IA013 - Introdução a computação natural

Segundo relatório da lista de exercício

Sergio Luiz Novi Junior (RA-118676)
Stenio Magalhães Ranzini (RA-142372)

October 11, 2017

Exercício 2 - Solução

Nesse exercício implementaremos a solução para o problema clássico do caixeiro viajante utilizando três métodos diferentes: Algoritmo genético, Kohonen e Hopfield. Cada uma das soluções são tratadas individualmente a seguir. Infelizmente, para o caso do Hopfield, não conseguimos implementar o algoritmo sugerido no material de apoio.

Algoritmo genético

Nesta etapa do problema, resolvemos o problema do caixeiro viajante através de um algoritmo genético. Para resolver o problema, definimos uma função de *fitness* adequada, operadores de mutação, cruzamento, seleção e de busca local. Testamos o nosso algoritmo em três conjuntos de dados fornecidos pelos professores da disciplina. No primeiro conjunto, temos as posições de 52 cidades e no segundo e terceiro de 100 cidades. O *fitness* de cada indivíduo foi definido como a distância total do trajeto. Deste modo, o indivíduo mais bem adaptado é aquele que possui o menor valor de *fitness*. Cada indivíduo é representado por um vetor linha com número de colunas iguais ao número de cidades. Ou seja, para cinco cidades um indivíduo possível seria [1 2 3 4 5]. Este indivíduo representa o trajeto, sair da cidade 1 passar, nesta sequência, pelas cidades 2, 3, 4, 5 e retornar para a cidade 1.

Na Figura 1, mostramos o workflow do nosso algoritmo genético. Primeiramente, geramos um conjunto de indivíduos respeitando os vínculos do problema (passar uma vez por cada cidade e retornar a cidade de origem), em seguida realizamos mutação, reprodução, busca local e por último realizamos a seleção dos indivíduos que irão continuar para a próxima geração. Abaixo descrevemos como cada um dos operadores funciona.

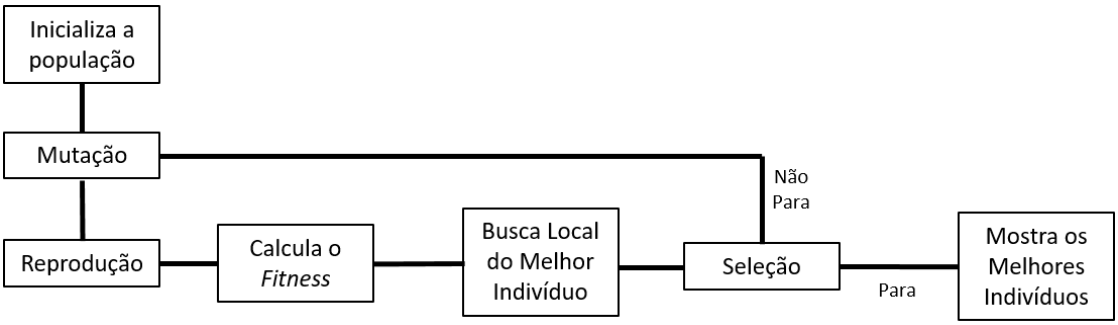


Figure 1: ***WorkFlow*** **do algoritmo genético para a solução do problema do caixeiro viajante.**

Operador de Mutação: Este operador foi aplicado nos indivíduos com uma taxa de 25%. Os indivíduos melhores adaptados são selecionados para a aplicação dos operadores de mutação. Três operadores foram utilizados: inversão, troca e desliza.

- Inversão: seleciona duas cidades de maneira aleatória e troca a sequência entre elas de maneira que parte do trajeto tem o sentido invertido quando comparado com o original. Para facilitar o entendimento, suponha o seguinte exemplo com cinco

cidades [1 2 3 4 5]. Se as cidades 2 e 4 são selecionadas, aleatoriamente, então o vetor final seria: [1 4 3 2 5].

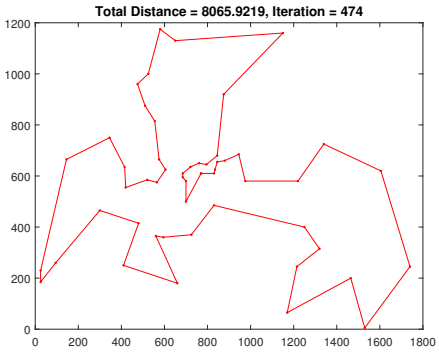
- Troca: seleciona duas cidades de maneira aleatória e troca uma pela outra.
- Desliza: seleciona duas cidades de maneira aleatória e desliza de uma posição em relação ao vetor original. Suponha o mesmo exemplo que do primeiro item, com as mesmas cidades selecionadas. Nesse caso, o vetor final seria: [1 4 2 3 5].

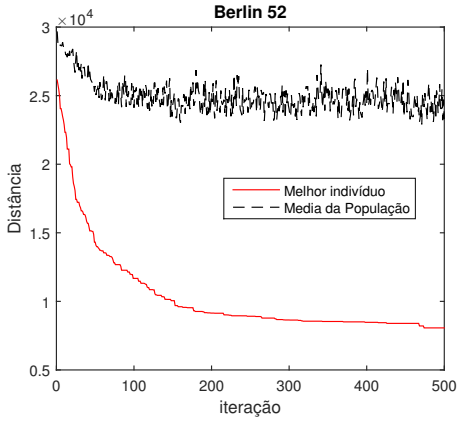
Operador de Reprodução: Duas opções foi implementada e escolha de uma dela foi feita de maneira aleatória com taxa de 50%. Todo os indivíduos sofrem uma das duas mutações. Primeiramente, os indivíduos são ordenados por ranking do melhor para o pior. Em seguida, a cada dois trajeto troca-se os números pares ou ímpares de um trajeto pelo o do outro, dependendo do sorteio da mutação selecionada. Ao final do processo teremos o dobro dos indivíduos iniciais.

Operador de Busca Local: Este operador se mostrou muito importante para encontrarmos soluções melhores depois de um alto número de iterações. A busca ocorre apenas no indivíduo mais adaptado. Escolhemos esta opção devido ao seu alto custo computacional. A busca local ocorre da seguinte forma, pega-se a melhor solução e em seguida troca-se no percurso a posição de duas *cidades vizinhas* no trajeto. Se a troca gerou um indivíduo mais bem adaptado, a mesma é mantida. E o processo continua até que todas as cidades vizinhas tenham sido trocadas uma vez. Para ilustrar este processo, suponha que tenhamos uma configuração de apenas 5 cidades com o seguinte trajeto: [1 2 3 4 5]. O que fazemos é trocar primeiramente a cidade 1 com a 2, então teríamos o percurso [2 1 3 4 5]. Se este novo percurso for melhor que o primeiro, manteremos ele e então trocaremos a cidade 1 com a 3. Teremos então, a configuração [2 3 1 4 5]. Repetimos este processo até trocarmos a última cidade com a primeira e mantemos a melhor solução.

Operação de Seleção: Este operador é o mesmo utilizado no problema 3 desta lista de exercícios. O mesmo está descrito em melhor detalhe na questão 3. De forma sucinta, utilizamos um operador de seleção bi classista. Para a nossa aplicação, escolhemos 20% dos indivíduos mais adaptados, 10% dos indivíduos menos adaptados e depois completamos o grupo com indivíduos que possuem *fitness* intermediário, aleatoriamente. Tais parâmetros se mostraram eficientes para convergência de uma boa solução bem como para a manutenção da diversidade da população. O operador de seleção também ficou incumbido de ao final de uma geração preservar o número de ínvidos iguais ao original.

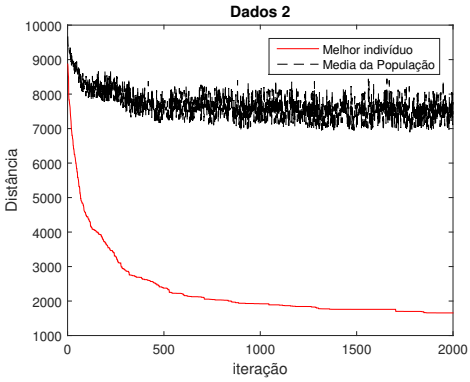
Na Figura 2 mostramos os resultados que obtivemos para a solução do caixeiro viajante através do algoritmo genético. Mostramos o trajeto final bem como o desempenho em função do número de iterações do melhor indivíduo da população e da média de todos os indivíduos. Lembramos que o nosso *fitness* é definido como a distância do percurso, portanto o melhor individuo possui o menor *fitness*. A diferença do *fitness* médio e do *fitness* do melhor sujeito sugere que o algoritmo manteve uma certa diversidade da população.





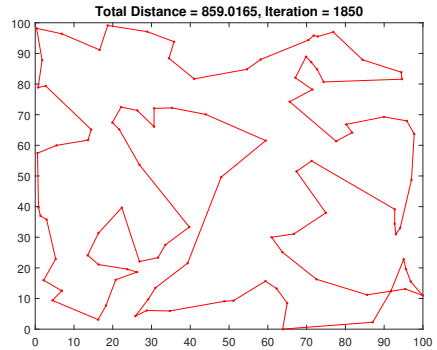
(a) Trajeto para o conjunto de dados Berlin (52 cidades).

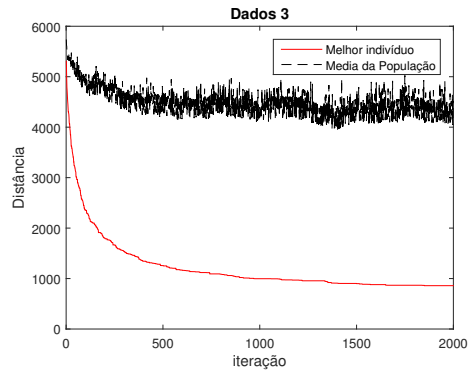
(b) Desempenho melhor e média da população para Berlin 52.



(c) Trajeto para o segundo conjunto de dados (100 cidades).

(d) Desempenho melhor e média da população para o segundo conjunto de dados.





(e) Trajeto para o terceiro conjunto de dados (100 cidades).

(f) Desempenho melhor e média da população para o terceiro conjunto de dados.

Figure 2: Trajetos e desempenho para os dados disponíveis.

Kohonen

Nesta parte do problema, tentamos otimizar os códigos disponibilizados como material de apoio para as redes de Kohonen. Na Figura 3, mostramos uma resposta encontrada utilizando todos os parâmetros padrões que já estavam configurados no material de apoio para o conjunto de dados Berlin 52. Note que a resposta encontrada já é muito boa. O trajeto sugerido aparenta não apresentar nenhum cruzamento. Esta qualidade na solução dificulta a otimização do problema.

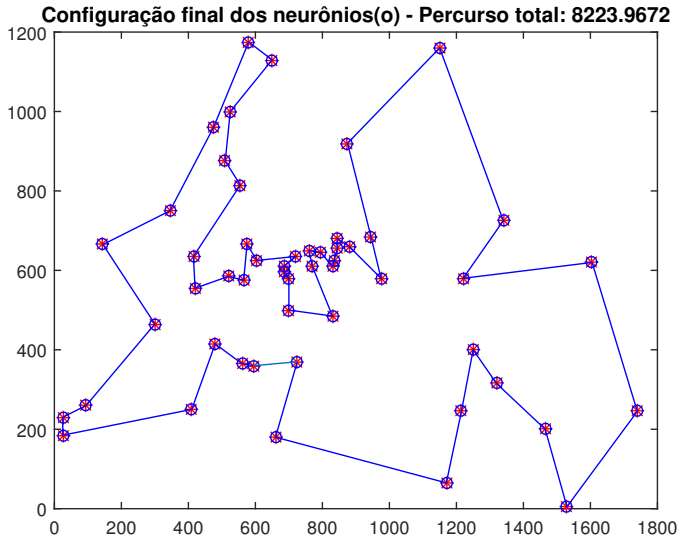


Figure 3: **Trajeto encontrado utilizando o código sugerido sem nenhuma alteração.**

Nas configurações padrões, o algoritmo funciona basicamente da seguinte forma. 10 neurônios são inicializados na forma de anel e igualmente espaçados entre si. O centro do anel de neurônios corresponde ao centro de "massa" das cidades. Para o cálculo de neurônio vencedor usa-se a distância euclidiana. Quando um neurônio ganha dos demais para uma determinada cidade, o mesmo caminha em direção a este vetor e carrega consigo a sua vizinhança. Vale ressaltar que a rede é unidimensional. O interessante do algoritmo fornecido é que a inserção e poda de neurônios ocorre apenas ao fim de um período que corresponde a 5 épocas. Nesta etapa, poda-se neurônios que nunca vencem e insere-se neurônios nas posições de outros neurônios que vencem com frequência. Os neurônios são inseridos no mesmo local que os anteriores e a vizinhança é estabelecida diretamente com o neurônio que já estava lá e um de seus adjacentes. Para um neurônio ser inserido em uma determinada localização, o neurônio que estava nesta posição, precisa atender dois critérios: ter vencido ao menos uma vez e mais da metade que o neurônio mais vencedor.

Na tentativa de otimizar este código, utilizamos a referência sugerida pelos professores que é a tese de Lalinka Gomes. Nesta tese, a autora indica que a inibição de neurônios que

vencem muito em uma mesma época se mostrou eficaz na solução do problema. Também é dito que neurônios que vencem mais de uma vez o processo competitivo em uma mesma época devem ser inibidos e deve-se inserir um novo neurônio nas coordenadas do neurônio inibido. Em seguida, a condição de vizinhança deve ser reestabelecida, o novo neurônio se move e é atribuído como vencedor. Uma outra consideração que a autora faz é sobre a inicialização dos neurônios. Ela indica que a inicialização de apenas um neurônio no centro de massa das cidades se mostrou muito eficaz.

Em nossa primeira abordagem, fizemos exatamente o que foi proposto na tese. Entretanto, obtivemos resultados piores do que o que já havíamos. Na tese, é discutido que os neurônios que vencem mais de uma vez na mesma época necessitam ser inibidos. Esta sugestão é de fato muito interessante, porém como existem uma área (configuração de Berlin 52) em que a densidade de cidades é muito alta, a inibição de um neurônio por uma época inteira acarreta na inserção de muitos neurônios. Isto acaba por aumentar o número de cruzamentos, diminuindo a qualidade da solução. Além disto, a poda de neurônios é feita ao final de três épocas. Ou seja, um neurônio é removido apenas se ele não ganhou nenhuma vez nas últimas três épocas.

Tendo em vista o comportamento da rede de Kohonen com as modificações sugeridas por Lalinka Gomes, nós optamos por manter a mesma estratégia de inibição de neurônio, porém com algumas alterações que mostraram-se eficientes. Observamos que a inibição de neurônios por uma época inteira acarretava na inserção de muito neurônio, corroborando para percursos com muitos cruzamentos. Então, decidimos manter a inibição apenas por um número de amostras da mesma época. Para ser mais preciso, a cada 25% do total de uma mesma época, neurônios que estavam inibidos passam a ser aptos e podem vencer o processo competitivo. Uma outra coisa que se mostrou eficiente foi diminuir o número de épocas para que uma poda seja efetuada. No código original o período era de 5 épocas, Lalinka Gomes sugere 3 épocas e nós observamos que um melhor balanceamento entre poda e inserção ocorre para 2 épocas. Uma outra mudança foi utilizar o número de neurônios iniciais como sendo 100 (aproximadamente duas vezes o número de cidades) e manter a inicialização na forma de anel como no código original.

Na Figura 4, mostramos o percurso do caixeiro viajante que obtemos com a configuração acima descrita e com o número de neurônios iniciais igual a 100. Note que apesar do trajeto apresentar alguns cruzamentos, o nosso percurso diminuiu de 8223.97 para 7.980.55 o que representa uma queda na trajetória de aproximadamente 3%. Para fazer uma comparação mais justa, rodamos novamente o código original, porém com 100 neurônios iniciais. A Figura 5 mostra o trajeto encontrado com uma distância de 8584.25. Ou seja, maior do que a nossa versão. Para chegarmos em uma conclusão definitiva sobre a melhoria dos ajustes, seria necessário rodar os dois algoritmos diversas vezes e ver como é o comportamento de ambos. Fazendo isto algumas vezes, observamos que o desempenho de ambos é muito similar. Entretanto, o código original em nenhum momento gerou um percurso menor do que 8 mil. Apesar da melhora de 3% parecer irrisória, o código original já estava muito otimizado o que dificulta a sua melhora.

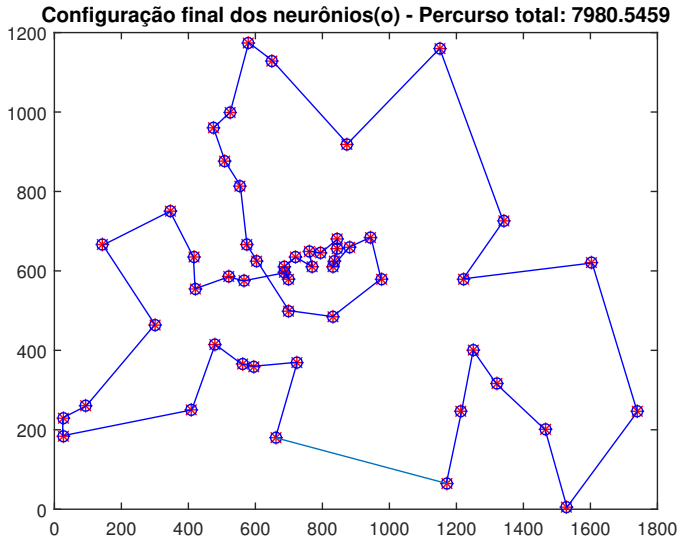


Figure 4: **Trajeto encontrado utilizando nossas modificações nas sugestões da tese de Lalinka Gomes. Observa-se uma melhora no trajeto apesar dos cruzamentos.**

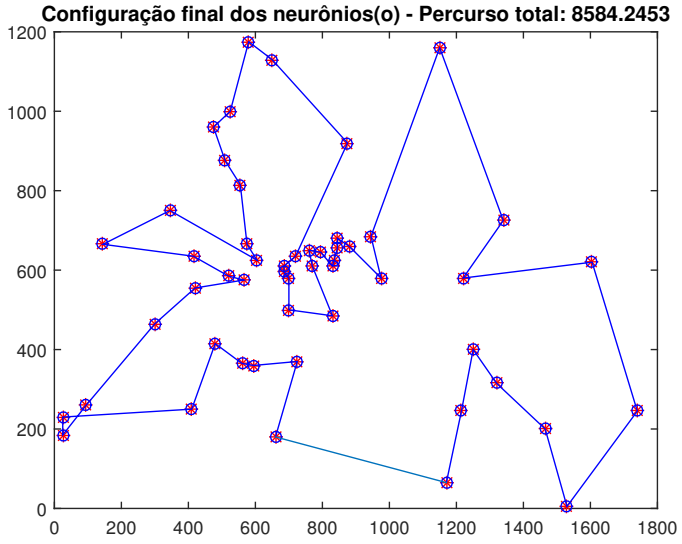


Figure 5: **Trajeto encontrado utilizando o código sugerido com uma inicialização de 100 neurônios.**

Exercício 3 - Solução

Nesta questão resolveremos o problema de atribuição didática. Neste problema, cada professor indica três matérias que gostaria de lecionar no próximo semestre letivo. A

escolha das matérias segue uma ordem de preferência. Ou seja, se algum professor indicar as matérias 1 – Mecânica Quântica, 2 – Física Estatística e 3 – Eletromagnetismo, isto significa que o mesmo prefere primeiramente lecionar quântica, depois física estatística e por último eletromagnetismo. Apesar deste problema parecer simples, o mesmo é de grande importância no contexto acadêmico.

Para modelar este problema assumiremos que existem um mesmo número de disciplinas que professores e que cada professor ficará responsável por apenas uma delas e que todas as disciplinas devem ser oferecidas. Então, o vínculo é da seguinte forma: Para cada disciplina precisamos ter exatamente um único professor. Cada indivíduo será representado por uma matriz A quadrada com dimensões do número de disciplinas pelo número de professores. Esta assume valores 0 ou 1. A codificação é feita de tal forma: se o elemento da matriz a_{ij} for igual 1, isto significa que a disciplina i será lecionada pelo professor j. A inicialização da população se dá de forma aleatória e respeita os vínculos do problema. Como não possuímos dados reais, as preferências de cada professor é feita de forma aleatória e respeita o vínculo de que cada professor precisa indicar três disciplinas e elas devem ser distintas e ranqueadas.

Resolveremos este problema de tal forma que o resultado final favoreça todo o grupo de professores e não apenas algum professor em específico. Para tanto, definimos a função fitness da seguinte forma: se o professor ficou com a primeira opção ele recebe 10 pontos, se ele ficou com a segunda opção ele recebe 6 pontos, se ele ficou com a terceira opção ele recebe 3 pontos e se ele ficou com nenhuma opção sugerida ele recebe 0 pontos. Ao final, extraímos o fitness do indivíduo como a soma dos pontos de cada professor, lembrando que um indivíduo engloba a atribuição de todos os professores.

A Figura 6 mostra o *workflow* do nosso algoritmo genético e em seguida explicaremos como cada operador funciona (mutação, reprodução e seleção). O critério de parada utilizado foi o número de iterações.

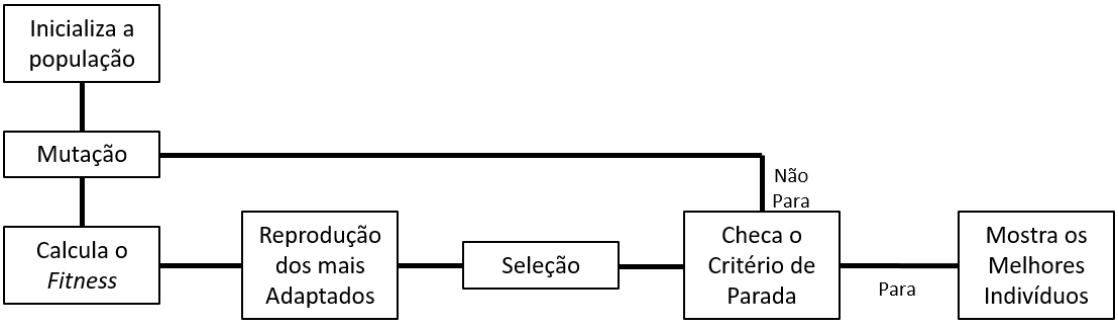


Figure 6: ***WorkFlow* do algoritmo genético para a solução do problema de atribuição didática.**

Operador de mutação: Cada indivíduo da população pode sofrer dez tipos diferentes de mutação com base em uma probabilidade (ou taxa) que precisa ser definida pelo usuário. Em nossas simulações, a taxa de 0,2 mostrou-se eficiente em evoluir o *fitness* da população e também para manter um certo grau de diversidade. A mutação ocorre através da troca

de colunas de cada indivíduo. Escolhemos variar o número de trocas entre duas e dez vezes. Cada troca ocorre de forma independente com uma taxa preestabelecida e gera um indivíduo. Por exemplo, suponha que iremos trocar apenas um par de colunas. Então, geramos de forma aleatório dois números distintos que correspondem a coluna que será trocada. Em seguida, realiza-se esta troca. Por outro lado, se resolvermos trocar 10 colunas, geramos de forma aleatório 10 pares de colunas que devem ser trocadas entre si. Para clarificar, para cada indivíduo escolhemos um número aleatório, se este número for menor do que 0.2, o indivíduo sofrerá 9 mutações levando a criação de outros 9 indivíduos. Caso existam menos de 10 disciplinas e professores, efetuamos apenas uma mutação.

Operador de Reprodução: Cada cruzamento ocorre entre dois indivíduos e o operador de reprodução atua apenas nos melhores indivíduos do grupo. Uma vez que os melhores indivíduos são selecionados, damos a chance de na média todos os indivíduos efetuarem dois cruzamentos. Ou seja, se temos ao todo 10 indivíduos para fazer a reprodução, faremos ao todo dez cruzamentos. O cruzamento ocorre da seguinte formas: escolhemos aleatoriamente dois indivíduos distintos e em seguida escolhemos uma coluna que será trocada entre eles. A troca ocorre apenas do segundo indivíduo para o primeiro. Deste modo, cada reprodução gera apenas um novo indivíduo. Neste processo, tomamos o cuidado de não termos colunas repetidas para isso é necessário rearranjar as colunas que são trocadas. Ou seja, antes de fazermos a troca verificamos como é a coluna nova que será inserida. Depois disto, trocamos a coluna do sujeito que receberá a nova coluna na posição da coluna que o sujeito tem e que contém a nova coluna. Deste modo, garantimos que os vínculos do problema são respeitados.

Operação de Seleção: Esta etapa é crucial para o algoritmo e necessita de atenção para que não ocorra uma perde de diversidade prematura. Sendo assim, optamos por utilizar o método de seleção bi classista. Neste método são escolhidos b% dos indivíduos mais adaptados, w% dos indivíduos menos adaptados e o restante é completado com indivíduos intermediários, aleatoriamente. Em nosso trabalho, optamos por manter o número de indivíduos fixo e igual ao inicial. Ou seja, o operador de seleção separa os indivíduos e os limita ao número de indivíduos inicias. Por exemplo, suponha que tenhamos como parâmetros b=10%, w=10% e o número de indivíduos iniciais igual a 10. Depois da mutação e da reprodução teremos mais de 10 indivíduos. Então, o nosso operador de seleção com os parâmetros selecionados irá separar o melhor indivíduo, o pior indivíduo e oito indivíduos intermediários. Em nossa aplicação escolhemos os parâmetros b = 50% e w=30%. Tais parâmetros se mostraram eficientes na melhoria do *fitness* da população bem como na manutenção da diversidade.

Na Figura 7, mostramos uma aplicação do nosso algoritmo utilizando uma população inicial de apenas 10 indivíduos e com o número de docentes e disciplinas igual a 100. As preferências de cada professor foram escolhidas ao acaso. Note que o *fitness* do indivíduo mais bem adaptado (linha sólida vermelha) cresce rapidamente e em menos de 1000 iterações atinge valor acima de 500. Apesar de não sabermos qual seria a resposta ótima, sabemos que caso todos os professores tenham escolhido uma disciplina preferencial diferente dos demais, a melhor configuração renderia um *fitness* igual a 1000. Note também a distância entre o *fitness* médio da população (linha preta pontilhada) e o *fitness* do melhor

indivíduo, esta discrepância sugere que a diversidade é mantida na populações.

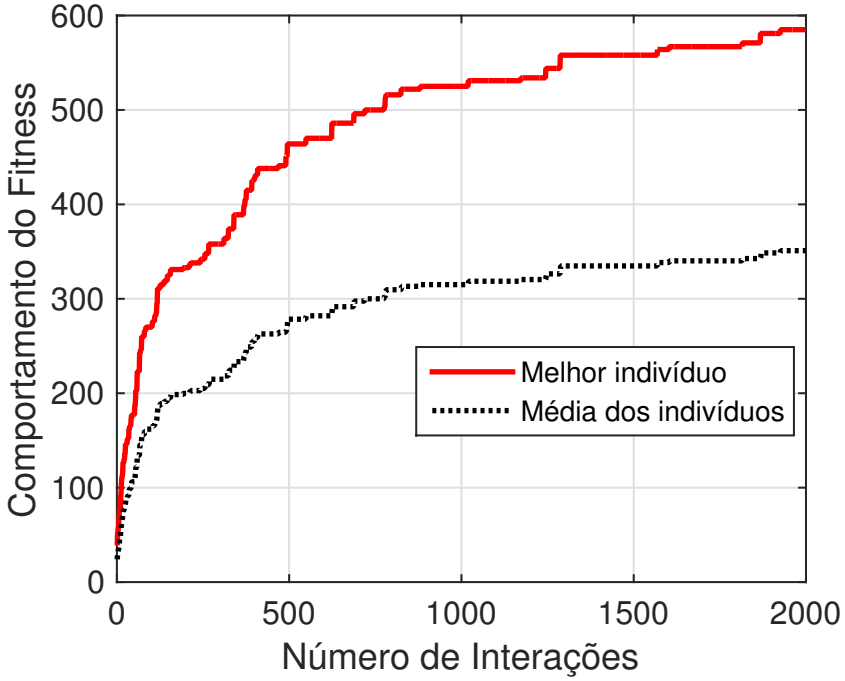


Figure 7: **Comportamento do fitness do melhor indivíduo (linha sólida vermelha) da população e do fitness médio da população (linha pontilhada preta).**

Para verificar o poder de convergência do nosso algoritmo, definimos o número de disciplinas e de professores iguais a 50 e configuramos as preferências de cada professor de tal modo que cada um indicou uma disciplina como primeira opção. Tendo em vista o cálculo do nosso *fitness*, a melhor solução seria alcançada com cada professor lecionando a sua primeira opção. Com uma população de apenas 10 indivíduos, ao final de 4000 iterações, apenas 3 professores não terminaram a simulação com as suas primeiras escolhas.

Exercício 4 - Solução

Nesse exercício o objetivo é aplicar os conceitos de algoritmo genético em um exercício de ponto flutuante. Para isso, escolhemos minimizar a função Rastrigin, cuja a descrição é:

$$z(x,y) = 10 * 2 + x^2 + y^2 - 10 * (cos(2 * pi * x) + cos(2 * pi * y)); \quad (1)$$

onde: x e y variam entre -5,12 e 5,12.

A Figura 8 mostra um gráfico em três dimensões para essa função.

A Figura 9 mostra o *workflow* do nosso algoritmo genético. Cada operador é tratado individualmente a seguir. O critério de parada utilizado foi o número de iterações.

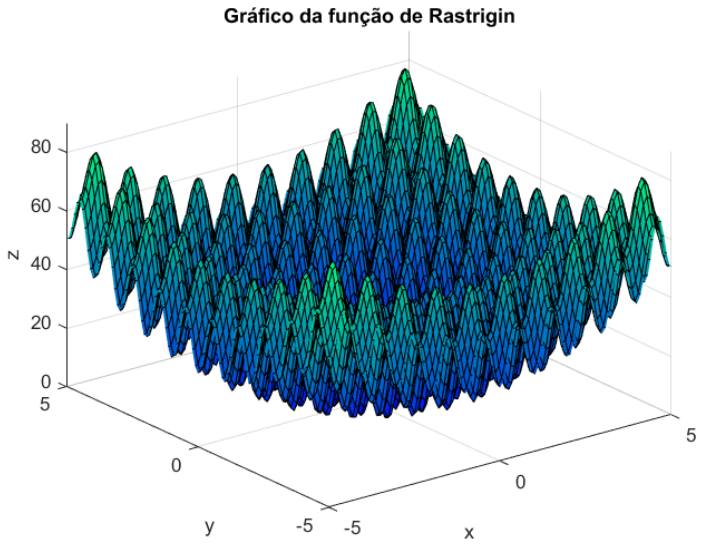


Figure 8: **Representação em três dimensões da função Rastrigin com os vetores x e y variando de -5,12 à 5,12.**

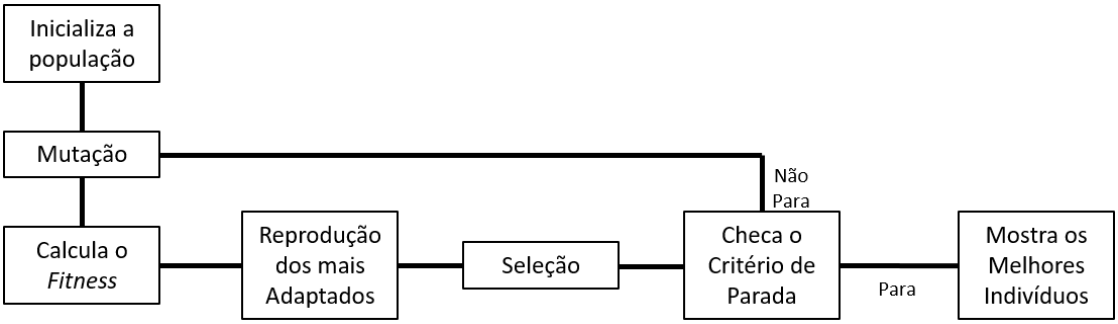


Figure 9: ***WorkFlow* do algoritmo genético para a solução do problema com variável em ponto flutuante.**

Operador de mutação: Após a criação dos indivíduos de maneira aleatória, cada um deles possue uma chance de 20% em sofrer mutação. Quando o indivíduo é selecionado, este sofre nove mutações que são acrescentadas a população inicial. As mutações acontecem da seguinte maneira:

- Troca-se x por y;
- Troca-se x por y e inverte o sinal de ambos;
- Acrescenta-se um delta no valor de 0,1 para o vetor x e y;
- Troca-se x por y e acrescenta-se o mesmo valor de delta nos dois vetores;

- Acrescenta-se o mesmo delta, nos dois vetores, mas troca o sinal apenas de y;
- Acrescenta-se três vezes o valor do delta, para ambos os vetores;
- Acrescenta-se três vezes o valor do delta, para ambos os vetores, e inverte o sinal apenas de y;
- Acrescenta-se três vezes o valor do delta, para ambos os vetores, e troca-se x por y;
- Acrescenta-se sete vezes o valor do delta, para ambos os vetores.

Operador de Reprodução: Esse operador é aplicado em todos os indivíduos para a geração da próxima geração. Foi utilizado a média aritmética como forma de cruzamento entre indivíduos. Os indivíduos são selecionados de maneira aleatória entre toda a população.

Operação de Seleção: A seleção utilizada foi a bi classista. Neste método são escolhidos b% dos indivíduos mais adaptados, w% dos indivíduos menos adaptados e o restante é completado com indivíduos intermediários, aleatoriamente. Em nosso trabalho, optamos por manter o número de indivíduos fixo e igual ao inicial.

A Figura 10 mostra o comportamento do algoritmo genético para 50 iterações. A Figura é a projeção do gráfico de três dimensões mostrado na figura 8. Os pontos vermelhos representam a população de indivíduos no função de Rastrigin. Apenas as iterações 1, 13, 25 e 47 são mostradas. Na figura não aparece, mas mesmo após o algoritmo ter encontrado um mínimo local, por causa das mutações ele é capaz de olhar a vizinhança e trocar de mínimo caso este tenha um *fitness* melhor. Foi enviado junto com o documento um gif completo com 50 iterações que mostra o comportamento do algoritmo.

A Figura 11 mostra a função de avaliação ao passar as iterações. Nota-se que a diferença do melhor indivíduo para o pior muda ao longo do tempo. Quando a função encontra um mínimo a diferença entre as duas populações são mínimas. Entretanto, a mutação logo começa a agir e a diversidade da população começa a aumentar novamente.

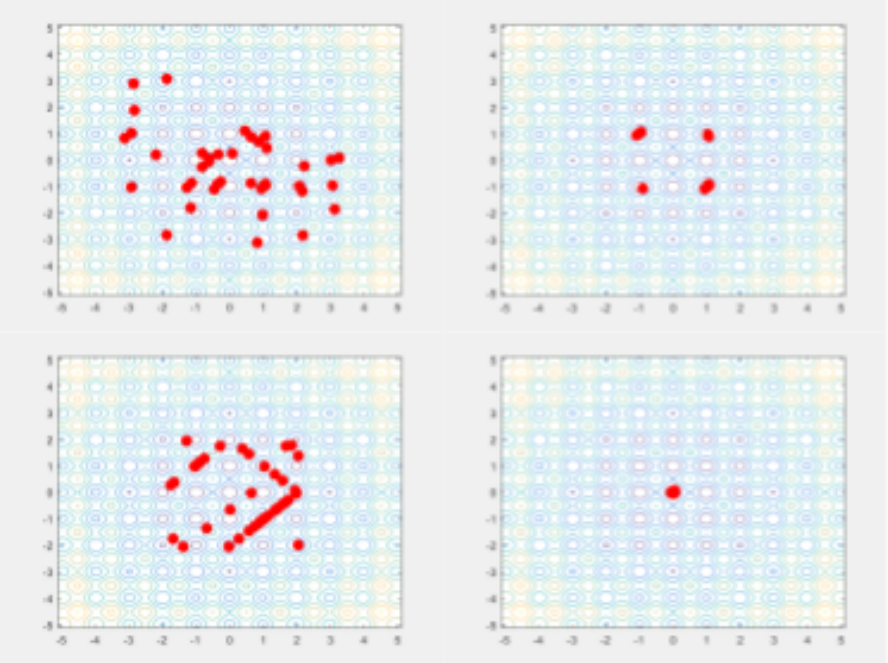


Figure 10: População de indivíduos na projeção da função Rastrigin ao longo das iterações. O gráfico superior esquerdo é a primeira iteração. O gráfico superior direito é a décima terceira iteração. O gráfico inferior esquerdo é a vigésima quinta iteração. O gráfico inferior direito é a quadragésima sétima iteração.

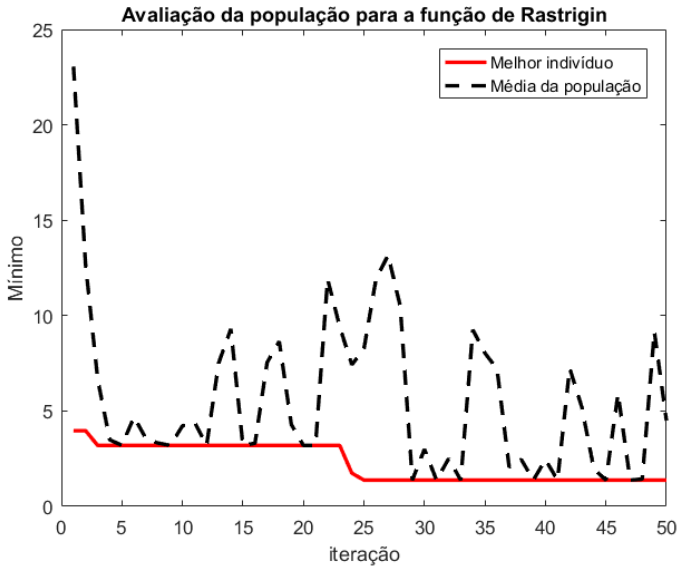


Figure 11: Avaliação do melhor indivíduo da população e do melhor para a função de Ratrigin ao longo das iterações.