UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

IA707/EG507 - COMPUTAÇÃO EVOLUTIVA

EFC I

Alunos
João Pedro O. Pagnan - 199727
João Pedro M. Ferreira - 218762

Professor
Levy Boccato

Campinas, 18 de abril de 2023

Neste exercício, vamos abordar a instância do QAP tratada em [1], que estudou a alocação de 19 clínicas em um departamento do Hospital Ahmed Maher, no Cairo, Egito, considerando o fluxo de pacientes entre elas, com o auxílio de um algoritmo genético (GA, do inglês *genetic algorithm*). O arquivo `elshafei_QAP.mat` traz a matriz $\mathbf{W}$ com os fluxos previstos entre as instalações, assim como a matriz $\mathbf{D}$ com as distâncias entre as localidades.

a) Escolha uma representação adequada para as soluções candidatas e uma função de avaliação (*fitness*). Discuta as razões que o motivaram a tais escolhas, além de apontar a cardinalidade do espaço de busca.

Uma representação possível para as soluções deste problema seria considerar cada solução seja representada por um vetor $\mathbf{x} \in \mathbb{N}^n$, sendo n o número de instalações e localidades, tal que:

$$\mathbf{x} = [x_0 \; x_1 \; ... \; x_{n-1}]^T,$$

sendo que, $1 \leq x_o \leq n$ para $o \in \mathbb{N}, 0 \leq o \leq n-1$, de forma que a posição de cada elemento do vetor $\mathbf{x}$ simboliza uma localização e o valor do elemento, dado por $p \in \mathbb{N}, 1 \leq p \leq n$, representa a p-ésima instalação.

Observação: Como o algoritmo será implementado em PYTHON, o índice inicial dos vetores foi considerado como sendo igual a 0. Apesar disso, as instalações variam de 1 a 19.

Esta representação foi escolhida pois, além de ser a utilizada em [1], permite representar cada solução por um vetor de n inteiros entre 1 e n, ou seja, representar cada solução como se fosse uma cadeia de números, facilitando a aplicação de diversos operadores de mutação. Caso fossemos representar cada solução por uma matriz $n \times n$, em que $x_{ij} \in 0, 1, 0 \leq i, j \leq n-1$, a aplicação desses operadores seria mais complexa.

A representação escolhida possui cardinalidade $n!$, afinal, trata-se da permutação de um conjunto de n elementos, indicando que, caso n for muito grande, uma solução por força bruta demorará para chegar na solução ótima, ou nem a alcançará. Para esse problema, temos que $n = 19$, de forma que a cardinalidade é igual a $19!$.

A função de avaliação que será utilizada é uma variação da função custo. No caso, deseja-se minimizar a seguinte função custo:

$$f(\mathbf{x}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \boldsymbol{D_{ij} W_{x_i x_j}}, \tag{1}$$

sendo D_{ij} a distância entre as localidades i e j e $W_{x_i x_j}$ o fluxo entre as instalações dadas por x_i e x_j, respectivamente.

Como a função custo está relacionada a uma minimização, foi realizado um mapeamento dela para o algoritmo genético buscar maximizar o *fitness*, dado por:

$$\phi(f(\mathbf{x})) = \frac{1/f(\mathbf{x})}{1/17212548} \tag{2}$$

Com isso, valores menores da função custo das soluções produzem *fitness* maiores e, como o algoritmo busca maximizar esta grandeza, ele irá o menor custo.

Também vale ser ressaltado que, como o custo ótimo do problema é conhecido, seu valor foi utilizado para que a função de ajuste fique num intervalo entre 0 e 1, sendo que este último corresponde a solução ótima.

b) Descreva um algoritmo genético adaptado ao QAP tendo em vista a codificação adotada. Justifique as escolhas feitas com relação aos operadores (e.g., seleção) e parâmetros do algoritmo (e.g., tamanho da população e critério de parada).

O algoritmo genético a ser utilizado possui as seguintes etapas:

1. Gerar a população inicial com N indivíduos;
2. Calcular o *fitness* para cada indivíduo da população inicial;
3. Aplicar o operador de seleção para selecionar dois indivíduos baseado em seus *fitness*;
4. Aplicar o operador de recombinação para gerar dois descendentes;
5. Aplicar o operador de mutação nos descendentes para que os respectivos cromossomos tenham uma probabilidade p_m de serem mutados;
6. Caso o número total de indivíduos e descendentes for menor que $2N$, repetir do passo 3 ao 5;
7. Calcular o *fitness* dos descendentes;
8. Eliminar os N indivíduos de menor *fitness*;
9. Repetir do passo 3 ao 8 até o critério de parada ser atingido.

A **geração da população inicial** será da forma indicada no processo 1, sendo N o tamanho da população e n o número de instalações e localidades.

Vale ser mencionado que, para esta representação, composta por uma permutação de n números diferentes, a quantidade de alelos é igual ao tamanho do cromossomo. Além disso, cada indivíduo da população irá armazenar o seu cromossomo, seu valor de *fitness* e seu custo.

O **cálculo do *fitness*** seguirá a equação 2 e pode ser descrito pelo processo 2.

A função `calc_custo_qap()` calcula o custo para a respectiva solução. Com isso, o algoritmo determina os valores da função de avalização e do custo de cada indivíduo da população.

Na quarta etapa, o GA aplica o **operador de seleção de torneio** na população. Esse método utiliza o parâmetro q, que é igual a quantidade de indivíduos da população selecionados por torneio. Esse método é executado duas vezes no algoritmo genético, de forma a determinar dois pais para a recombinação e é descrito pelo processo 3.

Este operador foi escolhido por possibilitar o ajuste da pressão seletiva na população pelo parâmetro q.

Processo 1 Geração aleatória da população inicial (`gerar_pop(N, n)`)

Paramêtros: Tamanho da população desejada e tamanho do cromossomo
Saída: População gerada

```
população ← [ ]
i ← 0
enquanto i < N faça
    cromossomo ← [ ]
    j ← 0
    enquanto j < n faça
        gene ← alelo ∈ {1, 2, ..., n}
        enquanto gene ∈ cromossomo faça
            gene ← alelo ∈ {1, 2, ..., n}
        fim enquanto
        cromossomo [j] ← gene
        j ← j + 1
    fim enquanto
    fitness ← 0
    custo ← 0
    população [i] ← [cromossomo, fitness, custo]
    i ← i + 1
fim enquanto
retorna população
```

Processo 2 Cálculo do *fitness* e do custo (`calc_fitness_qap(populacao, D, W)`)

Entradas: população, matriz com as distâncias e matriz com os fluxos
Saída: população com os valores de *Fitness* e custo atualizados

```
para indivíduo em população faça
    se indivíduo[1] = 0 faça
        custo = calc_custo_qap(indivíduo, D, W)
        fitness = (1/custo)/(1/17212548)
        indivíduo[1] = fitness
        indivíduo[2] = custo
    fim se
fim para
retorna população
```

Em seguida, na quinta etapa, o **operador de recombinação** que será aplicado foi o método de *crossover* ordenado (OX1), devido a sua facilidade de implementar e por permitir ser aplicado a permutações. Sendo $P1$ e $P2$ os pais e $D1$ e $D2$ os descendentes, esse método é descrito pelo processo 4.

Como pode ser observado, decidiu-se que será utilizada apenas uma seção do cromossomo do pai escolhido para o descendente e esta seção não será circular. Isso foi escolhido para facilitar a implementação do código e ainda assim permitir que haja uma recombinação das permutações.

Por fim, resta detalhar o **operador de mutação** utilizado. Neste caso, foi escolhido

Processo 3 Operador de seleção por torneio (`selecao_torneio(populacao, N, q_torneio)`

Entrada: população
Parâmetros: quantidade de indivíduos da população selecionados por torneio e tamanho da população inicial
Saída: o melhor indivíduo escolhido no torneio

$i \leftarrow 0$
indivíduos participantes $\leftarrow [\,]$
enquanto $i < q$ **faça**
    índice indivíduo selecionado $\leftarrow n \in \{0, 1, ..., N-1\}$
    indivíduos participantes $[i]$ $\leftarrow$ população [índice indivíduo selecionado]
    $i \leftarrow i + 1$
fim enquanto
indice do melhor indivíduo $\leftarrow$ indivíduos participantes[`argmax`(indivíduos participantes [:,1]])
melhor indivíduo $\leftarrow$ indivíduos participantes[índice do melhor indivíduo]
retorna melhor indivíduo

a mutação de reverter a ordem dos genes presentes em um segmento do cromossomo. Além disso, foi escolhido que a mutação tem uma probabilidade p_m de acontecer, ou seja, dela não acontecer no cromossomo do descendente e ser mantido o cromossomo original.

O processo pode ser descrito da forma indicada no processo 5.

Novamente, escolhemos não considerar o cromossomo e o genótipo como algo circular para facilitar a programação desse operador.

Com isso, foram descritos os operadores escolhidos para serem utilizados na geração de indivíduos desse algoritmo genético.

Estes operadores serão executados nessa sequência até que a população tenha $2N$ soluções, quando ocorrerá uma eliminação dos N indivíduos de menor *fitness*. Esse processo será repetido até o critério de parada ser atingido.

O **critério de parada** escolhido para o algoritmo foi o número de gerações n_g.

Esse algoritmo tem, portanto, os seguintes parâmetros:

- N: tamanho da população inicial;
- q: quantidade de indivíduos presentes no torneio;
- p_m: probabilidade de ocorrer uma mutação;
- n_g: número de gerações.

O valor de cada parâmetro foi determinado de maneira exploratória, sendo que os valores escolhidos estão indicados na tabela 1.

Processo 4 Operador de recombinação (`crossover_ox1(cromossomo_p1, cromossomo_p2)`)

Entrada: cromossomo dos dois pais
Saída: cromossomo de ambos os filhos

$i \leftarrow 0$
pais $\leftarrow [cromossomo\_p1, cromossomo\_p2]$
$n_{alelos} \leftarrow$ `len(cromossomo_p1)`
descendentes $\leftarrow [\,]$
enquanto $i < 2$ **faça**
  cromossomo do descendente $\leftarrow [\,]$
  posição de corte 1 $\leftarrow p \in \{0, 1, 2, ..., n-2\}$
  posição de corte 2 $\leftarrow p \in \{$posição de corte 1 + 1, posição de corte 1 + 2, ..., $n-1\}$
  seção do cromossomo $\leftarrow$ pais $[i]$[posição de corte 1 : posição de corte 2]
  cromossomo do descendente [posição de corte 1 : posição de corte 2] $\leftarrow$ seção do cromossomo
  locus $\leftarrow 0$
  locus pai $\leftarrow 0$
  enquanto locus $< n_{alelos}$ **faça**
    se locus $<$ posição de corte 1 **ou** locus $>$ posição de corte 2 **faça**
      enquanto pais[i - 1][locus pai] $\in$ cromossomo do descendente **faça**
        locus $\leftarrow$ locus pai $+1$
      fim enquanto
      cromossomo do descendente[locus] $\leftarrow$ pais$[i-1]$[locus pai]
      locus $\leftarrow$ locus + 1
      locus pai $\leftarrow$ locus + 1
    caso contrário
      locus $\leftarrow$ locus $+1$
    fim se
  fim enquanto
  descendentes $[i] \leftarrow$ [cromossomo do descendente, 0, 0]
  $i \leftarrow i + 1$
fim enquanto

$\boldsymbol{N}$	1000
$\boldsymbol{q}$	10
$\boldsymbol{p_m}$	0.5
$\boldsymbol{n_g}$	20

Tabela 1: Valores para os parâmetros do algoritmo genético.

c) Execute o GA construído no item b) 10 vezes, gerando, em cada caso, uma população inicial aleatória. Apresente a melhor solução obtida pelo algoritmo e o respectivo custo em cada execução. Além disso, compute o custo médio e o desvio padrão nesse conjunto de execuções. Comente.

Executando o algoritmo construído no item anterior 10 vezes, e em cada caso ge-

Processo 5 Operador de mutação (`mutacao_reversao(cromossomo)`)

Entrada: cromossomo a ser mutado
Parâmetros: probabilidade da mutação ocorrer
Saída: cromossomo mutado

número $\leftarrow k \in \{0, ..., p_m\}, k \in \mathbb{R}$
se número $<= p_m$ **faça**
  $n \leftarrow$ `len(cromossomo)`
  cromossomo mutado $\leftarrow [\,]$
  índice 1 $\leftarrow m \in \{0, 1, 2, ..., n-2\}$
  índice 2 $\leftarrow n \in \{$índice 1 + 1, índice 1 + 2, ..., $n-1\}$
  $i \leftarrow 0$
  $j \leftarrow 0$
  enquanto $i < n$ **faça**
    se $i \geq$ índice 1 **e** $i \leq$ índice 2 **faça**
      cromossomo mutado $[i] \leftarrow$ cromossomo [índice 2 $-j$]
      $j \leftarrow j + 1$
    caso contrário
      cromossomo mutado $[i] \leftarrow$ cromossomo $[i]$
    fim se
    $i \leftarrow i + 1$
  fim enquanto

rando uma nova população inicial aleatória, foram obtidas as melhores soluções e custos mínimos que estão mostrados na tabela 2.

Realização	Melhor solução	Custo
1	9,10,7,11,12,18,6,17,4,14,5,13,19,8,15,16,1,2,3	17539688
2	9,10,7,14,11,12,5,17,6,18,8,13,19,4,15,16,1,2,3	17382824
3	9,10,7,19,11,18,13,17,8,14,4,5,12,6,16,15,1,2,3	17240740
4	11,12,8,14,9,18,5,17,6,19,4,13,7,10,15,16,1,2,3	17464124
5	11,12,6,18,9,14,13,17,4,19,10,5,7,8,16,15,1,2,3	17433628
6	11,12,6,19,9,18,8,17,5,14,10,13,7,4,15,16,1,2,3	17464436
7	9,10,12,11,19,7,13,17,8,14,4,5,18,6,15,16,1,2,3	17540626
8	15,16,10,12,11,7,5,17,8,14,4,13,9,6,19,18,1,2,3	18099188
9	11,12,9,14,18,19,4,10,8,6,3,13,7,5,16,15,17,1,2	18683638
10	19,8,7,18,9,14,13,17,6,11,10,5,12,4,16,15,1,2,3	17478596
Solução ótima	9,10,7,18,14,19,13,17,6,11,4,5,12,8,15,16,1,2,3	17212548
Custo médio e desvio padrão do conjunto de execuções		17632748 ± 409315.2

Tabela 2: Melhores soluções e custos para as 10 execuções do algoritmo genético, juntamente com o custo médio e desvio padrão deste conjunto.

É interessante notar alguns aspectos importantes quando analisamos as melhores soluções e os custos obtidos. Primeiramente, vê-se que o desvio padrão dos custos obtidos foi de 2.32% do valor médio dessa grandeza, indicando que o algoritmo genético

projetado está tendo uma boa precisão ao gerar novas "ótimas"soluções quando é executado. Além disso, o desvio do valor médio em relação ao custo ótimo é de 2.44%, mostrando que o algoritmo também possui uma boa exatidão.

Em relação às melhores soluções de cada execução, é interessante notar certos padrões nas permutações obtidas. Vê-se que diversas permutações da tabelas possuem a sequência $15, 16, 1, 2, 3$ no fim delas, podendo haver uma troca nas posições dos números 15 e 16. Essa sequência também está presente na solução ótima, indicando que, possivelmente, se o algoritmo ser executado por mais gerações, ele pode convergir para a melhor permutação. Também nota-se outras sequências que também estão presentes na solução ótima, como sequência $9, 10$ nas primeiras posições do cromossomo.

d) Para cada uma das 10 realizações do item c), apresente as curvas de custo médio da população, bem como do custo do melhor indivíduo (ou seja, do custo mínimo) em função do número de gerações. Discuta o desempenho do algoritmo à luz dos resultados obtidos, das curvas de custo geradas e das próprias características do problema.

Registrando a evolução do custo médio na população e do melhor custo em cada geração para cada uma das dez realizações executadas do algoritmo genético, foram construídas as curvas indicadas na figura 1.

É interessante notar que, em todas as realizações, o custo mínimo foi sempre bem abaixo do custo médio na geração 0, ou seja, em relação ao custo médio da população inicial aleatória. Nas gerações seguintes, vê-se que houve uma queda brusca do custo médio da população, se assemelhando a uma função exponencial decrescente, indicando que os genes do cromossomo do melhor indivíduo da população inicial estão se propagando pelas gerações e, ao cromossomo se recombinar com o cromossomo de outros indivíduos ao longo das gerações do algoritmo, se torna mais parecido com o da solução ótima.

Além disso, como indicado no item anterior, sequências presentes na permutação da solução ótima estão presentes em diversas melhores soluções das dez execuções realizadas, indicando que, com o passar das gerações, características do genótipo que levam ao custo mínimo estão surgindo na população devido aos operadores implementados.

Desta forma, é de se esperar que, caso o número de gerações fosse maior ainda ou houvesse um aumento no tamanho da população inicial, de forma a introduzir maior variedade nela, o algoritmo seja capaz de se aproximar ainda mais da solução ótima, ou, até mesmo, alcançá-la.

Portanto, pode-se concluir que o algoritmo genético desenvolvido teve um bom desempenho em encontrar soluções ótimas para esse QAP.

e) Implemente, por fim, uma busca aleatória e faça 10 execuções independentes considerando a mesma quantidade de avaliações da função de *fitness* que o GA consumiu. Compare os resultados obtidos pelos dois métodos.

O algoritmo da busca aleatória é consideravelmente mais simples que o algoritmo

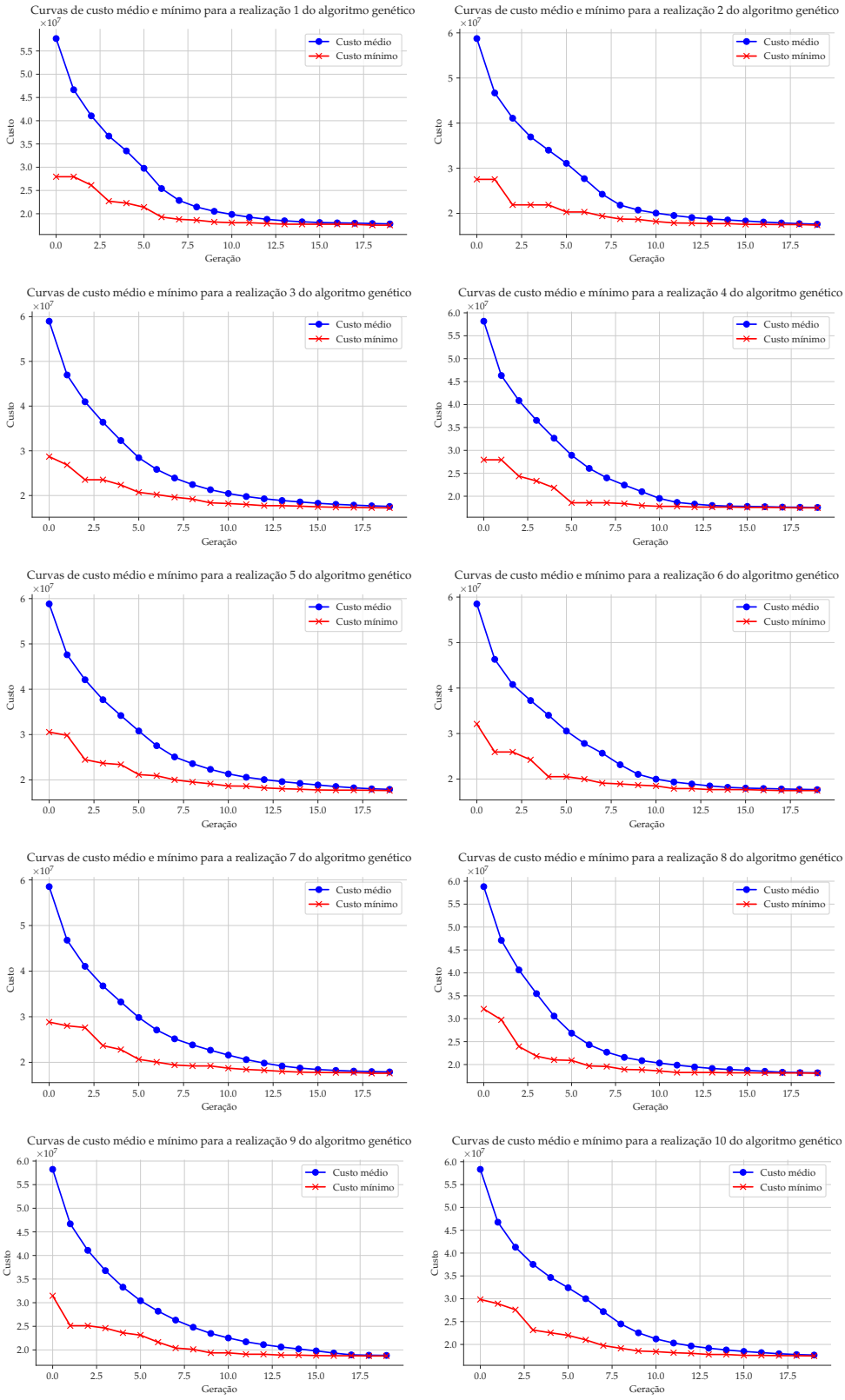

Figura 1: Curvas do custo médio da população e do custo mínimo para cada uma das dez realizações do algoritmo genético.

genético implementado e pode ser descrito da seguinte forma:

1. Gerar N soluções (permutações) aleatórias;
2. Calcular o *fitness* de cada solução utilizando a mesma função do algoritmo genético;
3. Selecionar, ou atualizar, a melhor solução com base no melhor valor da função de avaliação;
4. Repetir do passo 1 ao 3 até o critério de parada ser atingido.

No caso, a busca aleatória realiza a mesma quantidade de avaliações do valor de *fitness* para o conjunto de soluções por iteração, afinal, o critério de parada é igual ao do algoritmo genético.

Portanto, os parâmetros da busca aleatória são dados pela tabela 3, sendo n_i o número de iterações da busca executadas até o critério de parada ser atingido.

$\boldsymbol{N}$	1000
$\boldsymbol{n_i}$	20

Tabela 3: Valores para os parâmetros da busca aleatória.

Implementando um algoritmo de busca aleatória, com dez avaliações da função de *fitness* escolhida e mantendo a solução aleatória que possui o melhor valor para essa função, foram obtidos os resultados nas dez realizações indicados na tabela 4.

Realização	**Melhor solução**	**Custo**
1	19,11,14,7,4,12,13,3,6,9,18,17,8,10,16,15,2,5,1	24203108
2	15,16,10,7,9,5,14,18,8,6,3,4,11,13,19,12,1,17,2	23210176
3	13,7,11,12,10,6,17,3,9,19,8,14,18,5,15,16,4,2,1	24851468
4	16,15,11,14,18,10,12,6,9,4,17,19,7,8,13,5,1,2,3	24988252
5	12,14,5,19,11,9,13,7,3,6,10,8,17,4,15,16,18,1,2	25264160
6	10,5,9,11,12,7,8,13,3,18,17,14,6,4,16,15,1,2,19	24630270
7	9,18,19,7,13,14,5,3,11,4,10,17,12,8,16,15,1,2,6	23987840
8	5,19,14,17,3,8,6,9,18,2,4,13,1,10,16,15,11,12,7	26913084
9	11,9,19,12,16,14,5,17,4,15,8,10,18,6,13,7,1,3,2	24220168
10	8,14,10,5,11,12,17,1,6,4,7,9,19,18,16,15,3,2,13	23383336
Solução ótima	9,10,7,18,14,19,13,17,6,11,4,5,12,8,15,16,1,2,3	17212548
Custo médio e desvio padrão do conjunto de execuções		24565186 ± 1004124.28

Tabela 4: Melhores soluções e custos para as 10 execuções da busca aleatória, juntamente com o custo médio e desvio padrão deste conjunto.

E, utilizando a progressão do custo mínimo e médio do conjunto de soluções, foram obtidas as curvas apresentadas na figura 2.

Analisando as soluções obtidas em cada realização, e seus respectivos valores de custo, vê-se que a busca aleatória não foi nem próxima de se aproximar da solução

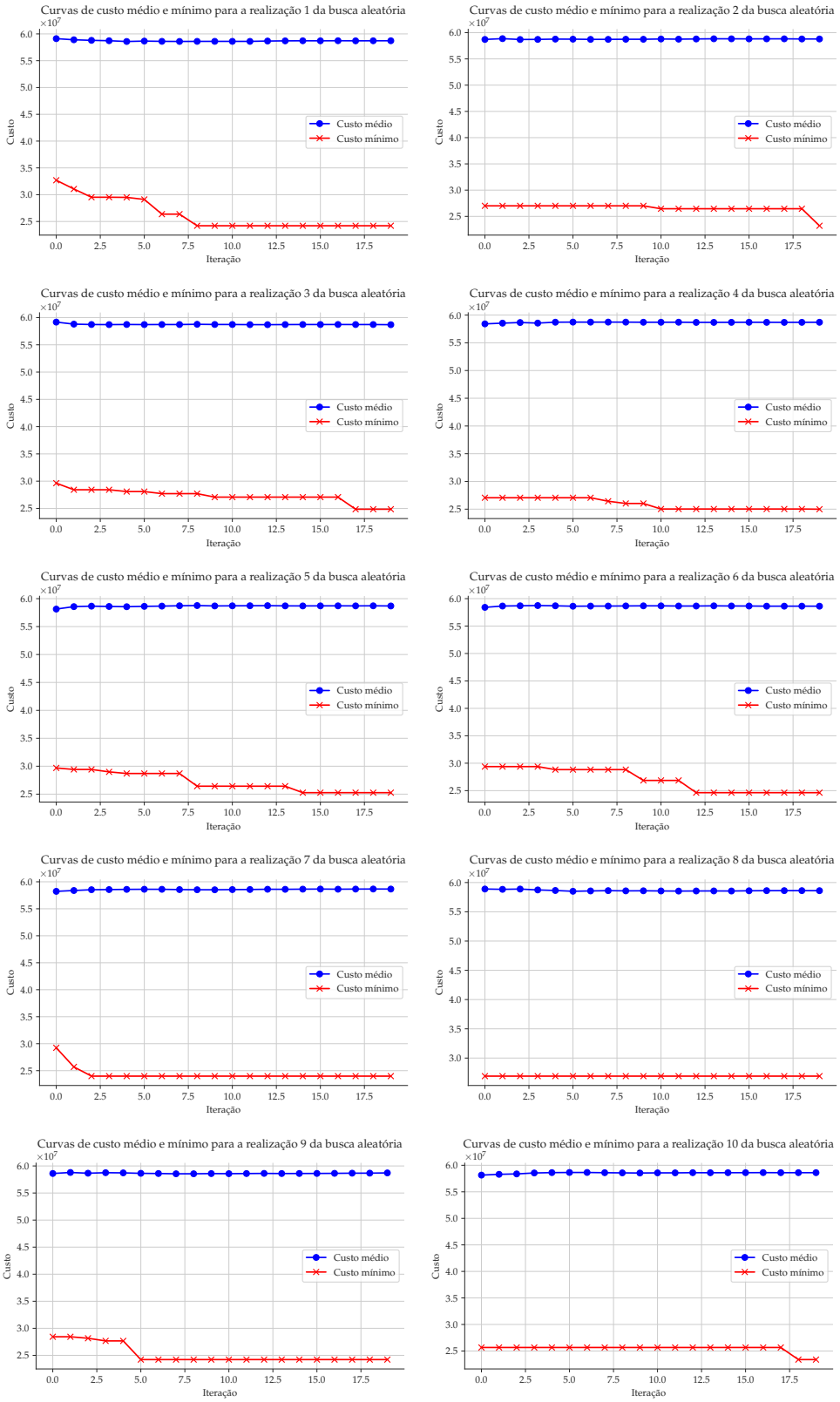

Figura 2: Curvas do custo médio da população e do custo mínimo para cada uma das dez realizações da busca aleatória.

ótima. Apesar do desvio padrão ser cerca de 4% do valor médio do custo do conjunto de soluções indicado, observa-se que o desvio do valor médio em relação ao custo

ótimo foi de 42.7%, indicando que uma busca aleatória seria um método muito ineficaz para resolver esse QAP, como esperado. Um outro fator que deve ser mencionado é que, como mencionado anteriormente, a cardinalidade desse problema é $19!$, de forma que há cerca de 1.216×10^{17} soluções para esse problema, evidenciando que uma busca aleatória seria extremamente ineficaz para resolvê-lo.

Quando as curvas do custo mínimo e médio ao longo das iterações são analisadas, vê-se algumas características interessantes. A primeira delas é que não há nenhuma grande redução em ambas as curvas, afinal, nenhum método de busca ou processo evolutivo está sendo utilizado. Além disso, vê-se que o valor do custo médio no conjunto de soluções é próximo ao valor dessa grandeza na população inicial, bem como o valor mínimo sendo próximo ao valor obtido pela melhor solução na geração inicial.

Desta forma, vê-se que o algoritmo genético possui um desempenho bem superior a busca aleatória quando aplicado a resolução de problemas quadráticos de alocação. Caso n fosse menor e houvesse menos permutações possíveis, a busca aleatória poderia ter um desempenho mais comparável ao algoritmo genético, mas, como esse não é o caso, vê-se que o método evolutivo foi extremamente superior.

Referências

[1] A. N. Elshafei, "Hospital layout as a quadratic assignment problem," *Journal of the Operational Research Society*, vol. 28, no. 1, pp. 167–179, 1977.