Solução do problema da entrega de presentes natalinos utilizando algoritmo genético

ANDREZA A SANTOS, DANIELA PALUMBO, GUILHERME FURLAN, LUCAS CUNHA E THAMIRIS COELHO *

*Ciência da Computação - Graduação (UNICAMP)
E-mail: a164213@dac.unicamp.br, d166301@dac.unicamp.br,
g160160@dac.unicamp.br, l172655@dac.unicamp.br,
t187506@dac.unicamp.br

***Resumo* – Este trabalho descreve uma solução para um problema proposto no site Kaggle: Santa Gift Matching Challenge [1]. Neste desafio, cada criança possui uma lista de presentes que deseja ganhar, enquanto o Papai Noel possui uma lista de crianças para as quais prefere entregar cada presente. A solução do problema consiste em achar a melhor combinação de crianças e seus presentes de forma a otimizar tanto a felicidade delas, quanto a do Papai Noel. Para tal, é proposta uma solução usando modelo evolutivo baseado em algoritmo genético, que utiliza como operadores as técnicas de *cross-over* e mutação. Diferentes combinações dos parâmetros de tamanho da população, quantidade de gerações, taxa de mutação e valor de corte para a próxima geração foram testados e para cada uma dessas combinações os valores máximos, médios e mínimos da função de adequação de cada geração foram computados e o tempo de execução médio do código foi medido. Os melhores resultados obtidos foram utilizando a configuração inicial mostrada na seção V-A, que alcançou um valor de adequação máximo de 0,36.**

***Palavras-chave* – Santa Gift Matching Challenge, Algoritmo genético**

I. INTRODUÇÃO

Este trabalho foi realizado utilizando os conceitos de modelos evolutivos vistos na disciplina de Inteligência Artificial, junto com os materiais didáticos propostos, como slides [2], [3], e o livro base da disciplina [4]. O problema foi selecionado buscando uma série de problemas presentes na plataforma Kaggle [5], com foco naqueles nos quais seriam possível utilizar uma abordagem de otimização, se encaixando nos modelos de algoritmos genético vistos em aula.

II. SEÇÕES

Esse trabalho está dividido nas seguintes seções:

III. TRABALHO PROPOSTO

No natal, as crianças possuem uma lista de presentes que gostariam de ganhar, enquanto o Papai Noel possui para cada presente, uma lista ordenada com as crianças para as quais ele gostaria de entregá-los. O objetivo do algoritmo genético é maximizar tanto a felicidade do papai noel quanto a felicidade das crianças. Essa relação de felicidade é computada pela função de adequação, definida na seção IV.

Também existem algumas regras específicas para a solução do problema:

1) Entre as crianças existem gêmeos e trigêmeos, correspondendo a 1.5% e 8% do total de crianças respectivamente. Crianças que se enquadram nesses casos devem ganhar os mesmos presentes por solicitação dos seus pais.
2) O número total de presentes é sempre igual a quantidade total de crianças, de modo que nenhuma criança ficará sem presente.

Base de Dados

A base de dados para o problema é composta por dois arquivos, ambos com $n_{children}$ linhas, onde $n_{children}$ representa a quantidade total de crianças. São eles:

1) whishlist-$n_{children}$.csv: cada linha deste arquivo representa uma criança $child_{ID}$ com seus oito respectivos $gifts_{ID}$ ordenados por preferência.
2) gifts-$n_{children}$.csv: cada linha deste arquivo representa uma criança $child_{ID}$ com os respectivos $gifts_{ID}$ que o Papai Noel deseja que ela ganhe ordenados por preferência.

IV. MATERIAIS E MÉTODOS

Para solucionar o problema, utilizamos algoritmo genético. Nesta modelagem, cada indivíduo foi representado por um array Numpy [6] l de tamanho $n_{children}$. Devido a presença de trigêmeos e gêmeos, e da regra específica (1) na seção III, os primeiros 1.5% índices representam os trigêmeos, os 8% índices seguintes representam os gêmeos e o restante são crianças que não se encaixam nessas categorias. Para cada índice desse array, $l[i]$ representa o presente que a criança de índice i receberá do Papai Noel.

A. Função de Adequação

A função de adequação utilizada para maximizar a felicidade do Papai Noel e das crianças está definida na (Equação 1) a seguir.

$$\mathcal{F} = \frac{\sum_{g \in Solution} \frac{CH(g)+SH(g)}{2}}{n_{children}} \quad (1)$$

De modo que CH, definida na (Equação 2), representa a felicidade da criança por receber aquele presente e SH, definida na (Equação 3), representa a felicidade do Papai Noel em dar aquele presente para aquela criança.

$$CH(g) = \begin{cases} \frac{n_{wishlist} - Wishlist(g)}{n_{wishlist}} & \text{se} \quad g \exists Wishlist \\ -1 & \text{se} \quad g \nexists Wishlist \end{cases} \quad (2)$$

$$SH(g) = \frac{n_{giftTypes} - SantaRank(g)}{n_{giftTypes}} \quad (3)$$

As variáveis presentes nessas equações devem ser interpretadas como:

- g : presente sendo pontuado
- $n_{whishlist}$: tamanho da lista de presente das crianças
- $n_{giftTypes}$: quantidade de tipos de presentes
- $Whishlist$: lista de desejos de uma criança
- $SantaRank$: lista de preferências do Papai Noel

A função de adequação proposta pela competição do Kaggle [1] não foi utilizada, pois aos dados gerados para esse trabalho não seguem o padrão da competição.

B. Implementação

A solução foi implementada em Python 3 [7], sendo dividida da seguinte maneira:

- *main.py:* arquivo que resolve o problema usando Algoritmo Genético.
- *generate-dataset.py:* arquivo que gera aleatoriamente os datasets utilizado na solução.
- *defines.py:* arquivo no qual são definidos os parâmetros relacionados à execução do algoritmo genético, como taxa de mutação, tamanho de população, quantidade máxima de gerações e demais valores.
- *crossover.py:* arquivo que possui as implementações das técnicas de *croo-over*.
- *mutation.py:* arquivo que possui as implementações das técnicas de mutação.
- *fitness_function.py:* arquivo que possui as implementações da função de adequação juntamente com os métodos de seleção.

Durante a modelagem do problema, algumas definições foram feitas:

- O número de tipos de presentes é igual a quantidade de cada tipo de presente
- O número de crianças é igual ao quadrado do número de presentes, satisfazendo a regra específica (2) na seção III
- As taxas de *cross-over* utilizadas nesse trabalho foram de 50% e de 99%.

Para a geração zero, o array de indivíduos com o tamanho da população definido foi preenchido com presentes de forma aleatória e seguindo as regras específicas da seção III. Para as gerações seguintes, é computado os valores de adequação de cada indivíduo através da função de adequação (Equação 1). A seleção dos indivíduos que seguirão para a próxima geração é feita através da seleção por torneios ou da seleção pelo método da roleta. No primeiro caso, os valores de adequação dos indivíduos são comparados entre si e os indivíduos com maiores valores são selecionados. Já no segundo, é feito o cálculo da probabilidade de um indivíduo ser selecionado através da divisão do seu valor de adequação pelo somatório dos valores de todos os indivíduos da geração e a seleção é feita de forma aleatória utilizando essas probabilidades. É importante notar que para o segundo caso, é possível que o mesmo indivíduo seja selecionado mais de uma vez e que indivíduos com valores altos de adequação não sejam selecionados, situações que não são possíveis na seleção por torneio.

A composição da próxima geração é feita usando o método de substituição chamado *Steady-State* ou o método chamado elitismo. No primeiro, os indivíduos com melhores valores de adequação farão parte da próxima geração e os demais serão substituídos por novos indivíduos gerados a partir daqueles que continuaram, utilizando as técnicas de *cross-over* e de mutação. Já no segundo, somente o indivíduo com valor de adequação mais alto passará para a próxima geração, sendo os demais indivíduos substituídos por novos, criados através das técnicas de *cross-over* e de mutação. Os critérios de parada utilizados foram o número de gerações ou a convergência da função de adequação.

C. Técnicas de cross-over

Duas técnicas de *cross-over* foram utilizadas nesse trabalho: *cross-over* uniforme e *cross-over* de ponto único.

O *cross-over* uniforme funciona copiando as informações dos dois cromossomos pais de forma alternada para o cromossomo filho, porém caso não seja possível transmitir o cromossomo de nenhum dos dois pais devido ao limite de quantidade que cada presente possui, um presente válido é escolhido de forma aleatória. Já o *cross-over* de ponto único funciona copiando todos os elementos do primeiro cromossomo para o cromossomo filho até aproximadamente a metade e depois completa o restante das posições do cromossomo filho com as informações do segundo cromossomo. Caso não seja possível copiar um elemento do segundo cromossomo devido à quantidade máxima de presentes daquele tipo, outro presente válido é selecionado aleatoriamente.

D. Técnicas de Mutação

A técnica de mutação utilizada foi uma mesclagem das técnicas de *swap*-simples, onde é selecionado pares de genes de forma aleatória do cromossomo para realizar a troca, e da técnica de *swap*-sequencial, onde há a troca de uma

sequência de itens de uma vez, por questão da regra específica (2) III. Sendo assim, caso algum dos genes selecionados para a troca seja um trigêmeo ou gêmeo, deve-se selecionar 3 ou 2, respectivamente, exemplares do outro gene de modo que a troca, utilizando o *swap*-sequencial, respeite a quantidade de cada tipo de presente dentro do indivíduo. Para as crianças que não se encaixam nessas categorias, podemos realizar a troca com qualquer outra criança entre elas utilizando o *swap*-simples. O número de trocas realizadas é controlado pela taxa de mutação definida.

V. Resultados e Discussão

A. Experimentos Iniciais

Foram realizados vinte experimentos variando os seguintes parâmetros: tipos de presentes, tamanho da população, taxa de mutação e valor de corte. Os valores utilizados em cada experimento estão apresentados na Tabela I. Além dos parâmetros utilizados a tabela também mostra o tempo médio de execução por geração em cada experimento. A configuração do algoritmo genético utilizada está especificada abaixo.

- **configuração do algoritmo genético:**
  - critério de parada: geração número 2000
  - método de seleção: torneios
  - método de substituição: *Steady-State*
  - técnica de *cross-over*: *cross-over* uniforme
  - técnica de mutação: *swap* e *swap*-sequencial
  - taxa de *cross-over*: 50%

#	tipos de presentes	tamanho da população	taxa de mutação	valor de corte	tempo médio por geração(s)
1	25	50	5%	20%	1,17
2	25	50	5%	50%	0,44
3	25	50	20%	20%	1,78
4	25	50	20%	50%	1,46
5	25	100	5%	20%	1,99
6	25	100	20%	20%	2,47
7	25	200	5%	20%	2,23
8	25	200	5%	50%	2,01
9	25	200	20%	20%	2,77
10	25	200	20%	50%	2,28
11	50	50	5%	20%	2,19
12	50	50	5%	50%	1,19
13	50	50	20%	20%	4,04
14	50	50	20%	50%	2,35
15	50	100	5%	20%	2,99
16	50	100	20%	20%	6,47
17	50	200	5%	20%	7,98
18	50	200	5%	50%	6,73
19	50	200	20%	20%	17,22
20	50	200	20%	50%	13,10

Tabela I: Conjuntos de parâmetros dos experimentos. Para todas as execuções o tamanho da lista de desejos das crianças foi igual a 8.

Como as configurações 7 e 8 diferem somente na porcentagem de indivíduos que são levados de uma geração para a próxima, podemos ver ao comparar a Figura 1 com a Figura 2 como isso afeta a evolução da solução, em especial como passar mais indivíduos para a próxima geração aumenta a adequação media, já que menos indivíduos da população final sofreram *cross-over* e mutação.

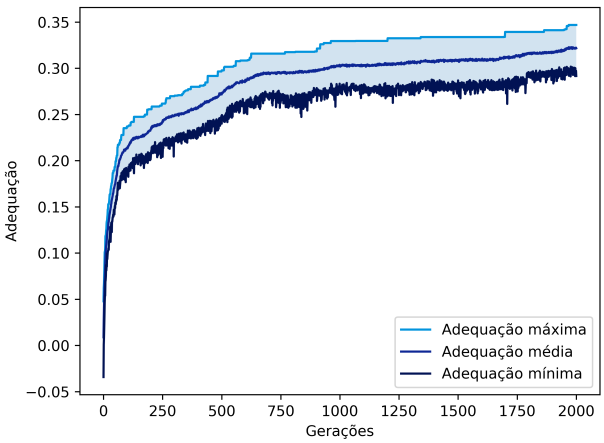

Figura 1: Gráfico de adequação das gerações resultantes da configuração 7

Também é possível ver os efeitos dessa variável na adequação máxima atingida, a configuração com 50% dos indivíduos continuando na população levou em torno de 1000 gerações para alcançar a marca de 0,30 de adequação, enquanto com 20% essa marca foi atingida na geração 500. Entretanto a constante evolução da primeira configuração fez com que ela terminasse as 2000 gerações com uma adequação maior.

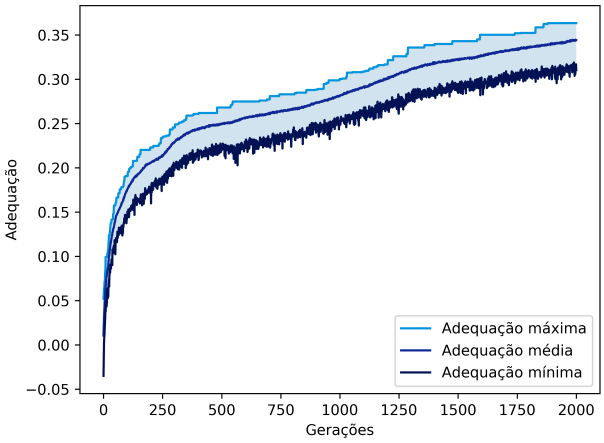

Figura 2: Gráfico de adequação das gerações resultantes da configuração 8

As configurações 8 e 10 são similares, somente se diferenciando na taxa de mutação. Isso permite analisarmos o efeito dessa variável na solução final; é possível ver na Figura 3 como uma taxa de mutação grande faz com que os novos indivíduos da população sejam aleatórios demais, eliminando

possíveis genes bons que herdaram dos pais, fazendo com que o algoritmo convirja muito rápido. Usando um valor mais baixo podemos ver que, na Figura 2, a taxa de mutação gera uma curva de constante evolução, chegando a valores de adequação maiores. Também é possível perceber que a curva em 2 ainda estava ascendendo na geração 2000, significando que mais gerações possivelmente melhorariam os resultados obtidos.

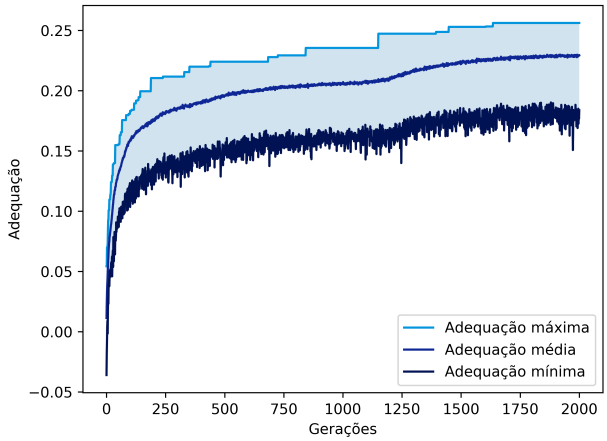

Figura 3: Gráfico de adequação das gerações resultantes da configuração 10

Comparando as Figuras 4 e 1, cujas configurações divergem somente no tamanho da população, percebe-se que uma quantidade menor de indivíduos torna a evolução mais difícil, dependendo de *breakthrough*s para sair de longos platôs na adequação. A população maior consegue evoluir constantemente, graças à maior variabilidade que uma população maior trás.

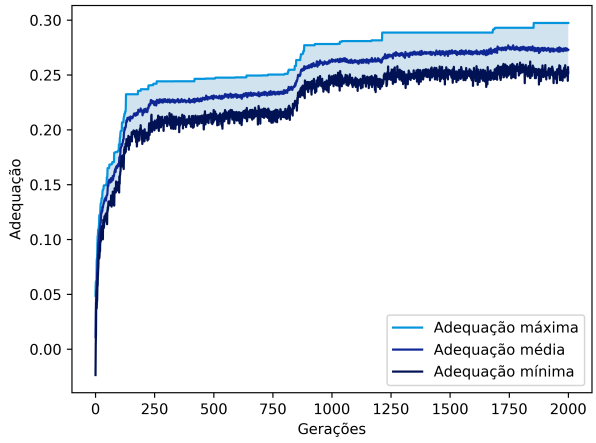

Figura 4: Gráfico de adequação das gerações resultantes da configuração 1

Temos na Figura 5 um exemplo de configuração com 50 tipos de presentes, o que significa que temos 2500 crianças, um aumento considerável das 625 crianças quando se tem 25 tipos de presentes. Por conta desse aumento, o algoritmo leva mais tempo por geração e leva muito mais gerações para conseguir aumentar sua adequação, nunca chegando perto da adequação do caso de 25 presentes, mas ainda mostrando uma evolução constante e bem definida no gráfico.

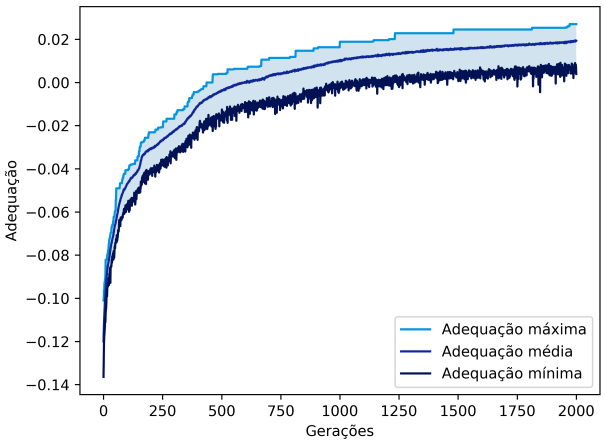

Figura 5: Gráfico de adequação das gerações resultantes da configuração 18

É possível perceber na Figura 6 que o valor da adequação é baixo no fim da última geração. Esse resultado pode ser relacionado ao fato de que a taxa de mutação aplicada é muito alta para 2500 crianças, tendo o mesmo problema da perda de genes apontado na Figura 3 combinado ao comportamento da quantidade de gerações da Figura 5.

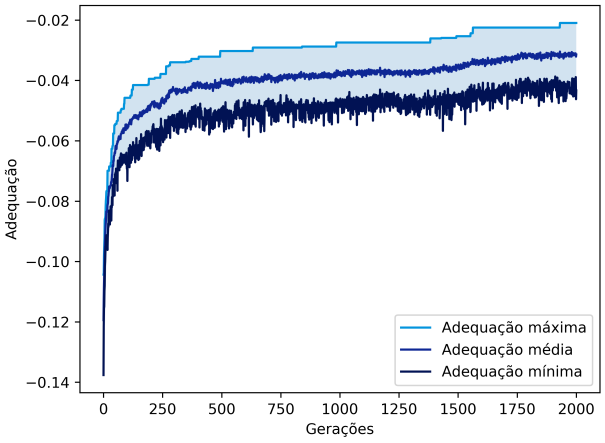

Figura 6: Gráfico de adequação das gerações resultantes da configuração 13

Outra coisa que pode ter afetado o resultado é o número de gerações. Por ser uma grande quantidade de crianças, é

possível que um maior número de gerações fosse necessário para atingir uma adequação mais alta.

Na Tabela II é possível ver o resultado ao fim da execução dos experimentos, o experimento que apontou o melhor resultado foi o 8º atingindo 0,36 de adequação. Já o experimento com o pior resultado foi o do 13º resultando em uma adequação negativa. É interessante observar que esses dois experimentos possuem todos os parâmetros diferentes.

Pela tabela também fica claro que os experimentos com um número de crianças maior atingiram resultados piores, provavelmente porque são necessárias mais gerações para convergir.

Experimento	Mínimo	Média	Máximo
1	0,25	0,27	0,29
2	0,23	0,25	0,27
3	0,15	0,18	0,22
4	0,16	0,20	0,22
5	0,28	0,31	0,33
6	0,17	0,21	0,24
7	0,29	0,32	0,35
8	0,31	0,34	**0,36**
9	0,18	0,23	0,26
10	0,18	0,23	0,26
11	-0,01	0,00	0,01
12	-0,01	0,00	0,01
13	-0,04	-0,03	-0,02
14	-0,03	-0,02	-0,01
15	0,00	0,01	0,02
16	-0,01	0,00	0,01
17	0,01	0,02	0,03
18	0,00	0,02	0,03
19	0,01	0,02	0,03
20	0,01	0,02	0,03

Tabela II: Resultados ao fim da execução das 2000 gerações para cada uma das 20 configurações realizadas.

Por último pode-se analisar o tempo médio de execução de uma geração em cada configuração proposta através da Tabela I. Não é possível fazer comparações entre quaisquer das configurações, pois foram executadas em computadores distintos. Entretanto, se tomarmos o tempo das últimas quatro configurações, que foram executadas no mesmo computador, podemos ver a influência de cada variável no tempo de execução.

Por exemplo, a configuração 18 foi mais rápida que a 17, já que com mais indivíduos passando de uma geração para a próxima, menos indivíduos novos devem ser gerados, diminuindo o número de *cross-over* e mutações necessárias. O mesmo tipo de análise pode ser aplicada as configurações 17 e 19, que mostra como uma quantidade maior de mutações por indivíduo também acarreta em maior tempo para processar essas modificações em cada um.

B. Experimentos Exploratórios do Melhor Resultado

A partir do melhor resultado obtido utilizando a configuração mostrada em V-A, foi realizado experimentos trocando algumas configurações como o critério de parada, a técnica de *cross-over*, método de seleção e o método de substituição, de modo a avaliar o efeito dessas mudanças no nosso algoritmo genético.

Primeiro foi analisado os efeitos de um critério de parada baseado na adequação máxima alcançada. Considerando o valor 0.36, que foi o maior valor obtido pela configuração 8, este critério de parada fez com que o experimento parasse na geração 1863, ao invés da geração 2000, que era o critério de parada antigo. Com a média de dois segundo por geração dessa configuração, isto significa que economizamos em torno de quatro minutos e meio de tempo de execução para obter o mesmo resultado final.

Em seguida modificamos a melhor configuração para utilizar o *cross-over* de ponto único utilizando elitismo como método de substituição, no lugar do *cross-over* uniforme com *Steady-State* que utilizamos inicialmente. Pode-ser ver na Figura 7 que, para o problema proposto, este tipo de *cross-over* não é adequado, pois a população encontra um platô rapidamente e, apesar do máximo subir gradativamente, a média se mantém quase constante, o que indica que o modelo não irá evoluir além dos valores de adequação já obtidos.

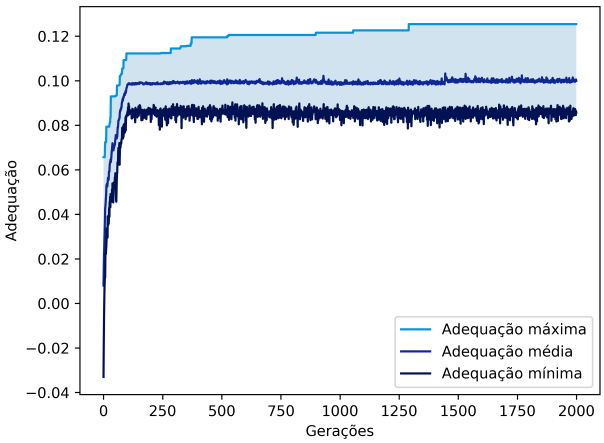

Figura 7: Gráfico de adequação das gerações resultantes da configuração 8 com *cross-over* de ponto único com elitismo

Por último a configuração 8 foi modificada para utilizar o método da roleta como método de seleção do algoritmo genético, ao invés do método de torneio utilizado anteriormente. Como este novo método aplicado, obtemos os dados da Figura 8, na qual podemos ver que a aleatoriedade que o método da roleta trás para o algoritmo não foi benéfico para o nosso problema, alcançando adequações maiores que com o *cross-over* de ponto único, mas caindo de volta, flutuando ao redor de uma média máxima de 0.13.

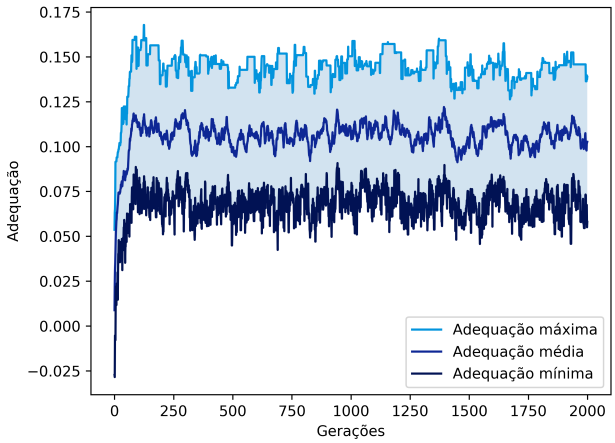

Figura 8: Gráfico de adequação das gerações resultantes da configuração 8 com seleção pelo método da roleta

VI. Divisão de Tarefas

Para a realização desse trabalho o grupo se reuniu via Google Meet e além de tomar todas as decisões juntos todo o código foi escrito e definido em conjunto. Portando as tarefas não foram divididas entre os membros, mas feitas em conjunto.

Além disso os experimentos foram divididos entre os integrantes a fim de executá-los em paralelo para otimizar o tempo de reprodução.

VII. Conclusões

Uma solução baseada em algoritmos genéticos para uma variação do desafio disponibilizado na plataforma Kaggle [5] foi modelada e testada. É possível concluir que, para o algoritmo implementado, a seleção de parâmetros faz muita diferença, de maneira que uma taxa de mutação muito alta não é o ideal para esse cenário; reaproveitar um percentual maior do pai para o filho torna a execução mais rápida e os resultados mais constantes enquanto que uma população maior torna a execução mais lenta, porém com maior variabilidade e assim menos platôs nos resultados. Foi possível observar que o conjunto de parâmetros para 25 tipos de presentes, população de tamanho 200, com 5% de taxa de mutação e 50% de valor de corte trouxe um melhor resultado na taxa de adequação. Os experimentos exploratórios mostrados na seção V-B não mostraram resultados melhores que a configuração inicial do algoritmo mostrada na seção V-A para a maioria dos casos, exceto para o critério de parada, que economizou cerca de 4 minutos para alcançar o mesmo valor de adequação alcançado na configuração inicial. O método da roleta trouxe muita aleatoriedade para os indivíduos, que não foi positivo para o algoritmo, e o *cross-over* de ponto único com elitismo convergiu para um valor de adequação do qual não conseguiu evoluir muito mais.

Referências

[1] Kaggle, "Santa Gift Matching Challenge," https://www.kaggle.com/c/santa-gift-matching, 2018, [Online; accessed 13-June-2020]. 1, 2

[2] E. L. Colombini and A. Simões, "Lecture 7 – Evolutionary Computing," https://classroom.google.com/u/2/w/NTcwNTk3OTc3MDRa/tc/MTI1NjQ5MzA0NDUw, 2020, [Online; accessed 13-June-2020]. 1

[3] ——, "Lecture 8 – Genetic Programming," https://classroom.google.com/u/2/w/NTcwNTk3OTc3MDRa/tc/MTI1NjQ5MzA0NDUw, 2020, [Online; accessed 13-June-2020]. 1

[4] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002. 1

[5] "Kaggle: Your Machine Learning and Data Science Community," https://www.kaggle.com/, [Online; accessed 13-June-2020]. 1, 6

[6] T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing, 2006–, [Online; accessed ¡today¿]. [Online]. Available: http://www.numpy.org/ 1

[7] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. 2