Algoritmo genético aplicado em resolução do problema de combinação de crianças e presentes

https://github.com/isabelatelles/mc906_collab

ISABELA TELLES FURTADO DOSWALDO
RA 170012
E-mail: i170012@dac.unicamp.br

NATHÁLIA HARUMI KUROMIYA
RA 175188
E-mail: n175188@dac.unicamp.br

***Resumo* – O trabalho descreve o problema de combinação de crianças e presentes, em que cada criança tem uma lista de presentes que prefere ganhar, e o Papai Noel tem uma lista de crianças as quais tem preferência para ganhar determinado presente. Além disso, há limitações, como o número de presentes e as crianças trigêmeas e gêmeas, que devem ganhar os mesmos presentes. Por ser um problema de otimização, dado que a intenção é otimizar a felicidade das crianças e do Papai Noel, o algoritmo genético é a abordagem utilizada na resolução. Após diversas tentativas de configurações do algoritmo, foi possível perceber que a taxa de mutação deveria ser alta, devido às restrições do problema, e o tamanho da população deveria ser médio, devido ao tempo de execução. Dessa forma, o algoritmo convergiu para uma solução ótima.**

***Palavras-chave* – Algoritmo Genético, Otimização, Combinação**

I. INTRODUÇÃO

Para a confecção desse trabalho, foi estudado métodos evolutivos para resolução de problemas. Nesse caminho, utilizou-se o livro [1], além dos slides [2] e [3] para encontrar o melhor modelo evolutivo, além de abordar todas as restrições e peculiaridades da implementação do algoritmo genético.

A motivação para trabalhar com o tema de métodos evolutivos permeia a ideia de entender melhor os conceitos e funcionamento de algoritmos genéticos para otimização de problemas gerais. Já a escolha do problema veio da dificuldade que as restrições trazem: dessa forma, era necessário adaptar os conceitos de métodos evolutivos para encaixar no problema, o que exige um conhecimento e entendimento mais profundo do tema.

II. SEÇÕES

Esse trabalho está separado nas seguintes seções:

III. TRABALHO PROPOSTO[4]

A. *O Problema*

O trabalho em questão se propõe em usar algoritmo genético para resolver um problema baseado em um *challenge* do *Kaggle*[5]:

Em um cenário de natal, cada criança possui uma lista de quais presentes gostaria de ganhar em ordem de preferência. Ao mesmo tempo, o Papai Noel carrega uma lista para cada tipo de presente com crianças para quem ele prefere dar o presente em questão. A ideia é maximizar a felicidade de das crianças e do Papai Noel a partir da combinação de crianças e presentes de natal que cada uma irá ganhar. Como restrições do problema temos:

1) Os presentes são limitados. Há exatamente 1 presente por criança e um número igual de exemplares de cada tipo de presente.
2) Na lista de crianças, há 1.5% de trigêmeos e 4% de gêmeos. Como uma requisição dos pais, essas crianças devem ganhar os mesmos presentes. Ou seja, se a criança 1, 2 e 3 são trigêmeos entre si, então essas crianças devem ganhar o mesmo tipo de presente.

B. *Dataset*

O *dataset* é composto pelas seguintes tabelas:

1) *Child Whishlist*: cada índice da linha representa uma criança, sendo que os primeiros 1,5% dos índices representam trigêmeos e os seguintes 4% dos índices representam gêmeos, e cada coluna possui um presente desejado pela criança em questão.
2) *Gift Good Kids*: cada índice da linha representa um tipo de presente e cada coluna possui uma criança para a qual o Papai Noel deseja dar o presente em questão.

IV. Materiais e Métodos

A. Modelagem

Para a resolução do problema, o **modelo evolutivo** adotado se baseia em algoritmo genético. A **representação do indivíduo** é dada por uma lista *l[]*, em que cada índice *i* representa uma criança e a informação contida na lista *l[i]* é o tipo de presente que a criança irá receber. Os primeiros 1.5% dos índices da lista representam os trigêmeos do problema e os 4% dos índices seguintes representam os gêmeos. A **população inicial** é escolhida aleatoriamente, com a restrição de que gêmeos e trigêmeos devem ganhar o mesmo tipo de presente. A **técnica de crossover** é baseada em uma combinação de ponto único e a **taxa de crossover** escolhida foi de 100%. O algoritmo se baseia em copiar a primeira metade do primeiro cromossomo e copiar o máximo possível da segunda metade do segundo cromossomo. Os campos não possíveis de serem copiados, devido a limitação da quantidade de presentes, são completados com os presentes que restaram. A **função de seleção** é baseada em seleção por torneio com dimensão de torneio variando aleatoriamente até, no máximo, a metade do tamanho da população, para que o algoritmo não conduza à convergência prematura. O **método de substituição** é o elitismo, em que o mais apto indivíduo da geração anterior é acrescentado na nova geração, e o resto da população é substituída pelos filhos gerados pelos pais selecionados. Dessa forma, a melhor solução pode sobreviver à sucessivas gerações e, ainda assim, quase toda a população é substituída. A **mutação** é implementada baseada na técnica de *swap*, de forma que se uma criança trigêmea ou gêmea é escolhida aleatoriamente, seus irmãos também trocam os presentes com outras crianças que tenham um mesmo tipo de presente. A **taxa de mutação** escolhida foi de 90%. A **condição de parada** se baseia em um número máximo de 2000 gerações. A **função de aptidão** a ser otimizada é dada por:

$$f = (ANCH)^3 + (ANSH)^3$$

onde

$$ANCH = \frac{1}{n_c} \sum_{i=1}^{n_c-1} \frac{ChildHappiness}{MaxChildHappiness}$$

$$ANSH = \frac{1}{n_g} \sum_{i=1}^{n_g-1} \frac{GiftHappiness}{MaxGiftHappiness}$$

As variáveis n_c, n_g, $ChildHappiness$, $GiftHappiness$, $MaxChildHappiness$ e $MaxGiftHappiness$ são variáveis dependentes do número de crianças e do número de presentes:

- nc é o número de crianças.
- ng é o número de presentes
- $MaxChildHappiness = len(ChildWishList) * 2$
- $MaxGiftHappiness = len(GiftGoodKidsList) * 2$
- $ChildHappiness = 2 * GiftOrder$ se o presente for encontrado na lista de presentes desejados da criança
- $ChildHappiness = -1$ se o presente não estiver na lista.
- $GiftHappiness = 2 * ChildOrder$ se a criança é encontrada na lista do Papai Noel referente àquele presente.
- $GiftHappiness = -1$ se a criança não estiver na lista.

Para esse trabalho, foram verificadas diferentes configurações em relação ao número de crianças e número de presentes, que serão tratadas na seção V. Resultados e Discussão. O **tamanho da população**, o **método de substituição,** a **taxa de mutação** e a **taxa de crossover** também foram variadas e serão tratadas na mesma seção. A **ténica de crossover** e a **técnica de mutação** não foram variadas devido às restrições do problema explicitadas na seção III. Trabalho Proposto.

B. Implementação

A implementação foi feita em *Python 3*[6]. Os *datasets* utilizados no problema foram manipulados a partir do Pandas[7] e gerados com dados aleatórios. A classe *SantaProblem* contém o número de crianças e o número de tipos de presentes que será considerado na modelagem, bem como a implementação de algumas funções como a de *crossover* e de mutação. Para visualizar a evolução da solução a cada geração foi utilizada a biblioteca Matplotlib[8]. Por fim, foi implementada uma classe separada para a função de aptidão devido sua complexidade.

V. Resultados e Discussão

Para avaliação, foram testadas várias configurações diferentes. A primeira ideia foi variar o número de crianças e o número de tipos de presentes. Para isso, os valores (n_c, n_g) testados foram (50, 10), (200, 40) e (500, 100). A variação desses números foi importante para determinar qual a extensão do *dataset* que o problema permitia. A configuração (500, 100) demorou cerca de 12 horas para terminar a execução com 70 indivíduos de tamanho da população e 65% de taxa de mutação, gerando a Figura 1.

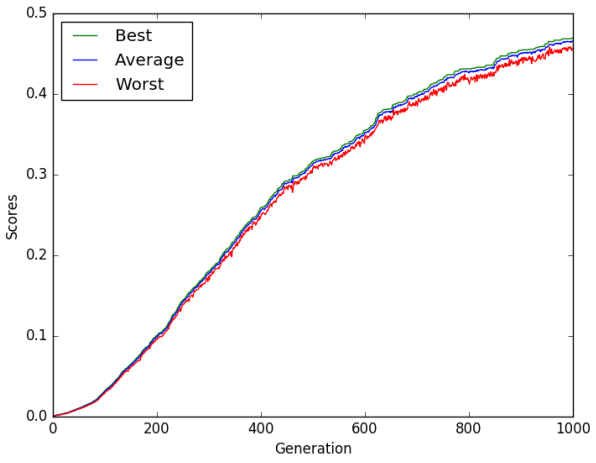

Figura 1. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 500 crianças, 100 tipos de presentes, 70 indivíduos por população, máxima geração de 1000 e taxa de mutação de 65%.

Apesar do algoritmo estar convergindo, não é possível ter certeza que sua solução final é a melhor solução que o

algoritmo é capaz de encontrar. Portanto, dado o limitante temporal, ficou inviável utilizar essa configuração para avaliações mais profundas.

Já a configuração (200, 40) foi testada com população de 70 indivíduos e 65% de taxa de mutação (gerando a Figura 2) e com população de 20 indivíduos e 65% de mutação (gerando a Figura 3).

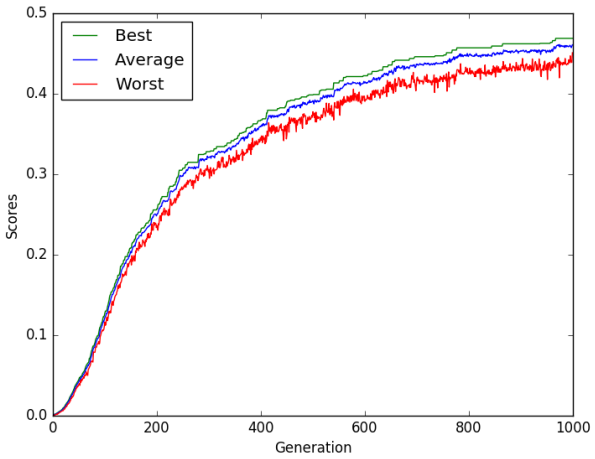

Figura 2. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, 70 indivíduos por população, máxima geração de 1000 e taxa de mutação de 65%.

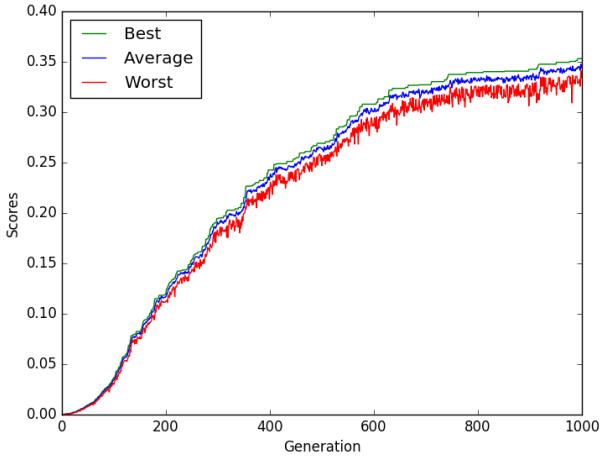

Figura 3. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, 20 indivíduos por população, máxima geração de 1000, taxa de mutação de 65%.

As duas configurações também sofrem do mesmo problema da configuração (500, 100): apesar dos resultados estarem convergindo, não conseguimos obter uma estabilidade clara nas últimas gerações.

Por outro lado, é possível perceber o efeito da variação do tamanho da população. A Figura 2 possui uma população maior do que a Figura 3, e, da mesma forma, é possível perceber que a primeira converge mais rápido do que a segunda, além da primeira também obter soluções melhores do que a segunda.

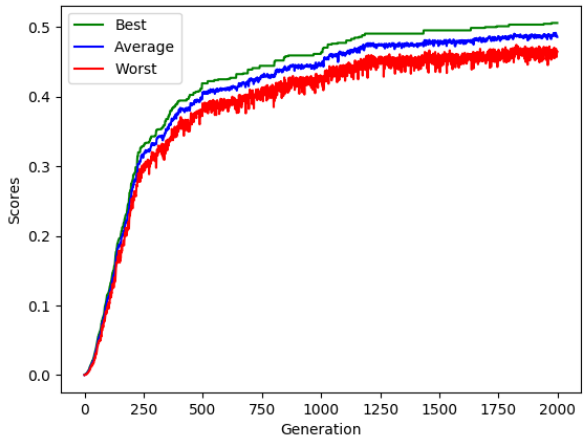

Figura 4. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, e a configuração mencionada na modelagem em IV. MATERIAIS E MÉTODOS.

Após muitas variações no algoritmo genético e seus parâmetros, foi possível obter a configuração mencionada na modelagem em IV. MATERIAIS E MÉTODOS, e representada na Figura 4 para 200 crianças e 40 tipos de presentes. Trataremos então, das variações que nos permitiram alcançar a modelagem apresentada.

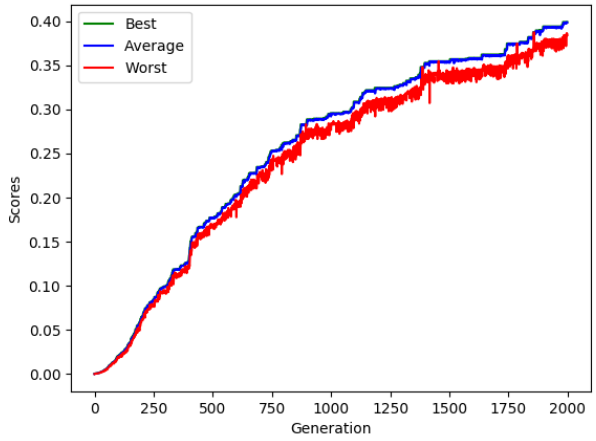

Figura 5. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, e a taxa de mutação em 10%.

O efeito da taxa de mutação é explicitado pela Figura 5, que mostra um gráfico quase linear da evolução da aptidão dos indivíduos. A melhor aptidão e a média das aptidões seguem as mesmas durante as 2000 gerações, o que indica que as gerações provavelmente possuíam pouca diversidade de indivíduos. A solução ótima não foi alcançada, ao contrário do

que é visto na Figura 4, em que a aptidão do melhor indivíduo alcança o valor de 0.5. Isso porque a alta taxa de mutação faz com que o algoritmo ache a solução ótima através de soluções geradas aleatoriamente.

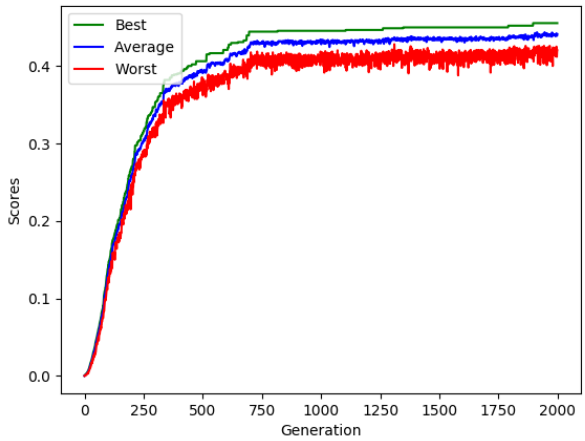

Figura 6. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, e a taxa de crossover em 50%.

A variação da taxa de crossover para 50% nos trouxe os resultados apresentados na Figura 6. Apesar de convergir a uma solução previamente, aproximadamente na geração 750, não conseguiu convergir à solução ótima. A aptidão do melhor indivíduo alcança o valor de 0.45, o que não corresponde ao valor da solução ótima.

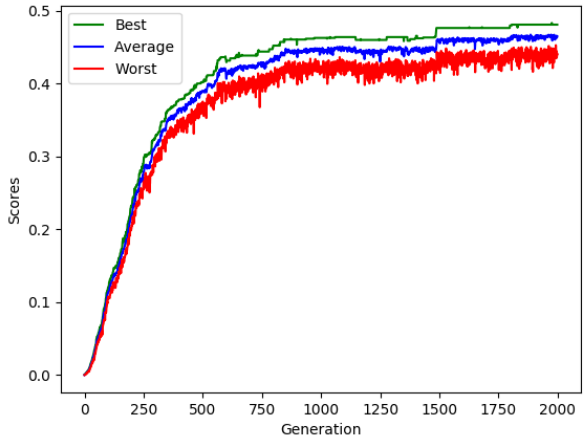

Figura 7. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, e o método de substiuição da troca de toda a população.

A Figura 7 mostra o uso do método de substituição da troca de toda a população, de forma que os descendentes substituem toda a população. Esse resultado foi o que mais se aproximou daquele apresentado na Figura 4, pois, de fato, o método utilizado diferencia pouco do método de elitismo. Mas, apenas o fato de adicionar em todas as gerações o indivíduo de melhor aptidão, já faz a diferença necessária para encontrar a solução ótima.

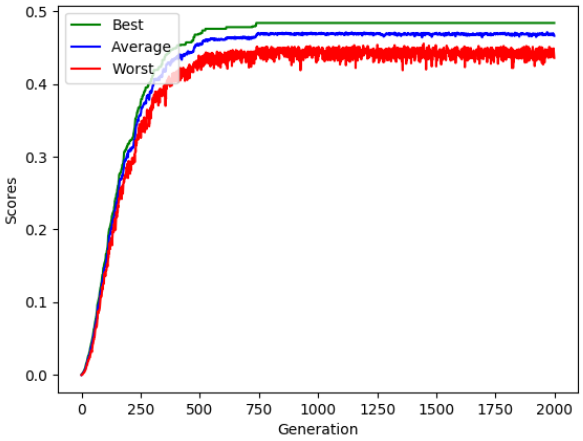

Figura 8. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 200 crianças, 40 tipos de presentes, e população de 140 indivíduos.

Por fim, foi testado uma população de 140 indivíduos, como pode ser visto na Figura 8. É importante ressaltar que esse aumento na população afetou de forma significante no desempenho do algoritmo. Com 70 indivíduos, o algoritmo rodou em um pouco menos de uma hora, em uma CPU, enquanto com o dobro, 140 indivíduos, trouxe uma espera de mais de 4 horas, em uma GPU.

Com o aumento do tamanho da população, é possível observar uma convergência muito mais rápida em relação aos resultados das outras variações de parâmetros, aproximadamente na geração 450. Uma população maior faz com que tenha mais chances de encontrar um indivíduo-solução em uma geração anterior. Entretanto, novamente, essa variação não foi o bastante para que se alcançasse a solução ótima, apesar de alcançar um valor próximo, de 0.48.

A terceira configuração testada foi a (50, 10). É uma configuração razoavelmente limitada, que com 70 indivíduos na população e 65% de taxa de mutação atinge a solução ótima, apesar dessa ter um valor de aptidão em torno de 0.076, um valor muito abaixo daqueles encontrados na configuração (200, 40). Os resultados são apresentados na Figura 9.

VI. Conclusões

O trabalho contou com algumas dificuldades durante a sua confecção. Primeiramente, a escolha do problema trouxe a dificuldade de gerar um *dataset*. O problema original[5] considera 1 milhão de crianças e mil tipos de presentes. Essas dimensões não eram compatíveis com a robustez do algoritmo genético e, por isso, foi necessário a criação de novos *datasets* que cabiam dentro dos limites do algoritmo. Ainda assim, algumas configurações demoraram muitas horas para obter um resultado. Se considerarmos essas dimensões, é seguro dizer que métodos evolutivos não são eficazes para lidar com um

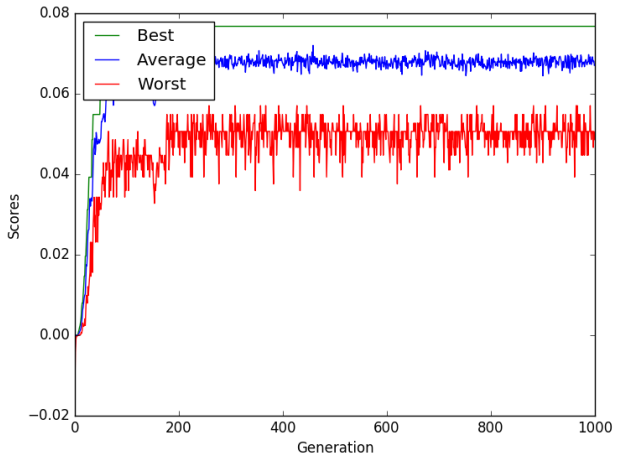

Figura 9. Gráfico da aptidão dos indivíduos pela geração do algoritmo para 50 crianças, 10 tipos de presentes, 70 indivíduos por população, máxima geração de 1000 e taxa de mutação de 65%.

número tão grande de dados no problema em questão. Por outro lado, após os ajustes, o trabalho se mostrou satisfatório para sua proposta.

É possível perceber também que quanto menos crianças e presentes, menor é o valor da solução ótima, já que uma criança ou presente infeliz tem mais influência no resultado da função de aptidão do que em uma configuração com mais crianças e presentes.

O impacto de uma taxa de mutação maior é notável, já que o *crossover* não proporciona a variedade de indivíduos necessária para que o algoritmo chegue a convergir em uma solução ótima, pois não interfere nos presentes dos gêmeos e trigêmeos, ao contrário da mutação.

O tamanho da população, por sua vez, influencia na convergência, em si, do algoritmo, mas exige um desempenho computacional maior.

Além disso, os resultados produzidos pelo algoritmo reforçam a teoria estudada previamente.

REFERÊNCIAS

[1] P. Norvig and S. Russell, *Artificial Intelligence: A Modern Approach*, 3rd ed. Elsevier, 2013. 1

[2] E. L. Colombini, "Computação evolutiva i." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula9.pdf 1

[3] ——, "Computação evolutiva ii." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula10.pdf 1

[4] ——, *Projeto 2*. [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/P2.pdf 1

[5] Kaggle, "Santa gift matching challenge." [Online]. Available: https://www.kaggle.com/c/santa-gift-matching/ 1, 4

[6] "Python 3." [Online]. Available: https://www.python.org/ 2

[7] "Pandas." [Online]. Available: https://pandas.pydata.org/ 2

[8] "Matplotlib." [Online]. Available: https://matplotlib.org/ 2