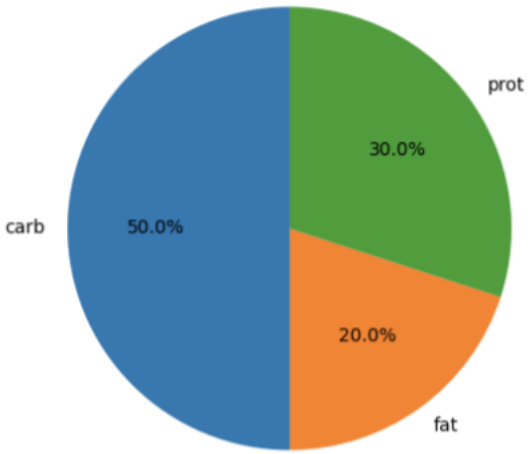


Figura 2. Porcentagem da meta



Podemos perceber nesse caso que a quantidade de carboidratos necessários é de 50%, de proteínas 30% e de gordura 20%. Além disso, foi considerada a abordagem semanal, com uma dieta para 7 dias que resulta em um total de calorias de 17.500 no qual o valor diário dela é de 2.500 calorias. Para fins deste trabalho, não foi avaliada como a alternância dos parâmetros de porcentagem de carboidratos, proteínas e gordura influenciam na resolução do problema. E sim, apenas o número de gerações. A Subseção 4.1 a seguir destina-se à explicar as métricas de avaliação utilizadas no trabalho e a Subseção 4.2 apresenta o desempenho do algoritmo desenvolvido comparando a versão mono objetivo e multiobjetivo.

4.1. Métricas

Com a finalidade de analisar a solução com maior assertividade, foram consideradas as métricas de tempo de execução do trecho de código do algoritmo genético, número de gerações e taxa de erro da solução em relação à meta. Todos os cálculos seguiram o modelo descrito anteriormente, em que cada simulação foi executada 35 vezes e tirada a média de todos os desempenhos. No caso do tempo de execução, considerou-se o tempo que as gerações daquela simulação leva para calcular a função *fitness*, seleção, recombinação, mutação e encontrar o melhor indivíduo de todas as gerações, que será a solução do problema. Adicionalmente, para descobrir a efetividade da solução proposta e quantas gerações em média são necessárias para encontrar a solução ótima, foram testados cenários com 50, 100 e 500 gerações. Não foi necessário calcular para mais gerações, pois foi percebido que nesse caso não há necessidade. Por fim, para analisar a taxa de erro, tanto para o cenário mono objetivo quanto para o multiobjetivo, foram testadas duas abordagens: RMSE, que calcula a raiz quadrática média, e MAE, que calcula o erro absoluto médio. A tabela 6 apresenta um teste realizado com a entrada de 50 gerações para a solução multiobjetivo. Como o objetivo é minimizar o erro, constatou-se que o método que apresenta melhor performance é o RMSE, pois como a tabela apresenta, é o que tem menor erro total.

Tabela 4. RMSE vs MAE

Métrica de erro	Erro (cal)
RMSE	3065
MAE	3512

4.2. Resultados

É possível observar na Tabela 5 que o tempo de execução do algoritmo genético é diretamente proporcional ao número de gerações do mesmo. Além disso, é visível que o desempenho para os casos mono e multiobjetivo são similares e ambos executam em tempo satisfatório.

Tabela 5. Tempo de execução

Número de gerações	Mono objetivo (s)	Multiobjetivo (s)
50	0,37	0,40
100	0,79	0,86
500	4,82	5,11

As Figuras 3, 4 e 5 representam, para os 3 cenários, o valor da função *fitness* com relação ao número de gerações em formato de gráfico *boxplot* e de linha (para a representação da média). Nos gráficos de 50 gerações (Figura 3) é possível perceber que conforme o número de gerações aumentam, de 5 em 5, há uma queda na Função Fitness. E quando chega próximo da trigésima geração, é visível que ocorre uma estabilidade nos valores das gerações seguintes. Além disso é possível perceber a distribuição de outliers ao longo das gerações, que se tornam cada vez mais próximos dos valores máximos e também a distância entre os terceiros e primeiros quartis diminui.

Figura 3. Gráficos de 50 gerações

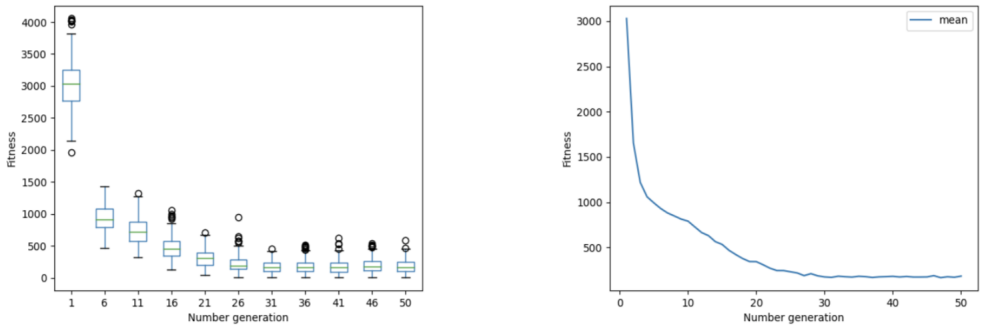


Figura 4. Gráficos de 100 gerações

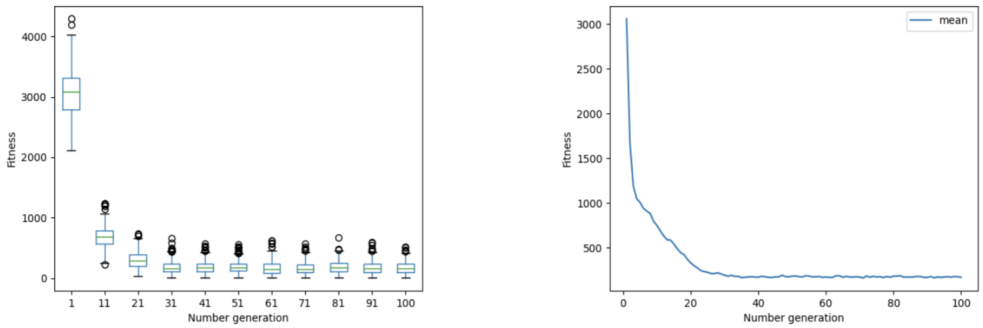
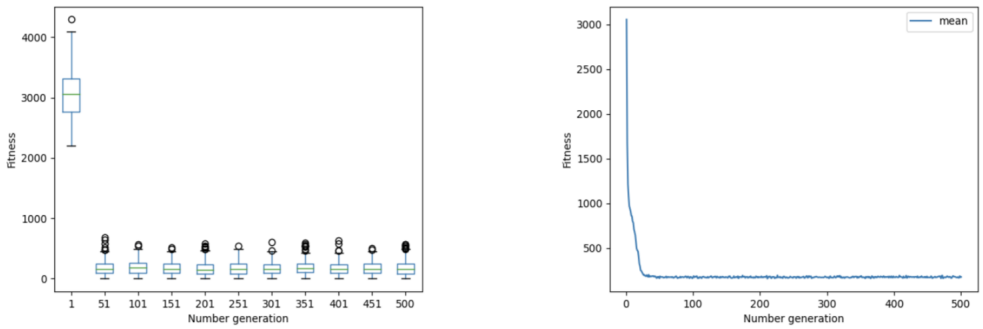


Figura 5. Gráficos de 500 gerações



O mesmo comportamento se repete para os gráficos de 100 e 500 gerações (Figuras 4 e 5). O gráfico de *boxplot* do primeiro caso representa os valores acrescidos de 10 em 10 e o segundo de 50 em 50. Em ambos, percebe-se a convergência da função

fitness ainda mais acentuada e é possível concluir, que tendendo ao infinito, esse comportamento se manterá. Portanto, para este projeto, não será necessário executar mais do que 30 gerações para encontrar a solução ótima.

Na Tabela 6 é possível verificar para os 3 tipos de gerações testadas, os valores de erro (em calorias) em comparação à meta para as categorias mono e multiobjetivo. Ao analisar para 50 gerações pode-se perceber que o erro em relação à meta para mono objetivo foi de 10.666 calorias, conforme vai crescendo para 100 e 500 gerações os valores aumentam para 11.120 e 11.387 calorias, respectivamente. Ao considerar o multiobjetivo podemos perceber que os valores de erro para 50, 100 e 500 gerações são bem próximos, sendo 3.065, 3.080 e 3.120 calorias, respectivamente. Com isso, podemos perceber que para uma meta de 17.500 calorias, o multiobjetivo teve em média 3.000 calorias de erro em relação ao objetivo, enquanto o mono objetivo aproximou-se do erro de 11.000 calorias.

Tabela 6. Erros

Número de gerações	Meta (cal)	Mono objetivo (cal)	Multiobjetivo(cal)
50	17.500	10.666	3.065
100		11.120	3.080
500		11.387	3.120

Adicionalmente, é possível perceber a consequência disso na porcentagem de cada macronutriente obtido a partir das soluções mono objetivo e multiobjetivo em comparação à meta, representada pela Figura 6. Foi apresentado neste exemplo apenas o resultado para 50 gerações, pois tanto para 100 quanto para 500 os gráficos são parecidos com o de 50, no qual divergem-se apenas em milésimos de porcentagem. Ao observar os gráficos em pizza, é possível perceber que as metas de carboidratos, proteínas e gorduras são 50%, 30% e 20%. Podemos concluir que o modo mono objetivo obteve porcentagens bem distintas da meta, no qual carboidratos, proteínas e gorduras diferem do objetivo em, aproximadamente, 20%, 10% e 30%. Em contrapartida, o multiobjetivo chegou mais próximo da meta, onde obteve para carboidratos, proteínas e gorduras, diferenças de, aproximadamente, 0,9%, 7,4% e 8,3%. O fato dessa solução não chegar ao valores exatos da meta e esse comportamento se manter constante ao longo das outras simulações com diferentes gerações pode ser justificável pela combinação dos ingredientes de entrada em conjunto com as restrições. Apesar disso, é possível comprovar que a solução que melhor se qualifica para resolver o problema da dieta é a multiobjetivo.

Figura 6. Gráfico de porcentagem das soluções para 50 gerações

5. Conclusão

Este trabalho abordou o problema de otimização da dieta para determinar quais os melhores alimentos a serem consumidos de acordo com informações fornecidas pelo usuário. Desta forma, empregou-se um algoritmo genético na resolução do problema, com abordagens mono e multiobjetivos, onde os resultados mostraram a eficiência do algoritmo que resolveu o problema em menos de 400ms. Assim, constatamos que os algoritmos genéticos são uma outra maneira de abordar um problema de computação linear e apresenta resultados promissores quando aplicado para multiobjetivos.

Referências

Barbosa, A. M., RIBEIRO, L. d. C., and ARANTES, J. d. O. (2010). Algoritmo genético multiobjetivo: Sistema adaptativo com elitismo. *Citado*, 2:16.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Foundation, P. S. (2021a). Python 3. `https://www.python.org/`. Acessado em 20/07/2021.

Foundation, P. S. (2021b). Time. `https://docs.python.org/3/library/time.html`. Acessado em 30/07/2021.

Fu, J.-L. and Wang, B.-Y. (2007). Development in nutritional epidemiological studies on types of dietary patterns and several chronic diseases. *Zhonghua liu xing bing xue za zhi= Zhonghua liuxingbingxue zazhi*, 28(3):297–300.

Harris, C. R. et al. (2020). Array programming with NumPy. *Nature*, 585:357–362.

Harrop, E. N. and Marlatt, G. A. (2010). The comorbidity of substance use disorders and eating disorders in women: Prevalence, etiology, and treatment. *Addictive behaviors*, 35(5):392–398.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Korstanje, J. (2020). Genetic algorithms in python using the deap library. `https://towardsdatascience.com/genetic-algorithms-in-python-using-the-deap-library-e67f7ce4024c`. Acessado em 18/07/2021.

Lv, Y. (2009). Multi-objective nutritional diet optimization based on quantum genetic algorithm. In *2009 Fifth International Conference on Natural Computation*, volume 4, pages 336–340. IEEE.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Mirjalili, S. (2019). *Genetic Algorithm*, pages 43–55. Springer International Publishing, Cham.

Steptoe, A. and Wardle, J. (2004). Health-related behaviour: Prevalence and links with disease.

USDA, U. S. D. o. A. (2021). How many calories are in one gram of fat, carbohydrate, or protein? `https://www.nal.usda.gov/fnic/how-many-calories-are-one-gram-fat-carbohydrate-or-protein`. Acessado em 18/07/2021.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85.