Algoritmo Genético: Busca pela melhor rota

Luiz Gustavo Rodrigues Martins e Rodrigo Ferreira Guimarães
Departamento de Ciência de Computação e Faculdade de Tecnologia
Universidade de Brasília, Brasília
E-mails: luiz.rodrigues@aluno.unb.br e rodrigofegui@aluno.unb.br
Matrículas: 13/0123293 e 14/0170740

I. Resumo

Aplicação da problemática do Caixeiro viajante ao contexto de rota de uma viagem entre amigos, tendo o ponto de partida fixo numa rota circular. O algoritmo genético foi utilizado para determinar a melhor rota, para um caso médio. A implementação foi em Python.

II. Embasamento Teórico

A computação evolutiva adapta a teoria evolutiva do Darwinismo à automatização da solução de problemas. Consiste, basicamente, em considerar uma população de indivíduos que crescerão, tendo sua adaptação ao ambiente avaliada, e passarão pela pressão da seleção natural. Os indivíduos mais ajustados poderão ser utilizados como semente na próxima geração [1].

Há diversos conjuntos de regras dentro da computação evolutiva como: algoritmos genéticos, estratégias evolutivas, programação evolutiva e programação genética. Algoritmos genéticos é o conjunto utilizado neste projeto.

A. Algoritmo Genético

Conjunto da computação evolutiva que utiliza a representação de sequência de *bits*, aplicando troca ou mudança de *bits* como suas operações [1].

Os indivíduos, que podem ser chamados de *cromossomos*, é composto por *genes* que armazenam grande significado em seu valor e na sua posição, onde uma alteração pode mudar completamente esse indivíduo. Além dos *bits*, inicialmente propostos, como genes outras estruturas podem ser utilizadas, adaptando-se ao problema que deseja ser resolvido.

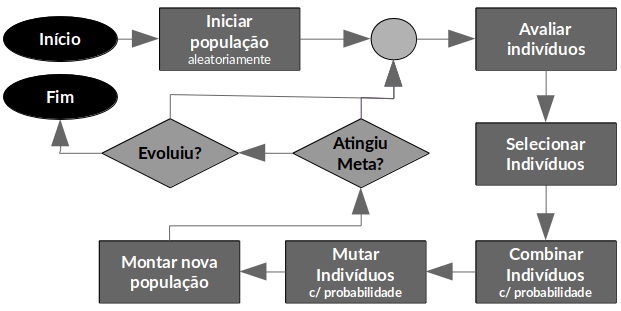

Figura 1: Etapas presente no algoritmo genético.

Conforme ilustrado na Figura 1, um algoritmo genético têm as seguintes etapas:

1) **Inicialização da população:** uma quantidade $\mathcal{N}$ de indivíduos é gerada aleatoriamente, respeitando a configuração dos cromossomos;
2) **Avaliação dos indivíduos:** assim como na teoria do evolucionismo, os indivíduos possuem sua adaptabilidade ao ambiente avaliada e mensurada e, então, utilizada para ordenação da população. Esta avaliação dos indivíduos está intrinsecamente vinculada ao problema em questão e deve ser avaliada como tal;
3) **Seleção de indivíduos:** com os dados referentes à adaptação dos indivíduos, somente os mais aptos podem ser selecionados para proliferar;
4) **Combinação de indivíduos:** com os mais adaptados separados ou não, os genes de um par de indivíduos são combinados de forma a trocar dados e poder gerar novos indivíduos;
5) **Mutação de indivíduos:** ainda considerando os mais adaptados ou não, os indivíduos possuem seus genes alterados internamente e pode gerar novos indivíduos;
6) **Montagem de uma nova população:** considerando a população atual, ou os indivíduos mais adaptados, e os indivíduos gerados, uma nova população é configurada e estará disponível como a próxima geração;
7) **Avaliação dos critérios:** este algoritmo poderia ser executado infinitivamente, mas como uma resposta está sendo esperada, critérios de parada devem ser definidos. Caso sejam alcançados: a última geração é considerada como provável portadora da resposta desejada; caso contrário, o processo é repetido a partir da **Avaliação dos indivíduos**.

Vale ressaltar que a **Avaliação dos critérios** informou que "a última geração é considerada como *provável* portadora da resposta desejada", por se tratar de uma busca pelo caso médio, não pelo melhor caso.

Embora todas as etapas tenham detalhes e minúcias, apenas a combinação e a mutação serão detalhadas. Para tanto, serão considerados cromossomos com 10 genes inteiros de 0 a 9, tendo o 9 como fixo na posição **0**, conforme demonstrado na Tabela I.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	4	6	0	1	2	7	5	8	3

Tabela I: Exemplo de cromossomo de 10 genes inteiros de 0 a 9.

B. Combinação

Quanto maior o cromossomo maiores as possibilidades de métodos para realizar a combinação de genes de dois indivíduos, eis algumas possibilidades:

- **Combinação de 2 pontos:** dados dois indivíduos (I_1 e I_2), duas posições de seus genes são sorteadas. Os genes fora do intervalo entre as posições serão copiados de I_1 para o filho F_1 e de I_2 para o filho F_2. Já os genes dentro do intervalo serão copiados de I_2 para F_1 e de I_1 para F_2, sendo verificado se o gene a ser copiado não está presente na parte fixa do filho, e, caso esteja, o filho mantém o gene do pai que forneceu sua parte fixa - adaptado [2]. Após as trocas de genes, ainda é possível haver duplicatas na região contida entre as posições sorteadas, isso é tratado substituindo um dos genes repetidos pelo gene não presente no filho - adaptado [3]. Exemplificando na Tabela II, as posições sorteadas foram a **4** e a **7**, então os genes das posições de 0 a 3 e de 8 a 9 (em negrito) serão fixos nos filhos F_1 e F_2, que recebem os genes fixos de I_1 e I_2 respectivamente. Os genes entre as posições 4 e 7 foram trocados evitando-se repetição com os genes fora desse intervalo (no exemplo, não houve troca nas posições que contém gene sublinhado, que, caso fosse trocado, geraria duplicação com gene fixo). No exemplo, ocorreu repetição entre genes dentro do intervalo e, com isso, um dos genes repetidos foi substituído pelo gene não presente no filho, como podemos observar na posição 5 de F_1, onde ocorreu repetição do gene 1 que foi substituído por 3, e na posição 4 de F_2, onde ocorreu repetição do gene 3 que foi substituído por 1.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	2	0	4	3	7	5	1	8	6
I_2	9	5	2	4	7	1	3	6	0	8
F_1	9	2	0	4	7	~~1~~ 3	5	1	8	6
F_2	9	5	2	4	~~3~~ 1	7	3	6	0	8

Tabela II: Exemplificação da combinação de 2 pontos.

- **Combinação de ordem 1:** dados dois indivíduos (I_1 e I_2), duas posições de seus genes são sorteadas, em seguida um dos indivíduos é escolhido como gerador. Dessa forma, os genes que estiverem no intervalo das duas posição são copiados para o filho (F_1), enquanto que o restante é reordenado aleatoriamente [4]. Exemplificado na Tabela III, as posições sorteadas foram a **2** e a **6** e o indivíduo I_1, então o trecho 04375 é copiado no F_1 e o resto rearranjado.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	2	0	4	3	7	5	1	8	6
I_2	9	5	2	4	7	1	3	6	0	8
F_1	9	1	0	4	3	7	5	6	2	8

Tabela III: Exemplificação da combinação de ordem 1.

- **Combinação ordenada:** dados dois indivíduos (I_1 e I_2), duas posições de seus genes são sorteadas, em seguida um dos indivíduos é escolhido como gerador. Dessa forma, os genes que estiverem no intervalo das duas posição são copiados para o filho (F_1), enquanto que o restante é reordenado em ordem crescente ou decrescente [2]. Exemplificado na Tabela IV, as posições sorteadas foram a **5** e a **7** e o indivíduo I_1, então o trecho 751 é copiado no F_1 e o resto rearranjado.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	2	0	4	3	7	5	1	8	6
I_2	9	5	2	4	7	1	3	6	0	8
F_1	9	8	6	4	3	7	5	1	2	1

Tabela IV: Exemplificação da combinação de ordenada na ordem decrescente.

- **Combinação uniforme:** dados dois indivíduos (I_1 e I_2), a cada posição algum dos indivíduos possui o gene sorteado para ser copiado ao filho (F_1). Se o problema não admitir duplicatas de genes, o filho pode ser descartado ou ter suas duplicadas tratadas (sofrer mutação). Exemplificado na Tabela V, o indivíduo I_1 foi copiado nas posições **1**, **4**, **5**, **7** e **9**, enquanto o I_2 nas posições **2**, **3**, **6** e **8**; infelizmente esta combinação gerou duplicadas e uma forma de resolvê-las é a aplicação de mutação (explicada na Seção II-C) aleatória nas primeiras aparições das mesmas, resultando no F_1'.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	2	0	4	3	7	5	1	8	6
I_2	9	5	2	4	7	1	3	6	0	8
F_1	9	2	2	4	3	7	3	1	0	6
F_1'	9	8	2	4	5	7	3	1	0	6

Tabela V: Exemplificação da combinação uniforme, com duplicatas tratadas.

C. Mutação

Com determinada probabilidade, um determinado indivíduo da população pode sofrer mutação em seus genes. Nesse processo, dois genes aleatórios trocam de valor entre si no cromossomo do indivíduo.

No exemplo da Tabela VI, os genes de valores 2 e 7 do indivíduo I_1 são selecionados para mutação. Com isso, os genes irão trocar de valor entre si, dando origem a I_1'.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
I_1	9	2	0	4	3	7	5	1	8	6
I_1'	9	7	0	4	3	2	5	1	8	6

Tabela VI: Exemplificação de uma mutação.

III. Desafio

A problemática do caixeiro viajante foi aplicada a um grupo de sete amigos que desejam realizar uma viagem por 10 cidades da América do Sul, passando por todas apenas uma vez numa rota circular, com ponto de partida fixo [5].

Somente duas informações são conhecidas sobre a viagem: as cidades - São Paulo (SAO), Salvador (SSA), Rio de Janeiro (RIO), Lima (LIM), Bogotá (BOG), Santiago

(SCL), Caracas (CCS), Belo Horizonte (CNF), Porto Alegre (POA) e Brasília (BSB) - e as distâncias diretas entre elas, conforme Tabela VII. Por ser uma matriz simétrica, somente a parte triangular superior foi representada.

A rota deve ser iniciada em BSB e retornar à mesma, devendo utilizar duas formas de reprodução e limitada em 1000 gerações.

	SAC	SSA	RIO	LIM	BOG	SCL	CCS	CNF	POA	BSB
SAC	-	17	3	35	43	26	44	5	8	9
SSA		-	20	31	47	11	51	22	8	23
RIO			-	38	45	29	45	3	11	9
LIM				-	19	25	27	36	33	32
BOG					-	43	10	43	46	37
SCL						-	49	30	19	30
CCS							-	42	48	35
CNF								-	13	6
POA									-	16
BSB										-

Tabela VII: Distâncias diretas entre as cidades, na escala de centenas de quilômetros (10^2 km)

IV. Implementação e Análise

A implementação foi realizada em Python, ver. 3.7.6, está disponível online no GitHub e foi modularizada em quatro arquivos: `variables`, `utils`, `genetic_algorithm` e `main`, com relacionamento demonstrado na Figura 2.

As constantes controlam o contorno do algoritmo genético, podendo alterar seu comportamento, tais como: **population_sz** é o tamanho da população, **crossover_pbty** é a probabilidade de combinação, **mutation_pbty** é a probabilidade de mutação, **current_indv_perc** é a porcentagem da geração atual a ser mantida na próxima geração, **stability_perc** é a porcentagem para ser considerável como estável e **stability_max** é a quantidade máxima de gerações a serem consideradas estáveis. Além disso, alguns trechos de código serão explicados:

- **etapas do algoritmo genético:** conforme demonstrado no Código 1, a população é ordenada considerando sua adaptabilidade ao ambiente, caso a estabilidade for atingida, o algoritmo é interrompido, caso contrário novos indivíduos são gerados a partir da combinação e da mutação da geração corrente;
- **combinação de cromossomos**: conforme demonstrado no Código 2, toda a geração é considerada como prováveis pais, mas os com adaptabilidade semelhantes formarão pares. Caso a probabilidade de combinação não tenha sido atingida, o par é ignorado, caso contrário um dos quatro tipos de combinação disponível é selecionado para o par.

Devido à sua natureza aleatória e baseada no evolucionismo, o algoritmo genético não terá o mesmo comportamento a cada geração para o mesmo conjunto de

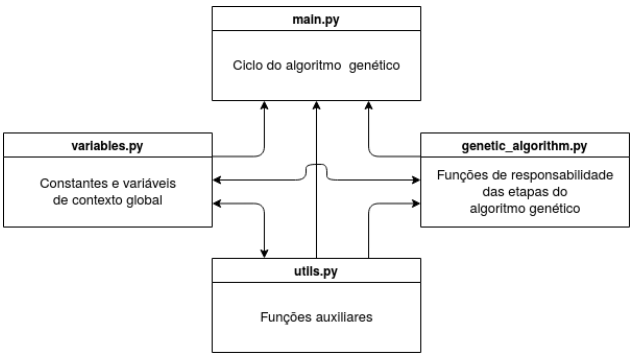

Figura 2: Relacionamento entre os arquivos da implementação.

Código 1: Trecho da implementação das etapas do algoritmo genético.

```
for cnt in range(GENERATION_MAX):
    generation = fitness(generation)

    if reached_stability(generation):
        break

    new_individuals = crossover(
        generation
    )
    new_individuals.extend(
        mutation(generation)
    )
```

parâmetros, uma vez que a Figura 3 encontrou a menor distância, 128, em poucas gerações, 88, enquanto que pela Figura 4 não foi encontrado a menor distância, obtendo 144, e utilizou todas as gerações possíveis; considerando os seguintes parâmetros iniciais: **population_sz** $= 40$, **crossover_pbty** $= 0.78$, **mutation_pbty** $= 0.1$, **current_indv_perc** $= 0.6$, **stability_perc** $= 0.1$ e **stability_max** $= 10$.

Utilizando o algoritmo com as constantes iniciais mencionadas anteriormente, foram encontradas, em algumas execuções do algoritmo, duas rotas para a distância mínima de 128: **Brasília -> Caracas -> Bogotá -> Lima -> Santiago -> Salvador -> Porto Alegre -> São Paulo -> Rio de Janeiro -> Belo Horizonte -> Brasília**, e **Brasília -> Belo Horizonte -> Rio de Janeiro -> São Paulo -> Porto Alegre -> Salvador -> Santiago -> Lima -> Bogotá -> Caracas -> Brasília**.

Com a finalidade de analisar o comportamento do algoritmo ao alterar alguns de seus parâmetros de contorno, foram realizadas modificações nos valores iniciais das constantes **population_sz**, **current_indv_perc**, **crossover_pbty** e **mutation_pbty**.

Na Figura 5, foi alterada a constante **population_sz** de 40 para 200. Com isso, observa-se que a média da distância a percorrer decai de forma mais suave, sem apresentar variações bruscas e, assim, atinge o intervalo de estabilidade em poucas gerações. Além disso, o algoritmo consegue atingir a distância mínima de 128 dentro do número de gerações até a estabilidade.

Código 2: Combinação de dois cromossomos.

```
def crossover(generation):
    new_individuals = []
    options = [
        _crossover_order1,
        _ordered_crossover,
        _uniform_crossover,
        _crossover_2_point
    ]

    for ind in range(0,len(generation),2):
        if random() > CROSSOVER_PBTY:
            continue

        new_individuals.extend(
            choice(options)(
                generation[ind],
                generation[ind + 1]
            )
        )

    return new_individuals
```

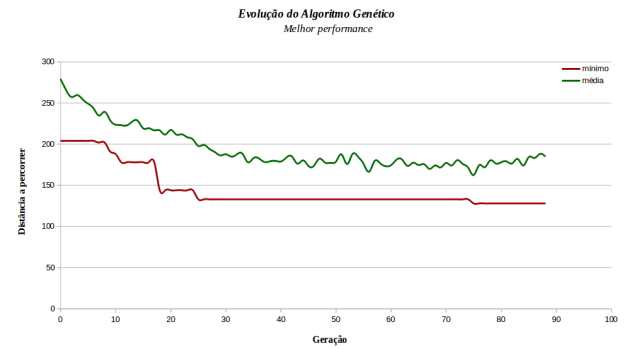

Figura 3: Um dos melhores comportamentos do algoritmo genético.

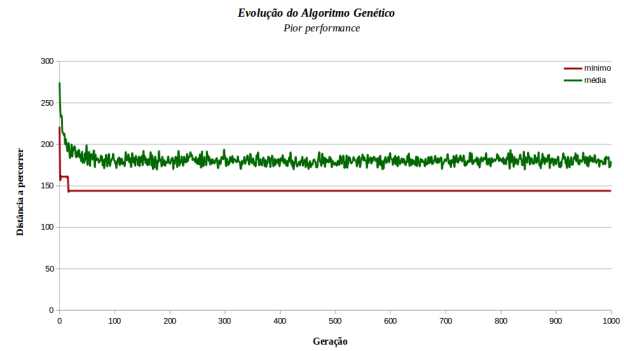

Figura 4: Um comportamento mediano do algoritmo genético.

Na Figura 6, foi reduzida a constante **current_indv_perc** de 0.6 para 0.1. Verificou-se que o algoritmo apresentou uma média da distância a percorrer mais elevada (acima de 200) e instável, o que levou a mais gerações até a estabilidade. Entretanto, ainda foi possível atingir a distância mínima de 128.

Na Figura 7, a constante **crossover_pbty** foi reduzida de 0.78 para 0.1. Observou-se que embora a distância mínima de 128 foi atingida em menos de 200 gerações, a menor porcentagem de cruzamento tornou a média da distância a percorrer mais instável e o algoritmo foi levado até o número máximo de 1000 gerações.

Por fim, na Figura 8, a constante **mutation_pbty** sofreu aumento de 0.1 para 0.3. Com essa alteração, observou-se que a distância mínima foi atingida por volta da 50ª geração. Esse valor baixo de gerações até encontrar a distância mínima pode ser explicado pelo fato de não haver genes repetidos em um indivíduo, o que pode tornar a mutação mais eficiente. Por outro lado, o algoritmo percorreu 250 gerações até atingir a estabilidade, pois a média da distância a percorrer ainda apresentou variação considerável.

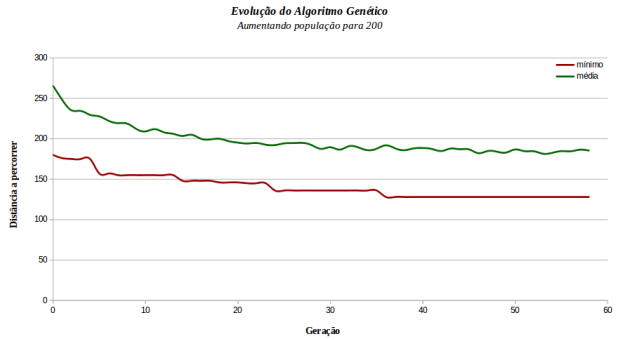

Figura 5: Comportamento aumentando o tamanho da população de 40 para 200.

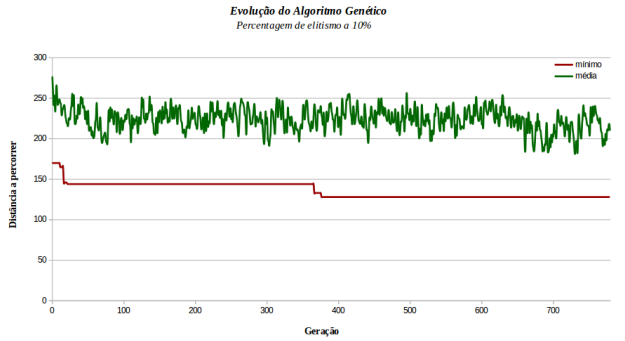

Figura 6: Comportamento ao reduzir a porcentagem da geração atual a ser mantida para a próxima geração de 0.6 para 0.1.

V. Conclusão

O algoritmo genético implementado foi capaz de obter as rotas com menor distância dentro do problema proposto de uma rota de viajem circular com 10 cidades, com ponto de partida fixo. A distância mínima de 128 foi atingida para duas rotas: **Brasília -> Caracas -> Bogotá -> Lima -> Santiago -> Salvador -> Porto Alegre -> São Paulo -> Rio de Janeiro -> Belo Horizonte -> Brasília**, e **Brasília -> Belo Horizonte -> Rio de Janeiro -> São Paulo -> Porto Alegre -> Salvador -> Santiago -> Lima -> Bogotá -> Caracas -> Brasília**.

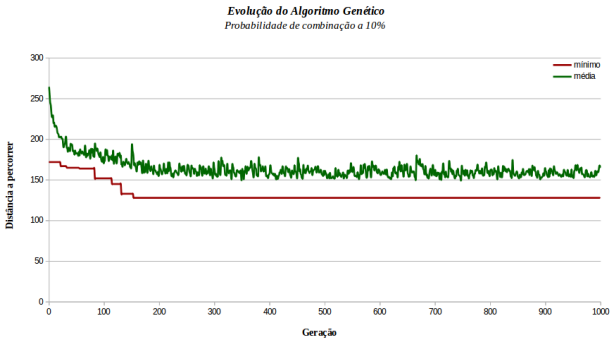

Figura 7: Comportamento ao reduzir a probabilidade de combinação de 0.78 para 0.10.

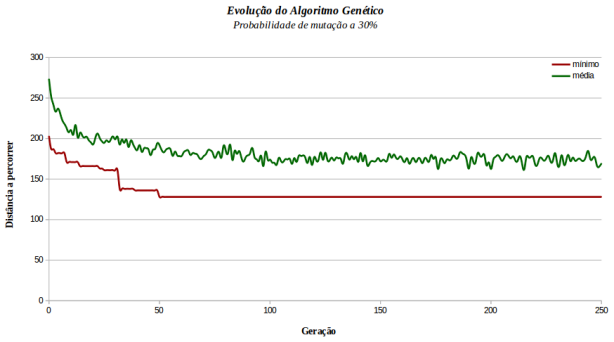

Figura 8: Comportamento ao aumentar a probabilidade de mutação de 0.1 para 0.3.

As estratégias de reprodução como a combinação e a mutação devem ser pensadas e escolhidas com a finalidade de otimizar os descendentes que irão substituir em parte a população anterior para dar origem a uma nova geração. No algoritmo proposto, a evolução das gerações promovem uma menor média da distância a percorrer (Figuras 3 e 4), o que gera melhor ajuste dos resultados e distâncias mínimas cada vez menores.

A importância em analisar o comportamento do algoritmo ante as constantes iniciais como, por exemplo, tamanho da população e probabilidades de combinação e mutação foi evidenciada na Seção IV. A escolha dos parâmetros que melhor se ajustam à problemática de um algoritmo genético permite a obtenção de resultados desejados ou aproximados dentro do número de gerações limite.

Portanto, os algoritmos genéticos são uma importante abordagem para o tratamento de problemas onde a otimização está presente. Por meio de ajustes através das gerações, os resultados obtidos tendem a evoluir em busca de melhor adaptação ao contexto imposto por um dado problema.

Referências

[1] A. Eiben and M. Schoenauer, "Evolutionary computing," *Information Processing Letters*, vol. 82, no. 1, pp. 1 – 6, 2002, evolutionary Computation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019002002041 [Accessed: 14/09/2020]

[2] E. Stoltz, "Evolution of a salesman: A complete genetic algorithm tutorial for python," Julho 2018. [Online]. Available: https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35 [Accessed: 14/09/2020]

[3] A. B. Ferreira, "Avaliação de operadores de algoritmos genéticos em otimização multidimensional," Master's thesis, Universidade Estadual Paulista (UNESP), 2007. [Online]. Available: https://repositorio.unesp.br/bitstream/handle/11449/88880/ferreira_ab_me_ilha.pdf?sequence=1&isAllowed=y [Accessed: 14/09/2020]

[4] D. L. Borges, "Introdução à Inteligência Artificial - Aula 05," Universidade de Brasília, Tech. Rep., 2020.

[5] ——, "Projeto 1 , Introdução à Inteligência Artificial, Turma A, 1/2020," Universidade de Brasília, Tech. Rep., 2020.