Universidade Federal de Ouro Preto

Computação Evolucionária

Prova

Autor:
Daniel Martins Reis

Matrícula:
14.1.8295

1 Algoritmo Evolutivo

A presente prova 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 usando crossover de 1 ponto combinando pai1 com pai2
            Mutação por bit no descendente1
            Avalia descendente1
        Se rnd [0,1] >= taxa de mutacao
            Cria descendente2 usando um pai aleatório na pop
            Mutação por bit no descendente2
            Avalia descendente2
        Adiciona o descendente com melhor FO ou o único gerado
    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 por bit é realizada.
- Segundo: é selecionado um pai aleatório na população e realizada uma operação de mutação por bit 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.

2 Função de Avaliação

Para validação do algoritmo, foi solicitado a utilização da função de avaliação proposta por *Schwefel's*, que deve ser minimizada. A Figura 3 representa tal função:

```
Cria e avalia pop inicial
Calcula o número de pais (% elitismo)
Para cada geração
    Move os x% melhores indivíduos (pop -> elite)
    Gera os (tamPop - numPais)
        Se rnd [0,1] <= taxa de crossover
            Seleciona dois pais aleatórios na pop
            Cria descendente usando crossover de 1 ponto combinando pai1 com pai2
            Mutação por bit no descendente
            Avalia descendente
            Adiciona descendente na novapop
    Define sobreviventes:
        Limpa pop
        Ordena novaPop pela FO
        Corta novaPop em [tamPop - numPais]
        Insere na pop [elite + novaPop]
```

Figura 2: Pseudocódigo.

$$f(\mathbf{x}) = -\sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|})$$

Figura 3: Função de avaliação.

As variáveis são definidas no intervalo -500 $\leq$ Xi $\leq$ 500. Além disso, o mínimo global é igual a f(x) = -n * 418,9829; o qual é obtido quando xi = 420,9687; com i = 1 : n. Para n = 2, essa função é representada pela Figura 4. Referências e outras informações podem ser obtidas em: <http://goolgl/v1OVB>.

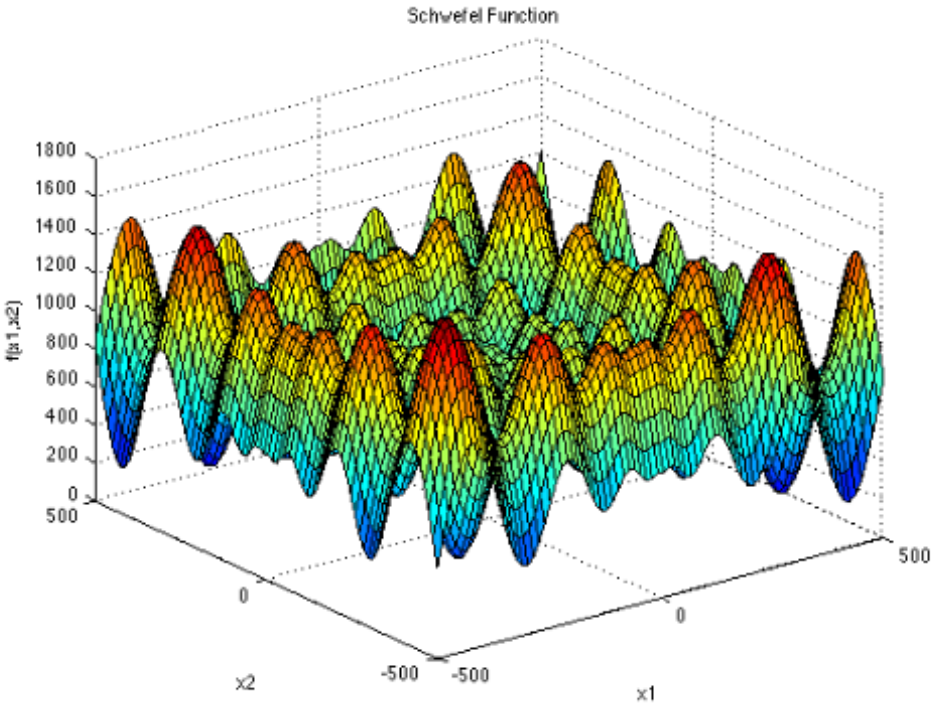

Figura 4: Função de *Schwefel's*.

3 Testes de execução

Parâmetros utilizados:

- numexecucoes = 30 [Número de execuções];
- min = -500.0 [Mínimo no intervalo da função];
- max = 500.0 [Maxímo no intervalo da função];
- nvar = 40 [Número de variáveis];
- geracoes = 300 [Número de gerações];
- tamPop = 100 [Tamanho da população];
- pSelecionados = 0.15 [Percentual de selecionados da população];
- pCrossover = 0.7 [Percentual de *crossover*];
- pMutacao = 0.05 [Percentual de mutação];

3.1 Comparativo Novo Algoritmo x Algoritmo Genético

Boxplots:

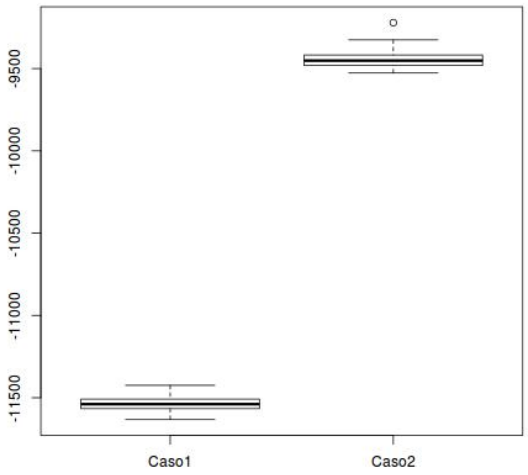

Figura 5: Média.

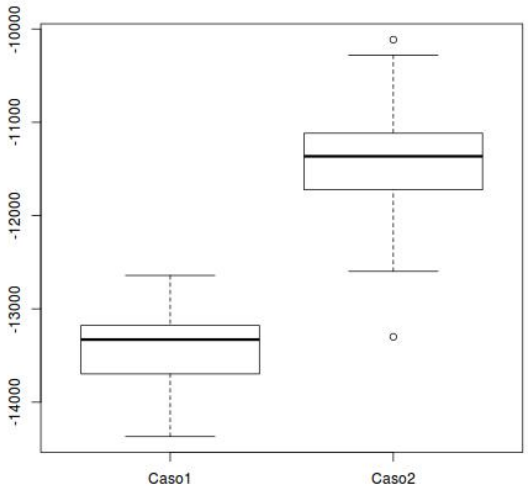

Figura 6: Melhor Resultado.

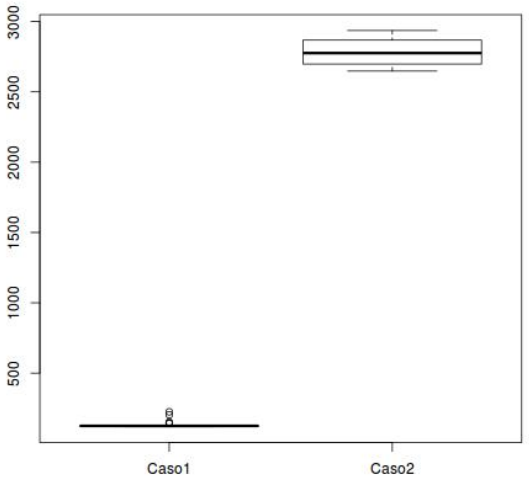

Figura 7: Tempo.

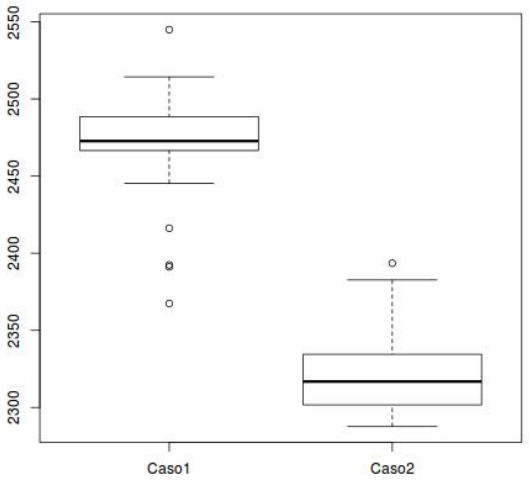

Figura 8: Desvio Padrão.

3.2 Teste T

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, temos:

Tipo	pvalor	pvalorL	pvalorG
Melhor $\sim$ Caso	8.78730397453842e-19	4.39365198726921e-19	1
Pior $\sim$ Caso	0.596957383551634	0.701521308224183	0.298478691775817
Media $\sim$ Caso	1.49981105682554e-68	7.4990552841277e-69	1
DesvioPadrao $\sim$ Caso	6.78735982161086e-24	1	3.39367991080543e-24
Tempo $\sim$ Caso	2.53130209627551e-49	1.26565104813775e-49	1

3.3 Resultados

Comparando 1 com 2, em que 1 é uma execução do AG e 2 do algoritmo criado, temos:

Caso	Melhor	DP	Média
1	-14368.5098094552	465.013735111565	-13452.2907265631
2	-13299.6223015673	668.031544866919	-11424.008418257

3.4 Conclusão

Estatisticamente falando, com base no *p-valor* do Teste T, o AG apresenta um melhor resultado, média e executa em menor tempo que o algoritmo criado, mas levando em consideração o desvio padrão, nota-se que no algoritmo criado este é menor, ou seja, os resultados obtidos nas execuções variam pouco. 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, mas não tanto como o AG (-14368.5098094552 x -13299.6223015673). Nota-se que a média nas 30 execuções do algoritmo criado está em -11424.008418257, o que não é tão ruim perante ao AG (-13452.2907265631).

Vale ressaltar, que para execução destes testes, utilizou-se 0% de elitismo para o AG, e os melhores parâmetros encontrados sem alterar o número de gerações e tamanho da população definidos, foram:

- pSelecionados = 0.15 [Percentual de selecionados da população];
- pCrossover = 0.7 [Percentual de *crossover*];
- pMutacao = 0.05 [Percentual de mutação];

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 15% 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%. Além disso, normalmente o AG demora em torno de 25 gerações para sair de aproximadamente -3.000 e chegar a -10.000. Já o algoritmo criado, necessita de cerca de 60 gerações para chegar ao mesmo resultados, o que faz com que se executado com poucas gerações, é bem pior se comparado ao AG.