universidade de aveiro

Otimização Combinatória

Algoritmos Genéticos: ***O Problema do Caixeiro Viajante***

Autores:
Miguel Ramos nº94369

Regente da Unidade Curricular:
Dr. Agostinho Agra

Departamento de Matemática
Universidade de Aveiro

Ano lectivo 2019/2020

Junho de 2020

Conteúdo

1 Introdução

Em muitas situações práticas somos confrontados com problemas de otimização combinatória que são $\mathcal{NP}-$ *dificeis* e cujas instâncias são de elevadas dimensões tornando as abordagens exactas sem valor prático (por vezes as instâncias podem ter alguma estrutura especial para a qual exista um algoritmo eficiente, um caso particular do problema). Até mesmo quando confrontados com problemas que pertencem a $\mathcal{P}$ e de grande dimensão, algoritmos exactos podem não ser o melhor recurso para atacar o problema. Nestas situações a melhor estratégia é recorrer a uma metaheurística, um procedimento de aplicabilidade geral que apresenta uma estrutura genérica e indicações para o desenvolvimento de uma heurística específica para o problema em mão.

As metaheurísticas não são específicas a nenhum problema, são métodos completamente gerais que têm de ser adaptados ao problema em estudo e geralmente empregam algum passo de natureza estocástica. Alguns exemplos de metaheurísticas muito conhecidas são a *tabu search*, *simulated annealing* , *ant colony* e *algoritmos genéticos*. Os *algoritmos genéticos* são métodos baseados em populações, inspirado pelo processo de selecção natural na biologia.

1.1 O Problema do Caixeiro Viajante

O problema do caixeiro viajante é um exemplo de um problema $\mathcal{NP}-$ difícil. Dado um conjunto $V = \{v_1, v_2, ..., v_n\}$ de cidades e uma matriz de distâncias $d_{ij} = |d(v_i) - d(v_j)|$, encontrar o circuito de caminho mais curto que passa por todas as cidades sem ocorrência de repetição de cidades. Isto é, encontrar um circuito hamiltoniano num grafo pesado não-orientado.

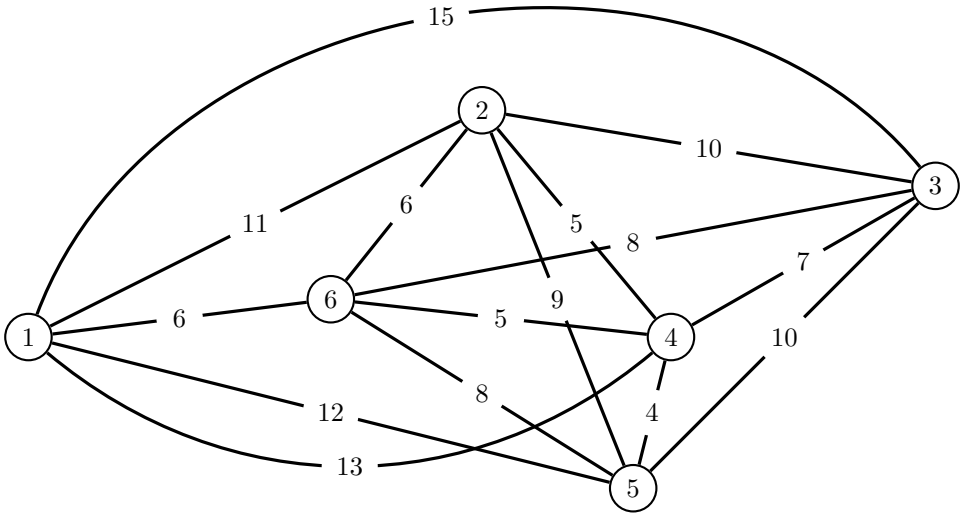


Figura 1: Exemplo

Um exemplo de uma solução seria

- $S := (1, 6, 4, 5, 3, 2)$

em que nesta notação representa o circuito $1 - 6 - 4 - 5 - 3 - 2 - 1$.

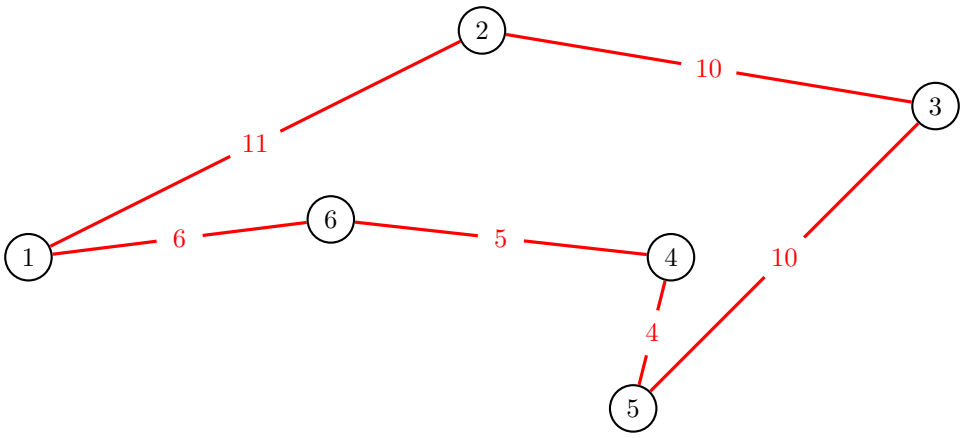


Figura 2: Solução exemplo $S := (1, 6, 4, 5, 3, 2)$

Para um grafo completo de n vértices existem $(n-1)!/2$ circuitos possíveis, ou seja, se considerar-mos $n = 10$ temos então $181,440$ percursos possíveis, se tivermos $n = 12$ temos $19,958,400$. Óbviamente que para uma instância de elevadas dimensões, uma busca exaustiva pela solução ótima do problema é impraticável.

2 Algoritmos Genéticos

Os Algoritmos Genéticos[4] [5] são inspirados na teoria biológica da selecção natural. Cada espécie na natureza, planta ou animal, apresenta uma grande variabilidade de indivíduos com diferentes características. Darwin observou que indivíduos que apresentam ter determinadas características que melhor se adaptam ao meio têm maior probabilidade de sobreviver e gerar descendência, a sobrevivência do mais apto. Numa espécie que se reproduza por reprodução sexuada, cada filho herda alguns dos cromossomas de ambos os pais, onde os genes nos cromossomas determinam as características indivíduais do filho. Um filho que acabe por herdar as melhores características dos pais é ligeiramente mais capaz de sobreviver até à idade adultar e por sua vez também gerar descendência e passar algumas dessas características para a geração futura.

A população tende a evoluir ao longo de gerações através deste processo. Um outro factor que desempenha um importante papel neste processo, de natureza aleatória também, são as mutações. As mutações são pequenas alterações num cromossoma e têm uma baixa probabilidade de ocorrer. Quando ocorre pode resultar numa característica benéfica para a sobrevivência do indivíduo, resultar numa debilitação ou ser simplesmente indiferente. Os filhos com mutações desejáveis terão uma maior probabilidade de sobrevivência e contribuir para a diversidade genética da espécie.

Podemos tratar um problema de otimização de uma forma bastante semelhante a este processo de uma forma muito natural. Uma solução admissível para o problema corresponde a um indivíduo da espécie. Um conjunto de soluções admissíveis serão o equivalente a uma população da espécie. A cada iteração nessa população irá ocorrer reprodução por um conjunto selecionado de soluções, gerando novas soluções que por sua vez irão competir com as soluções antigas para formar a nova população. Duas soluções admissíveis podem ser combinadas (ou cruzadas), de uma forma adequada, para gerar uma nova solução admissível. Essa solução admissível gerada têm a baixa probabilidade de sofrer uma mutação, que poderá revelar-se vantajosa.

As mutações são importantes, ajudam a que o algoritmo genético a explorar novas possíveis soluções, talvez melhores, do espaço de soluções. A sobrevivência do mais apto irá levar em princípio a que o algoritmo genético chegue a uma solução admissível que será quase-ótima pelo menos (um mínimo ou máximo local, dependendo do problema).

A analogia com a seleção natural não têm de ser seguida de forma rigida. Existem várias variantes e liberdade de escolha de funcionamento do mesmo. Por exemplo, há quem inclua o passo de mutação no ciclo de reprodução e há quem separe estes dois passos, criando primeiro todos os filhos e depois selecionando uma percentagem pequena que sofre mutação (assim garante-se que certamente pelo menos uma das crias irá sofrer mutação).

O que é exigido para se poder utilizar um algoritmo genético é[3]

- Uma representação codificada das soluções admissíveis do problema.
- A determinação de uma população inicial.
- A definição de uma função de avaliação de qualidade dos indivíduos.
- Definição de operadores adequados que permitam a produção de novos indivíduos, como também parâmetros que definem como ocorrem essas alterações (probabilidade de mutação por exemplo).
- Definição de um critério de paragem, tamanho da população e esquema de garantia de diversidade.

Segue um exemplo simples de um algoritmo genético em pseudo-código.

Algoritmo: Algoritmo Genético

```
Inicializar população ;
Avaliar população;
while !Condição de Paragem do
    selecionar os melhores indivíduos para reprodução;
    reproduzir os melhores indivíduos com possíveis mutações nas crias;
    avaliar os novos indivíduos;
    substituir os indíviduos menos capazes na população original pelas crias;
end
```

2.1 Representação das Soluções Admissíveis

Uma solução admissível deve ser representada de uma forma apropriada. A utilização de uma representação não apropriada pode levar a uma má performance do algoritmo genético (ou mesmo nem produzir uma solução viável). Para cada tipo de representação existem operadores de reprodução diferentes, caso a represetação para o nosso problema não seja apropriada pode-se cair no erro de produzir soluções não admissíveis no processo de reprodução.

Regra geral para um determinado problema surge sempre uma forma natural de representar a sua solução. No caso do algoritmo genético uma permutação das suas cidades (caso o grafo associado seja completo) representa uma solução admissível para o problema, como é representado no seguinte esquema.

1	2	3	4	5	6
001	010	011	100	101	110

Figura 3: Representação em forma de caminho(permutação das cidades do caixeiro viajante) e o mesmo caminho na sua representação em cadeia de *strings* binárias.

Outro tipo possível de representação é recorrer a cadeias *strings* binárias como é ilustrado no exemplo acima. No caso do caixeiro viajante existem várias possibilidades de representação:

- Strings Binárias
- Caminho

- Lista de Adjacência
- Lista Ordinal
- Matricial

No caso do caxeiro viajante, trabalhar com a representação de caminho é fácil e natural, em que um caminho é representado por uma sequência de cidades sem repetição de nenhuma delas, em que a primeira e a última cidade estão ligadas, formando o circuito hamiltoniano.

2.2 Geração de População Inicial

A população inicial deve ser gerada de forma a garantir a diversidade de soluções. Normalmente produzem-se soluções de forma aleatórias, mas se possível, caso exista alguma heurística computacionalmente viável para o problema deve-se recorrer a esta de forma a produzir já "boas soluções", isto irá guiar o processo de busca de forma a ser mais rápido a "convergir"para uma boa solução. No caso do caixeiro viajante existem várias heurísticas que podem ser utilizadas de forma a produzir boas soluções para uma população inicial, no entanto ter sempre cuidado para garantir diversidade populacional.

2.3 Selecção para Reprodução

A forma como se seleciona os pais têm impacto na solução final. Esta seleção é efetuada a partir de uma avaliação da qualidade de cada pai, dando um privilégio de reprodução aos indivíduos de melhor qualidade. No entanto há que ter cuidado no método de selecção pois este têm influência de dirigir a população para uma solução local de forma prematura. Não existe um suporte teorico que guie na escolha de um melhor método de selecção, portanto para um dado problema vários métodos devem ser testados. Existem vários métodos em literatura, como por exemplo o *Truncation Selection*, *Fitness Proportionate Selection* e o *Rank Based Selection*. Segue em exemplo dois deles.

2.3.1 Truncation Selection

Neste método de seleção, ordena-se os elementos da população pela sua qualidade, usando a função para avaliação de qualidade e escolhe-se uma percentagem dos melhores. Este método não é o mais adequado pois não garante diversidade dos pais para a reprodução, levando a uma convergência precoce para um ótimo local.

2.3.2 Fitness Proportionate Selection

Este método pega na população e calcula a qualidade de cada elemento usando a função de qualidade $F(p)$. Calcula então para cada elemento p na população a quantidade

$$P(p) = \frac{F(p)}{\sum_{i=1}^{N} F(p)}$$

definindo assim uma distribuição de probabilidade nos elementos da população, em que o elemento com maior qualidade têm maior probabilidade.

Recorrendo a um gerador de números aleatórios, selecionam-se os pais a partir desta distribuição, priveligiando assim que sejam escolhidos com maior probabilidade os elementos de melhor qualidade, mas permitindo que mesmo assim alguns dos menos bons sejam escolhidos mantendo assim a variabilidade.

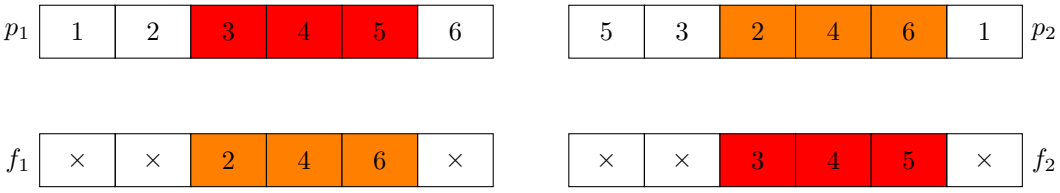
2.4 Reprodução

Para gerar descendência para a seleção de pais escolhida é necessário utilizar operadores que criem soluções válidas.

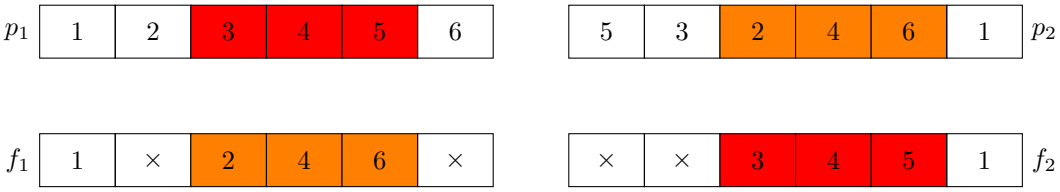
A escolha dos operadores de reprodução é dependente do tipo de representação escolhida para as soluções. No caso de uma representação de *caminho* (uma permutação das cidades que respeite os caminhos existentes e que representa um circuito) existem vários operadores em literatura dos quais os de mais comum utilização são o *Partially Mapped Crossover*, *Order Crossover*, *Cycle Crossover* e *Edge Recombination Crossover*[1][2]. Segue então a explicação de alguns deles.

2.4.1 Partially Mapped Crossover

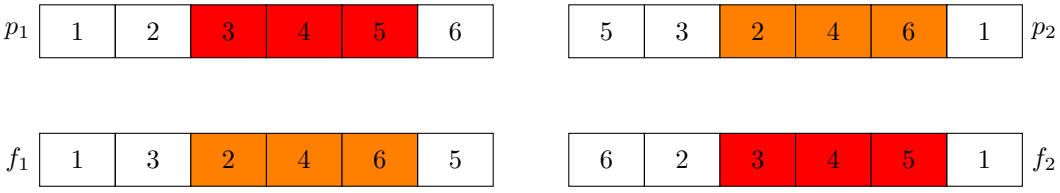
Este operador procede por escolher aleatoriamente dois pontos de corte aleatórios, após essa escolha é atribuída a secção de corte do primeiro pai ao segundo filho e a secção de corte do segundo pai é atribúida ao primeiro filho. Segue em exemplo a ilustração do prodecimento



Neste caso o corte gerado foi entre a terceira e quinta posição. Essas secções são copiadas para criar os filhos como na figura a cima. Agora verificando posição a posição as ligações entrada a entrada entre $p_1 \leftrightarrow f_1$ e $p_2 \leftrightarrow f_2$, preenchendo sempre que for válido com um gene do pai (ou seja, ceder o gene da 1º posição do pai à primeira posição do filho, caso o filho ainda não tenha esse gene).

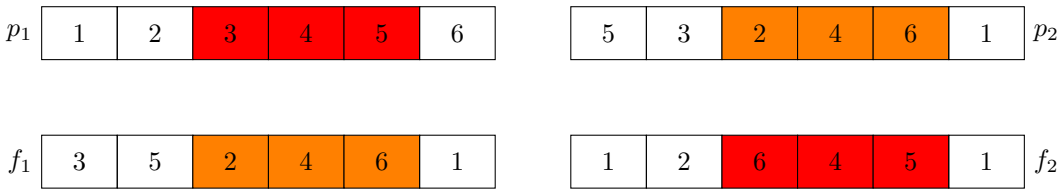


Verifica-se que no entanto alguns genes são incompativeis. O gene da segunda posição do p_1, 2, já existe no f_1 e foi herdado de p_2. Portanto vai-se ao p_2 e vê-se qual o gene na posição incompatível de f_1. Esse gene é o 3, que é compatível em f_1 e portanto pode-se inserir. Agora verifica-se o gene incompatível na sexta posição, o 6. Na posição equivalente no p_2 o gene é o 1, que é incompatível com f_1. Verifica-se então em f_1 qual é a posição associada do gene 1. É a primeira posição, portanto verifica-se em p_2 qual o gene correspondente na primeira posição. Esse gene corresponde ao 5 e é compatível com f_1. Procede-se então assim, seguindo este mapeamento entre os pais, obtendo as soluções finais:



2.4.2 Order Crossover

Este operador produz filhos escolhendo um subcaminho de um pai e preservando as ordens relativas das entradas do outro pai. Escolhendo o segundo corte do p_2, **1**, e construindo a sequência **1** 5 3 2 4 6, remove-se os elementos que foram resultantes do corte em p_1, 3 4 5, resultando na sequênica **1** 2 6. Agora insere-se essa sequência, a partir do segundo corte, no f_2. Procede-se de forma identica para formar f_1. De 6 1 2 3 4 5 remove-se 2 4 6, resutlando em 1 3 5, que é inserido em f_1 da mesma forma.



2.4.3 Edge Recombination Crossover

Esta técnica de reprodução tem como objectivo gerar filhos em que seja favorecida a presença de caminhos(arestas) que sejam comuns em ambos os pais, ou seja, combina os genes numa sequência tal que os genes permanecem vizinhos como eram no genótipo dos pais. Esta técnica baseia-se nas matrizes de adjacência, nas quais estão representados os vizinhos de cada gene em cada pai [6].

Segue um pequeno exemplo do *edge recombination crossover* aplicado ao grafo da figura 1, considerando como pais os cromossomas

- $P_1 : (1, 3, 2, 6, 4, 5)$
- $P_2 : (5, 3, 2, 4, 6, 1)$

Lista de Adjacência P_1	Lista de Adjacência P_2	União P_1P_2
$1 = \{3, 5\}$	$1 = \{5, 6\}$	$1 = \{3, 5, 6\}$
$2 = \{3, 6\}$	$2 = \{3, 4\}$	$2 = \{3, 4, 6\}$
$3 = \{1, 2\}$	$3 = \{2, 5\}$	$3 = \{1, 2, 5\}$
$4 = \{5, 6\}$	$4 = \{2, 6\}$	$4 = \{2, 5, 6\}$
$5 = \{1, 4\}$	$5 = \{1, 3\}$	$5 = \{1, 3, 4\}$
$6 = \{2, 4\}$	$6 = \{1, 4\}$	$6 = \{1, 2, 4\}$

Para construir uma nova solução procede-se da seguinte forma.

- Escolhe-se aleatóriamente um dos pais. Por exemplo o P_1. Seleciona-se a primeira entrada como primeiro gene da nova solução $f_1 := (1, \times, \times, \times, \times, \times)$
- Na lista de união vê-se quais os genes ligados a 1 e remove-se o 1. São $\{3, 5, 6\}$, então escolhe-se da lista de união, entre o 3,5 e 6 o elemento cujo conjunto associado têm menor cardinalidade. Caso haja cardinalidades iguais escolher aleatóriamente.

União P_1P_2
$1 = \{3, 5, 6\}$
$2 = \{3, 4, 6\}$
$3 = \{2, 5\}$
$4 = \{2, 5, 6\}$
$5 = \{3, 4\}$
$6 = \{2, 4\}$

- Escolhe-se o 3, portanto $f_1 := (1, 3, \times, \times, \times, \times)$. E remove-se o 3 das listas.

União P_1P_2

$1 = \{5, 6\}$

$2 = \{4, 6\}$

$3 = \{2, 5\}$

$4 = \{2, 5, 6\}$

$5 = \{4\}$

$6 = \{2, 4\}$

- O 3 tem ligação com $\{2, 5\}$. A lista com menor cardinalidade é o do 5, que contém o elemento 4. Portanto $f_1 := (1, 3, 5, \times, \times, \times)$

União P_1P_2

$1 = \{6\}$

$2 = \{4, 6\}$

$3 = \{2\}$

$4 = \{2, 6\}$

$5 = \{4\}$

$6 = \{2, 4\}$

- O elemento 4 tem o $\{2, 6\}$ na lista, escolhe-se o 2 por exemplo. Portanto $f_1 := (1, 3, 5, 2, \times, \times)$

União P_1P_2

$1 = \{6\}$

$2 = \{4, 6\}$

$3 = \{\}$

$4 = \{6\}$

$5 = \{4\}$

$6 = \{4\}$

- Na lista do 2 tem-se $\{4, 6\}$. Escolhe-se o 4, ficando $f_1 := (1, 3, 5, 2, 4, \times)$.

União P_1P_2

$1 = \{6\}$

$2 = \{6\}$

$3 = \{\}$

$4 = \{6\}$

$5 = \{\}$

$6 = \{\}$

No fim do ciclo, ficamos então com $f_1 := (1, 3, 5, 2, 4, 6)$. Caso ao longo do algoritmo, seja apontado um elemento com lista vazia procede-se escolhendo aleatóriamente um elemento que ainda não esteja na solução.

2.5 Mutação

A mutação serve para garantir a diversidade de soluções e que não se cai prematuramente num ótimo local. O procedimento de mutação pode ser a troca de posição de dois genes aleatórios, no caso to caixeiro viajante é uma forma válida de mutação. A forma de fazer uma mutação depende da representação e deve-se garantir que se preserva a admissibilidade das soluções.

2.6 Selecção de Sobreviventes

Após serem gerados os filhos estes devem ser usados para constituir a nova geração. Para este procedimento existem várias estratégias, uma das mais usadas é substituir os piores elementos da população original pelos filhos gerados. No entanto, não se tem necessáriamente de aceitar todos os filhos e pode-se colocar um critério que exclua um filho caso a sua aptidão seja pior do que algum dos piores elementos da população. No próprio processo de reprodução pode ser colocado um critério em que a produção de filho, se este tiver pior aptidão que os pais, deve ser excluido.

3 Resultados de uma implementação de um Algoritmo Genético para o Problema do Caixeiro Viajante

Na minha implementação decidi utilizar como critério de paragem um número máximo de iterações (gerações) decidido pelo utilizador. A população inicial é gerada aleatóriamente sem recurso a qualquer heurística do TSP e tem tamanho N. A cada iteração é avaliada a qualidade de cada elemento da população e ordena-se os elementos por ordem crescente de qualidade. A função de qualidade é a simples soma de todas as distâncias percorridas no caminho. Dessa lista ordenada são selecionados 50% dos melhores para gerarem descêndencia.

Para a reprodução decidi utilizar o *edge recombination crossover*. No ciclo de reprodução, itera-se um número de vezes equivalente à cardinalidade dos elementos em 50% da população, produzindo então um número de filhos em igual número a 50% da população original. Cada vez que um filho era produzido, tinha uma probabilidade baixa (5%) de sofrer mutação. Ao terminar a produção dos filhos, adicionava os mesmos à população original,voltava a ordenar a população por qualidade, e mantinha apenas os N melhores individuos como nova geração. Devo referir que este tipo de estratégia de truncação não é a mais correcta, pois tem tendência a resultar em solução não muito boas.

	1	2	3	4	5	6	7	8	9	10
1	0	928	987	1961	977	232	1610	1943	1361	1467
2	928	0	1656	577	1879	1584	1330	351	1909	470
3	987	1656	0	387	1575	332	1978	857	754	987
4	1961	577	387	0	1471	844	576	1615	129	92
5	977	1879	1575	1471	0	1358	727	434	872	671
6	232	1584	332	844	1358	0	1087	585	187	384
7	1610	1330	1978	576	727	1087	0	1151	1529	1361
8	1943	351	857	1615	434	585	1151	0	92	1507
9	1361	1909	754	129	872	187	1529	92	0	569
10	1467	470	987	92	671	384	1361	1507	569	0

Tabela 1: Instância aleatória de $n = 10$ para o problema do caixeiro viajante.

Para a instância gerada e para um número de 100 iterações e população de tamanho $N = 10$, obteve-se o seguinte comportamento geracional

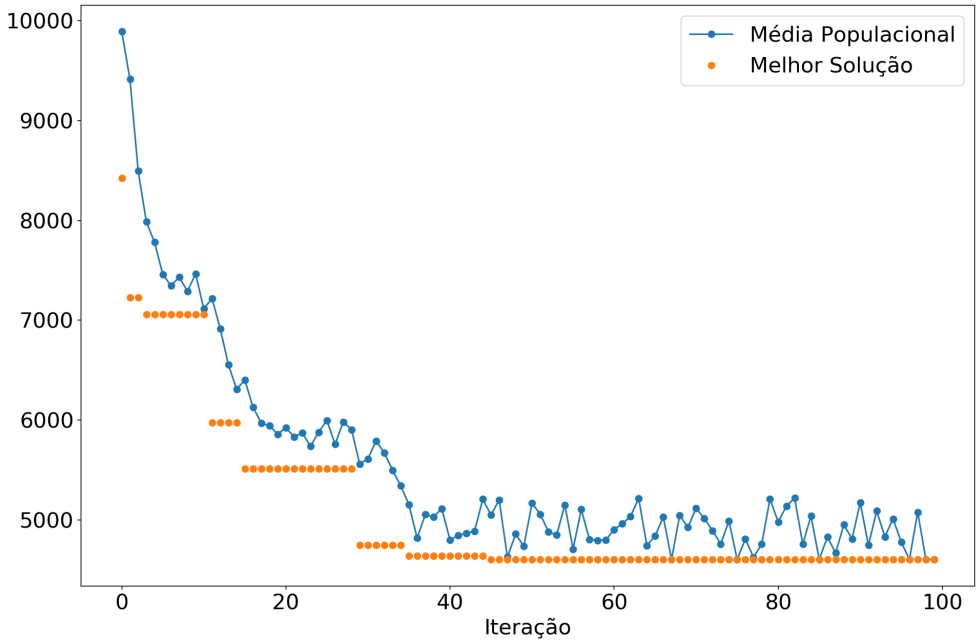


Figura 4: Comportamento da média populacional e da melhor solução, para uma população de tamanho $N = 10$ e número máximo de 100 iterações.

A melhor solução obtida tinha um custo 4603.0, com o caminho respectivo $C = (0, 5, 2, 8, 7, 1, 9, 3, 6, 4)$. Dado que a solução final depende da população inicial, deve-se repetir várias vezes o agoritmo, obtendo várias respostas e assim selecionar a melhor resposta da melhor repetição.

Em trabalho posterior devia implementar um melhor mecanismo de seleção dos pais e testar outros métodos de reprodução e mistura de reprodução e mutação.

Bibliografia

[1] Abid Hussain, et al.
Genetic Algorithm for Travelling Salesman Problem with Modified Cycle Crossover Operator
2017, Hindawai - Computational Intelligence and Neuroscience.

[2] P. Larrañaga,et al.
Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators
1999, Artificial Intelligence Review.

[3] M.C. Goldbarg, H.P.L. Luna
Otimização Combinatória e Programação Linear - Modelos e Algoritmos
2004, Elsevier.

[4] F.S. Hillier, G. J. Lieberman
Introduction to Operations Research - 9th edition
2010, McGraw Hill.

[5] El-Ghazali Talbi
Metaheuristics: From Design to Implementation
2009, Wiley.

[6] `http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/EdgeRecombinationCrossoverOperator.aspx`
Consultado a 7 de Junho de 2020.