Universidade Federal de Ouro Preto

Computação Evolucionária

TSP

Autor:
Daniel Martins Reis

Matrícula:
14.1.8295

1 Algoritmo Evolutivo

O presente trabalho consiste na implementação de uma versão híbrida de algum algoritmo evolutivo. Foi elaborado uma algoritmo que mescla parte do Algoritmo Genético, também conhecido como AG, e do Algoritmo de Evolução Estratégica. A Figura 1 representa o pseudocódigo utilizado no algoritmo criado:

```
Cria e avalia pop inicial
Calcula numr (numero de melhores indivíduos selecionados para novaPop)
Para cada geração
    Move os numr (pop -> novaPop)
    Gera os (tamPop - numr) -> Cria descendentes com base nos x melhores
        Se rnd [0,1] <= taxa de crossover
            Seleciona dois pais aleatórios na novaPop
            Cria descendente1 e descendente2 usando crossoverOX combinando pai1 com pai2
            Mutação SWAP no descendente1
            Avalia descendente1 e descendente2
            Se rnd [0,1] <= taxa de busca local
                Busca local no descendente1 (troca u por v)
            Se rnd [0,1] <= taxa de busca local
                Busca local no descendente2 (troca u por v)
            Adiciona descendente1 e descendente2 na novaPop
        Se rnd [0,1] >= taxa de mutacao
            Cria descendente3 usando um pai aleatório na pop
            Mutação SWAP no descendente3
            Avalia descendente3
            Se rnd [0,1] <= taxa de busca local
                Busca local no descendente3 (troca u por v)
            Adiciona descendente1 e descendente2 na novaPop
    Define sobreviventes:
        Insere novaPop na pop
        Ordena pop pela FO
        Corta pop em tamPop
```

Figura 1: Pseudocódigo.

Basicamente, o que se faz é mover um percentual de indivíduos melhores para a nova população. Daí, tentar gerar descendentes de dois tipos:

- Primeiro: é selecionado dois pais aleatórios na nova população e realizada uma operação de *crossover* entre eles. Logo após, uma mutação SWAP é realizada.
- Segundo: é selecionado um pai aleatório na população e realizada uma operação de mutação SWAP nele.

Portanto, o melhor entre os dois indivíduos gerados é adicionado a nova população.

A Figura 2 mostra o pseudocódigo utilizado para o Algoritmo Genético.

```
Cria e avalia pop inicial
Para cada geração
    Limpa a novaPop
    Gera os tamPop
        Se rnd [0,1] <= taxa de crossover
            Seleciona dois pais aleatórios na pop
            Cria descendente1 e descendente2 usando crossoverOX combinando pai1 com pai2
            Avalia descendente1 e descendente2
            Mutação SWAP no descendente1 e descendente2
            Avalia descendente1 e descendente2
            Se rnd [0,1] <= taxa de busca local
                Busca local no descendente1 (troca u por v)
            Se rnd [0,1] <= taxa de busca local
                Busca local no descendente2 (troca u por v)
            Adiciona descendente1 e descendente2 na novapop
    Define sobreviventes:
        Insere novaPop na pop
        Ordena pop pela FO
        Corta pop em tamPop
```

Figura 2: Pseudocódigo.

2 Função de Avaliação

A função de avaliação do algoritmo TSP que representa um problema do caixeiro viajante (*Traveling Salesman Problem - TSP*) é expressada pela função:

$$d_{A,B} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

Figura 3: Função de avaliação.

O problema em questão, representa a distância entre os vértices. Dessa maneira, a função de avaliação é definida como o menor percurso entre um ou mais vértices - distância euclidiana, ou popularmente falando, distância entre dois pontos.

3 Testes de execução

Parâmetros utilizados:

- numexecucoes = 30 [Número de execuções];
- geracoes = 100 [Número de gerações];
- tamPop = 1000 [Tamanho da população];
- pSelecionados = 0.05 [Percentual de selecionados da população];
- pCrossover = 0.7 [Percentual de *crossover*];
- pMutacao = 0.05 [Percentual de mutação];
- pBuscaLocal = 0.75 [Percentual de busca local].

3.1 Comparativo Novo Algoritmo x AG -> Berlin 52

Boxplots:

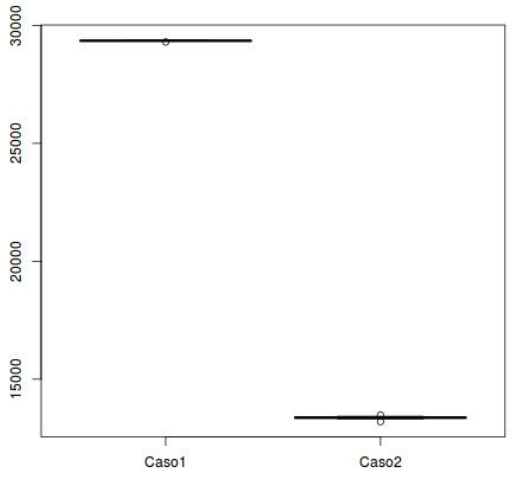

Figura 4: Média usando a instância Berlin 52.

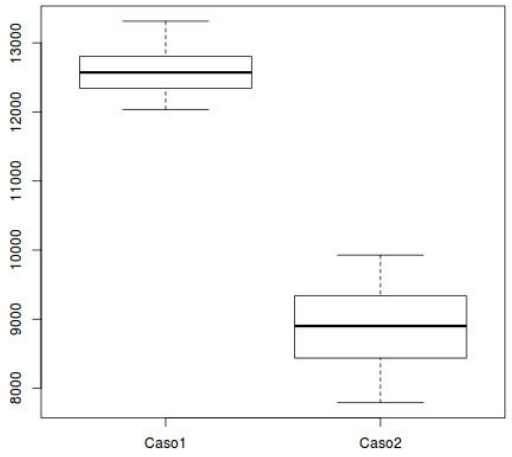

Figura 5: Melhor Resultado usando a instância Berlin 52.

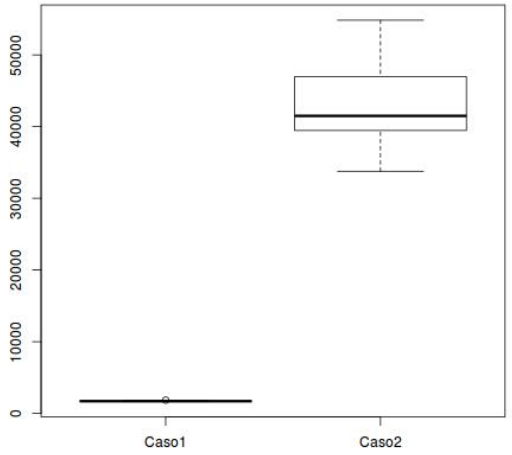

Figura 6: Tempo usando a instância Berlin 52.

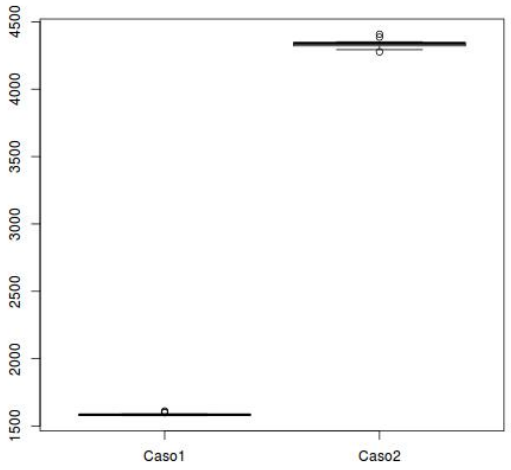

Figura 7: Desvio Padrão usando a instância Berlin 52.

3.2 Comparativo Novo Algoritmo x AG -> Att 48

Boxplots:

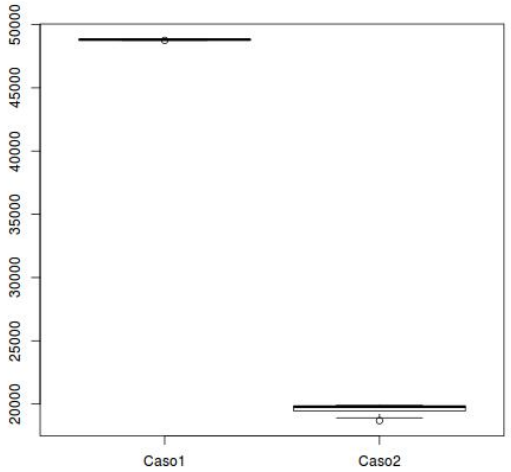

Figura 8: Média usando a instância Att 48.

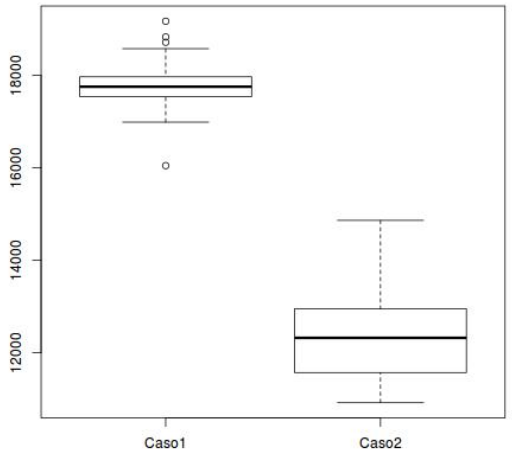

Figura 9: Melhor Resultado usando a instância Att 48.

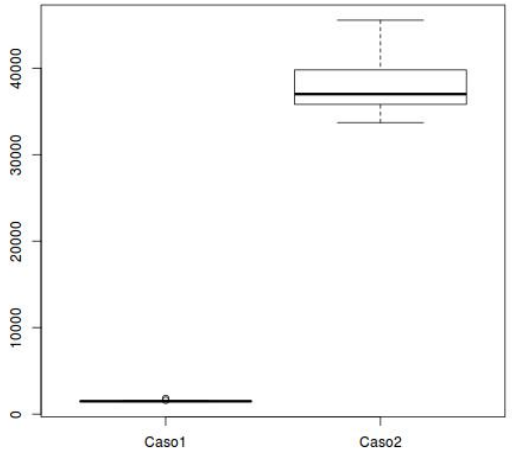

Figura 10: Tempo usando a instância Att 48.

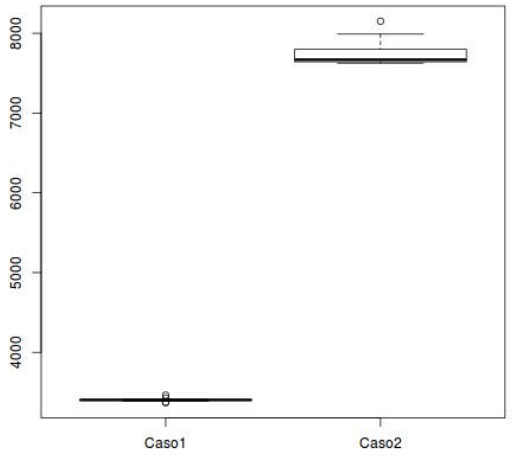

Figura 11: Desvio Padrão usando a instância Att 48.

3.3 Teste T - Instância Berlin 52

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, ambos sob a instância Berlin 52, temos:

Tipo	pvalor	pvalorL	pvalorG
Melhor ∼ Caso	2.80927319269287e-32	1	1.40463659634644e-32
Media ∼ Caso	6.48208737002271e-86	1	3.24104368501136e-86
DP ∼ Caso	1.19090465416907e-71	5.95452327084537e-72	1
Tempo ∼ Caso	2.91821624834869e-28	1.45910812417435e-28	1

3.4 Teste T - Instância Att 48

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, ambos sob a instância Att 48, temos:

Tipo	pvalor	pvalorL	pvalorG
Melhor ∼ Caso	2.03889502566081e-28	1	1.0194475128304e-28
Media ∼ Caso	3.16471474048073e-60	1	1.58235737024036e-60
DP ∼ Caso	4.75067441788163e-48	2.37533720894082e-48	1
Tempo ∼ Caso	6.41376583146505e-33	3.20688291573252e-33	1

3.5 Resultados - Instância Berlin 52

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, ambos sob a instância Berlin 52, temos:

Caso	Melhor	DP	Média
1	12036.2264517108	329.651926583879	12588.5014243297
2	7790.49461584654	571.940478501201	8921.91627832543

3.6 Resultados - Instância Att 48

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, ambos sob a instância Att 48, temos:

Caso	Melhor	DP	Média
1	16050	606.783815580619	17749.2333333333
2	10922	1025.70663559268	12426.0333333333

3.7 Conclusão

Estatisticamente falando, com base no *p-valor* do Teste T, o Algoritmo criado apresenta um melhor resultado e melhor média que o Algoritmo Genético, mas em compensação executa por uma faixa maior de tempo e tem um desvio padrão um pouco maior. Tais análises podem ser visualizadas nos *Boxplots*. Analisando o melhor resultado obtido pelo algoritmo criado, pode-se dizer que este chega bem próximo do mínimo local. Nas duas instâncias executadas, o Algoritmo criado tem um melhor caso bem próximo do mínimo local, o que não ocorre com o AG utilizando os parâmetros descritos. Nota-se que a média nas 30 execuções na instância Berlin 52 utilizando o algoritmo criado, está em 8921,91627832543 e no AG 12588,5014243297 numa situação em que o melhor resultado é 7542, e na média das 30 execuções na instância Att 48 utilizando o algoritmo criado, está em 12426,0333333333 e no AG 17749,2333333333 numa situação em que o melhor resultado é 10628. Observa-se um comportamento muito bom do algoritmo criado, obtendo um resultado melhor que o AG.

Vale ressaltar, que para execução destes testes utilizou-se os melhores parâmetros encontrados sem alterar o número de gerações e tamanho da população definidos, foram:

- pSelecionados = 0.05 [Percentual de selecionados da população];
- pCrossover = 0.7 [Percentual de *crossover*];
- pMutacao = 0.05 [Percentual de mutação];
- pBuscaLocal = 0.75 [Percentual de busca local].

Nota-se que a medida que se aumenta o percentual de indivíduos selecionados, ou seja, os indivíduos que serão utilizados como pais para gerar a nova população, o melhor resultado e média tendem a cair, sendo 5% o percentual em que se obtem uma melhor função objetivo neste problema. O mesmo efeito ocorre para o *crossover* em que abaixo de 70% ou acima disso os resultados pioram, e com a mutação com valores acima ou abaixo de 5%.