Relatório Intermediário

Maria Eduarda Bicalho

30 de abril de 2020

1 Descrição do Problema

O projeto Maximin Share tem como objetivo fazer a divisão mais justa possível de um número de objetos com diferentes valores entre um número diferente de pessoas. O problema pincipal se centra em como realizar essa divisão. Dessa forma, três diferentes técnicas - Herística, Busca Local e Busca Global - foram utilizadas para produzir três diferentes algoritmos para executar essa partição. Neste relatório essas três implementções serão analisadas com diferentes entradas para avaliar as alterções em suas saídas. Essas entradas possuírão diferentes tamanhos, alterando significamente. Primeiramente, em relação a quantidade de pessoas e depois a quantidade de objetos. Dentro de cada uma dessas análises, serão estudados o tempo, e a qualidade da solução em relação aos diferentes dimensões de entradas. A qualidade será analisada apartir do MMS (o valor da pessoa com o menor valor), ou seja quanto maior o MMS maior a qualidade da saída.

1.1 Máquina utilizada

2 Efeito número de pessoas

Análise do impacto de uma entrada com diferentes números de pessoas nos 3 diferentes algoritmos implementados no projeto. Se foi testando a capacidade da máquina, aumentando cada vez o input de pessoas. A busca global não conseguiu um tempo factível depois de um número de entrada de 10 pessoas, dessa forma, esse algoritmo aparecerá somente na até o valor de entrada 12, depois pode-se considerar um valor de tempo significamente maior do que a dos outros dois.

2.1 Tempo

Nesta seção, a medida utilizada para a análise será a do tempo.

```
for arq in arqs:
    heuristica.append(roda('./heuristica', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsh.append(int(line[0]))
for arq in arqs:
    local.append(roda('./local', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsl.append(int(line[0]))
for arq in arqsg:
```

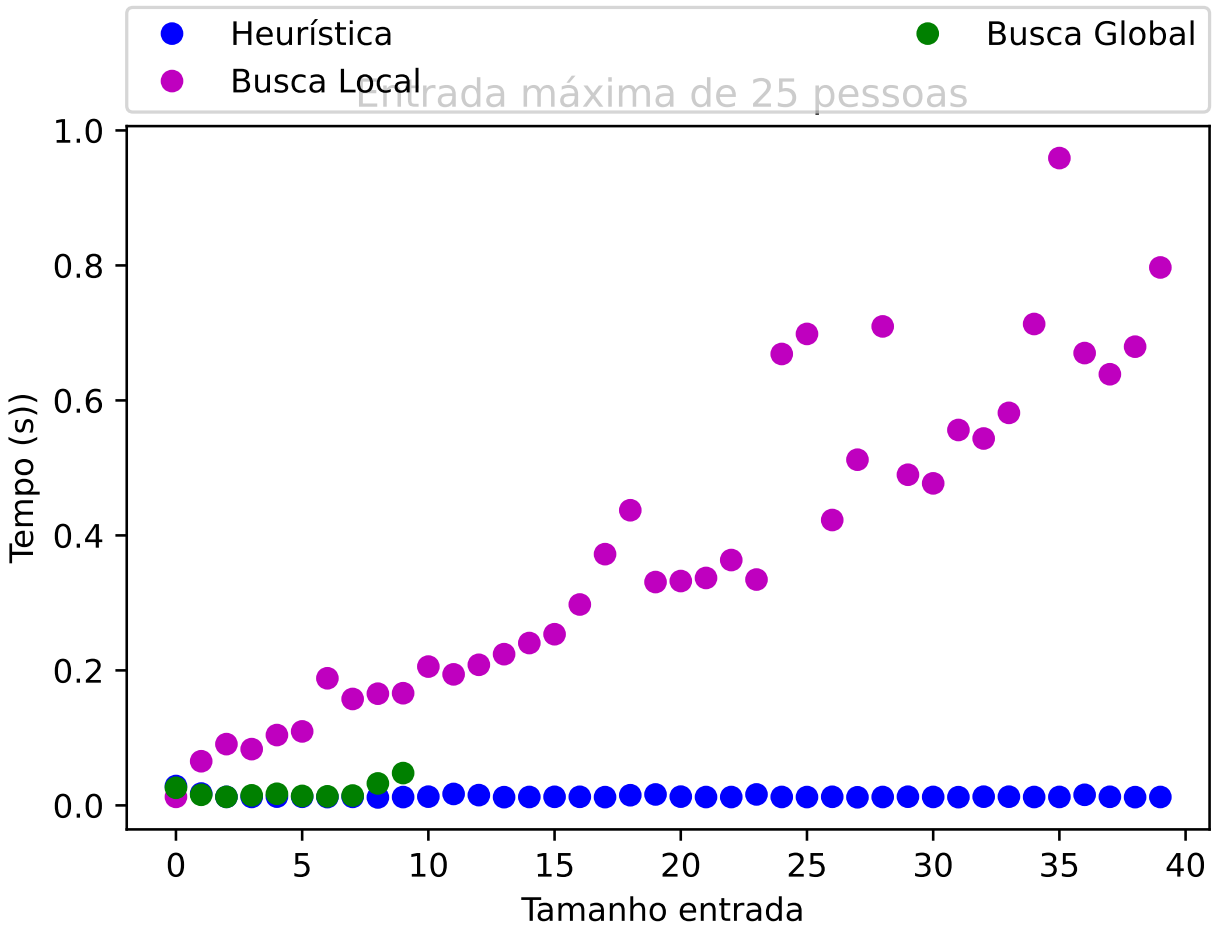
```
    exaus.append(roda('./global', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsg.append(int(line[0]))
```

```
---------------------------------------------------------------------------NameEr
Traceback (most recent call last)<ipython-input-1-f0f24a3e1536> in
<module>
      1 import matplotlib.pyplot as plt
----> 2 h,= plt.plot(x,heuristica[:10],'bo' ,label = "Heurística")
      3 l,= plt.plot(x,local[:10],'mo' ,label = 'Busca Local')
      4 g, = plt.plot(x,exaus, 'go',label = 'Busca Global')
      5 x= range(1,20,2)
NameError: name 'x' is not defined
```

```
h,= plt.plot(heuristica,'bo' ,label = "Heurística")
l,= plt.plot(local,'mo' ,label = 'Busca Local')
g, = plt.plot(exaus, 'go',label = 'Busca Global')

plt.title("Entrada máxima de 25 pessoas ")
plt.xlabel("Tamanho entrada ")
plt.ylabel("Tempo (s))")
plt.legend(handles=[h, l, g],bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left
```

```
<matplotlib.legend.Legend at 0x7f342a7a2d90>
```



A partir dos gráficos pode-se concluir que os algoritmos de busca local de busca global possuem uma variação significamente maior do que a heurística, que se mantem quase constante.

2.2 Qualidade da solução

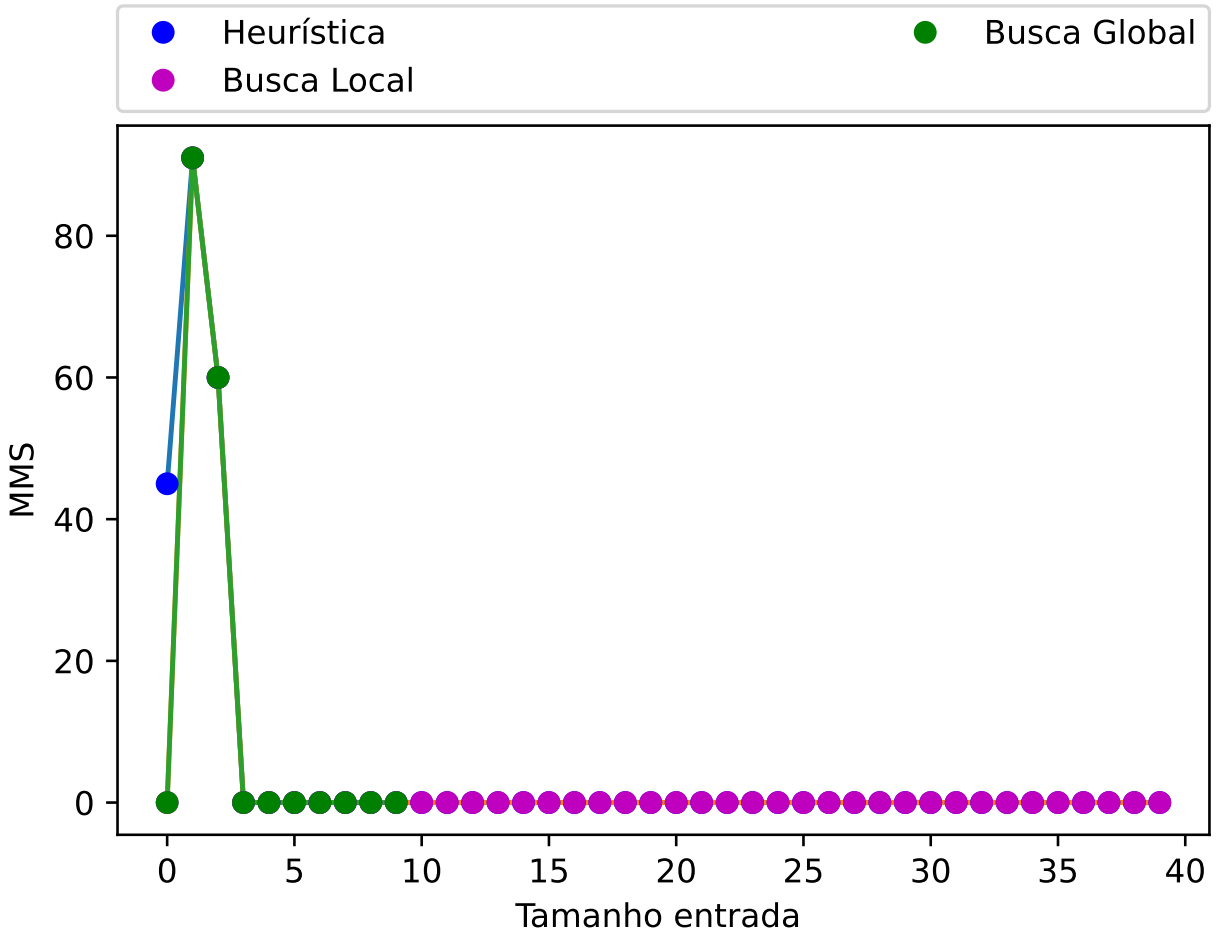
2.2.1 Gráficos

```
import matplotlib.pyplot as plt

plt.plot(outsh)
plt.plot(outsl)
plt.plot(outsg)

h,= plt.plot(outsh,'bo' ,label = "Heurística")
l,= plt.plot(outsl, 'mo',label = 'Busca Local')
g, = plt.plot(outsg,'go' ,label = 'Busca Global')
plt.xlabel("Tamanho entrada ")
plt.ylabel("MMS")
plt.legend(handles=[h, l, g],bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left
ncol=2, mode="expand", borderaxespad=0.)
```

```
<matplotlib.legend.Legend at 0x7f342a622d60>
```



3 Efeito número de objetos

```
arqs = [f"ino/in{i}.txt" for i in range(40)]
arqsg = [f"ino/in{i}.txt" for i in range(5)]


heuristica=[]
local=[]
exaus=[]
outsh=[]
outsl=[]
outsg=[]
```

3.1 Tempo

Nesta seção, a medida utilizada para a análise será a do tempo.

```
for arq in arqs:
    heuristica.append(roda('./heuristica', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsh.append(int(line[0]))
for arq in arqs:
    local.append(roda('./local', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsl.append(int(line[0]))

for arq in arqsg:
    exaus.append(roda('./global', arq)[1])
    with open("out.txt",'r') as f:
        line = f.readlines()
        outsg.append(int(line[0]))
```

3.1.1 Entrada máxima de 10 pessoas

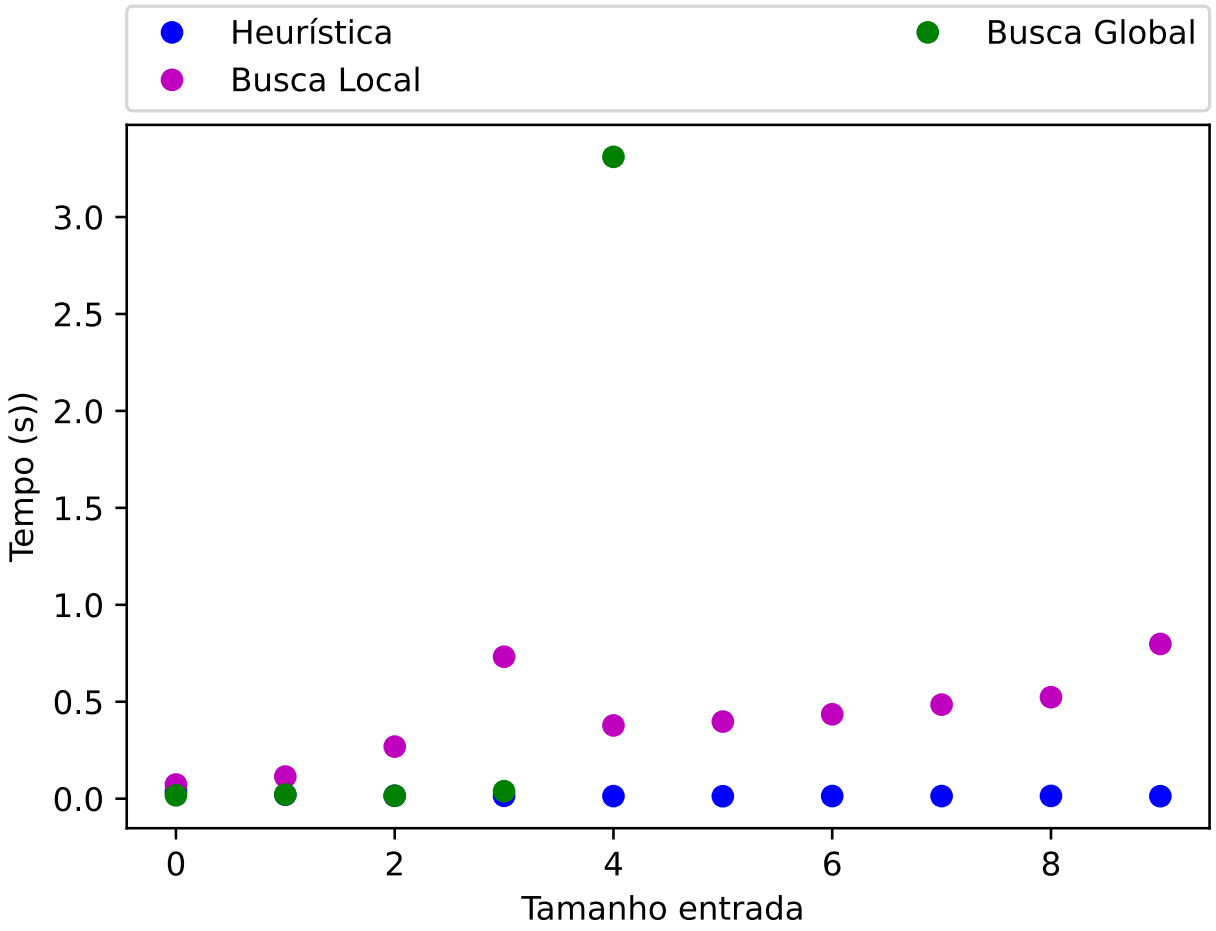
```
print(locals)
print(exaus)
import matplotlib.pyplot as plt
h,= plt.plot(heuristica[:10],'bo' ,label = "Heurística")
l,= plt.plot(local[:10],'mo' ,label = 'Busca Local')
g, = plt.plot(exaus, 'go',label = 'Busca Global')

plt.xlabel("Tamanho entrada ")
plt.ylabel("Tempo (s))")
plt.legend(handles=[h, l, g],bbox_to_anchor=(0., 1.02, 1., .102),
loc='lower left',ncol=2, mode="expand", borderaxespad=0.)
```

```
<built-in function locals>
```

```
[0.017361777001497103, 0.02116962199943373, 0.015234000002237735,
0.038703666999936104, 3.3102542769993306]

<matplotlib.legend.Legend at 0x7f342a5ad310>
```



3.2 Qualidade da solução

3.2.1 Gráficos

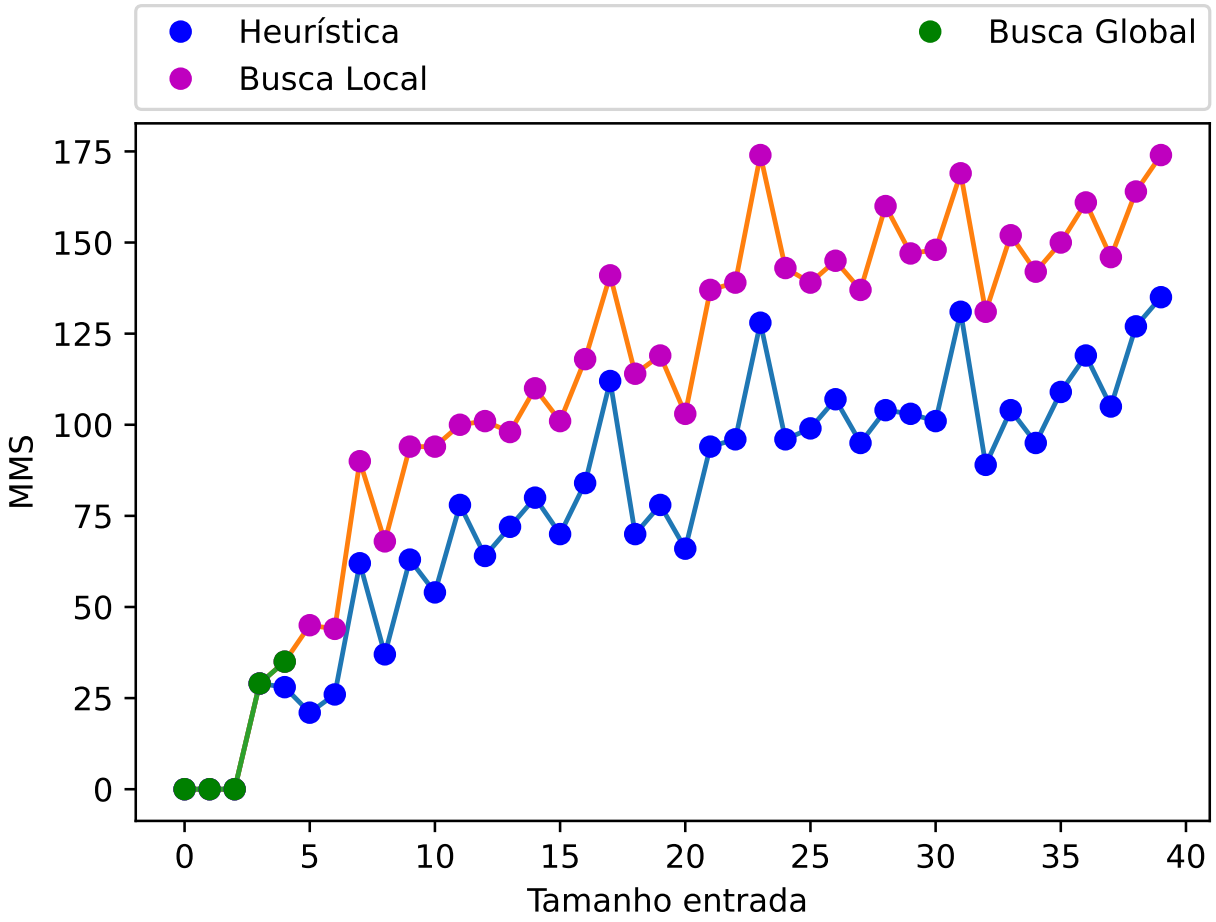
```
import matplotlib.pyplot as plt



plt.plot(outsh)
plt.plot(outsl)
plt.plot(outsg)

h,= plt.plot(outsh,'bo' ,label = "Heurística")
l,= plt.plot(outsl, 'mo',label = 'Busca Local')
g, = plt.plot(outsg,'go' ,label = 'Busca Global')
plt.xlabel("Tamanho entrada ")
plt.ylabel("MMS")
plt.legend(handles=[h, l, g],bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left
ncol=2, mode="expand", borderaxespad=0.)
```

```
<matplotlib.legend.Legend at 0x7f342a4f0730>
```



4 Conclusão