UNIFAL-MG
AEDS 3

ISABELLE SGRIGNERO - 2019.1.08.030
JASMINE GERMANO FRANÇA - 2020.1.08.023
JOSÉ FLÁVIO LOPES - 2019.1.08.045

TRABALHO PRÁTICO 3

RELATÓRIO

ALFENAS
2022

Descrição do Problema

Dado um grafo completo, encontrar o maior subgrafo induzido com 5 vértices.

Abordagem Utilizada

Para elaborar uma solução para o problema, utilizamos, primeiramente, uma heurística construtiva que gera uma população de soluções de maneira aleatória. A partir da população, fizemos uso de algumas ideias de Algoritmos Evolutivos, elaborando, assim, uma função de seleção, uma de combinação e uma de mutação. A função de seleção sorteia três soluções que pertencem a população e seleciona a com maior peso. A função de combinação recebe duas soluções que foram selecionadas pela função de seleção e gera uma nova solução a partir delas. Por fim, a função de mutação troca, aleatoriamente, dois elementos da solução gerada pela função de combinação.

Pseudo-código dos Algoritmos Utilizados

Heurística Construtiva para gerar população

```
geraPopulacao():
    i = 0
    enquanto i < capacidadePopulacao faca
        j = 0
        enquanto j < 5 faca
            v = aleatorio()
            enquanto v contido em solucoes[i] faca
                v = aleatorio()
            solucoes[i][j] = v
            j = j + 1
        i = i + 1
```

Função de Seleção

seleciona():

```
s1 = aleatorio() // índice da solução no array de soluções
s2 = aleatorio()
s3 = aleatorio()

se peso(s1) > peso(s2) & peso(s1) > peso(s2):
        retorna s1
senao se peso(s2) > peso(s1) & peso(s2) > peso(s1):
        retorna s2
senao
        retorna s3
```

Função de Combinação

combina(s1, s2**):**

```
i = 0
idx = menor() // índice da solução de menor custo

enquanto i < 5 faca:
        se par(i):
                v = s1[i]
        senao:
                v = s2[i]
        enquanto v contido em solucoes[idx] faca:
                v = aleatorio()
        solucoes[idx][i] = v
retorna idx
```

Função de Mutação

```
muta(idx):
    x = aleatorio() // índice do elemento a ser substituído
    y = aleatorio() // índice do elemento a ser substituído

    nx = aleatorio() // elemento a ser colocado em x

    enquanto nx contido em solucoes[idx] faca:
        nx = aleatorio()

    ny = aleatorio() // elemento a ser colocado em y

    enquanto ny contido em solucoes[idx] faca:
        ny = aleatorio()

    solucoes[idx][x] = nx
    solucoes[idx][y] = ny
```

Resultado para Cada Instância

- **a28.txt**

  10 segundos:

  0 77 95 241 1

  1184.09

  30 segundos:

  0 77 95 241 1

  1184.09

  60 segundos:

  0 77 95 241 1

  1184.09

- **att48.txt**
    - 10 segundos:
        - 16 3 34 18 36
        - 33123.17
    - 30 segundos:
        - 16 3 34 18 36
        - 33123.17
    - 60 segundos:
        - 16 3 34 18 36
        - 33123.17

- **att532.txt**
    - 10 segundos:
        - 488 0 1 471 464
        - 34989.03
    - 30 segundos:
        - 488 0 1 471 464
        - 34989.03
    - 60 segundos:
        - 488 0 1 471 464
        - 34989.03

- **berlin52.txt**
    - 10 segundos:
        - 1 51 10 6 16
        - 6719.27
    - 30 segundos:
        - 1 51 10 6 16
        - 6719.27
    - 60 segundos:
        - 1 51 10 6 16
        - 6719.27

- **bier127.txt**
    - 10 segundos:
        - 98 97 100 91 103
        - 74186.69
    - 30 segundos:
        - 98 97 100 91 103
        - 74186.69
    - 60 segundos:
        - 98 97 100 91 103
        - 74186.69

- **brd14051.txt**
    - 10 segundos:
        - 13864 5435 5226 13871 13861
        - 35482.17
    - 30 segundos:
        - 13864 5435 5226 13871 13861
        - 35482.17
    - 60 segundos:
        - 13864 5435 5226 13871 13861
        - 35482.17

- **burma14.txt**
    - 10 segundos:
        - 9 4 3 1 0
        - 39.02
    - 30 segundos:
        - 9 4 3 1 0
        - 39.02
    - 60 segundos:
        - 9 4 3 1 0
        - 39.02

- **d15112.txt**
    10 segundos:
        4487 10575 7953 14109 11008
        99869.62
    30 segundos:
        4487 10575 7953 14109 11008
        99869.62
    60 segundos:
        4487 10575 7953 14109 11008
        99869.62

- **eil101.txt**
    10 segundos:
        37 64 34 85 42
        341.66
    30 segundos:
        37 64 34 42 85
        341.66
    60 segundos:
        37 64 34 42 85
        341.66

Conclusão

Na nossa visão, esse trabalho foi o mais interessante de se fazer, principalmente pelo fato de não conseguirmos atingir uma solução ótima, fazendo com que possamos utilizar da criatividade para melhorar a solução que nosso programa gera. Acerca da implementação, nosso código gerava soluções diferentes para cada tamanho de população utilizado; optamos por utilizar uma população de cinquenta indivíduos que, embora não produzisse variação nas três medidas de tempos pedidas, serviu como uma ótima base para as soluções produzidas, atingindo, para algumas instâncias, resultados que populações com mais de mil indivíduos não produziam. No geral, foi um trabalho recompensador e serviu para despertar em nós a curiosidade sobre a área de Algoritmos Evolutivos.