

Relatório Intermediário

Maria Eduarda Bicalho

19 de Junho 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 principal se centra em como realizar essa divisão. Dessa forma, três diferentes técnicas - Heurística, Busca Local e Busca Global - foram utilizadas para produzir três diferentes algoritmos para executar essa partição. A busca local possui 3 métodos de implementação, um sequencial (forma mais comum, está presente também na heurística e na busca global), um paralelizado com openmp e o outro utilizando gpu para paralelizar o gpu ao invés de cpu.

Os algoritmos recebem como entrada o número de pessoas, o número de objetos e os valores dos objetos. Eles retornam o MMS e os ids dos objetos de cada pessoa. Na técnica da Heurística objetos são ordenados a partir dos valores e atribuídos para as pessoas nessa ordem. Já na busca local, um algoritmo de aleatorização é utilizado para atribuir os objetos. Depois é fica-se atribuir um objeto para a pessoa que possui o mms analisado até que não seja mais possível aumentar o mms. Na busca global, uma função recursiva é utilizada para checar todas as combinações possíveis e descobrir qual retorna o maior mms.

Neste relatório essas cinco implementações serão analisadas com diferentes entradas para avaliar as alterações em suas saídas. Essas entradas possuirão diferentes tamanhos, alterando significativamente. 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 a partir 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

Memória : 1.5 GiB - Disco : 63 GB - Processador : Intel i5 - Virtualização: KVM -

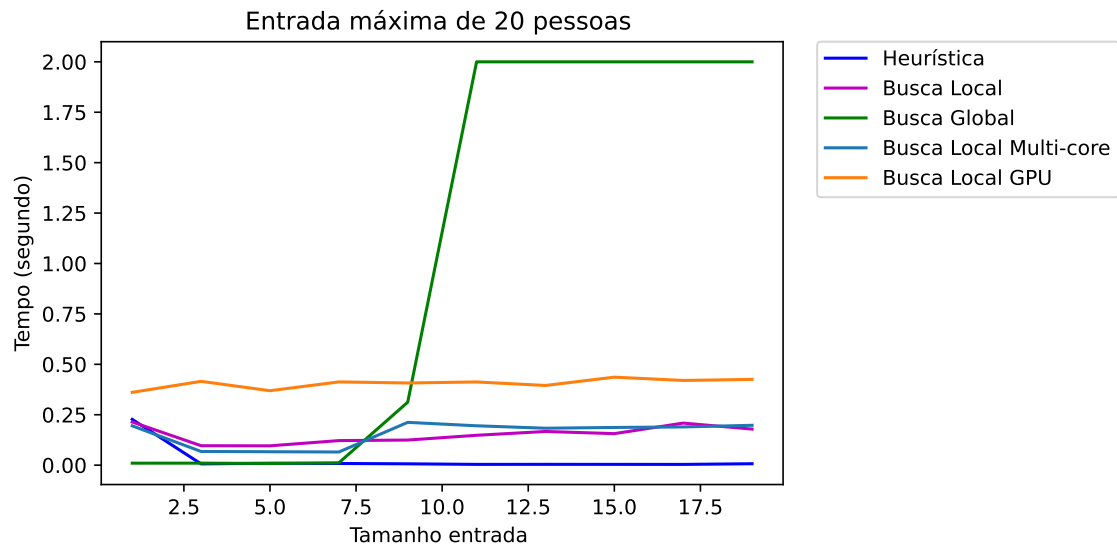
2 Efeito número de pessoas

Nessa seção será realizada a análise do impacto de uma entrada com diferentes números de pessoas nos 3 diferentes algoritmos implementados no projeto. A capacidade da máquina foi sendo testada, aumentando cada vez mais o input de com o número pessoas. A variação do número de objetos foi baixa, para que a mudança dessa variável não fosse importante para o tempo e qualidade da solução, e esses se mantivessem em função do delta pessoas. A busca global não conseguiu um tempo factível depois de um número de entrada de 10 pessoas, dessa forma, esse algoritmo aparecerá somente até o valor de entrada 10 - para as demais entradas foi assumido um valor limite.

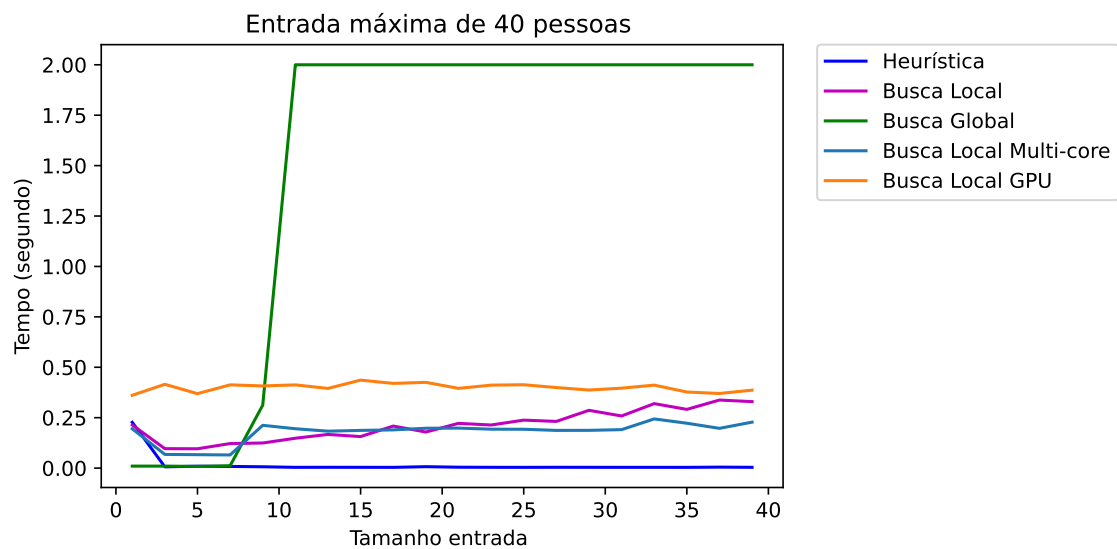
2.1 Tempo

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

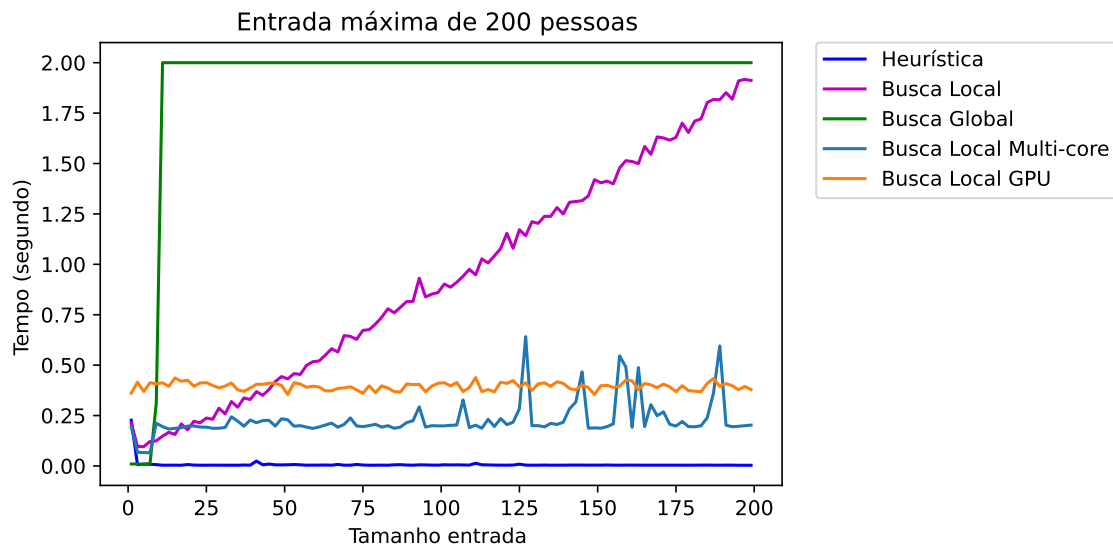
```
<matplotlib.legend.Legend at 0x7efdbbf24880>
```



```
<matplotlib.legend.Legend at 0x7efdbbe24460>
```



```
<matplotlib.legend.Legend at 0x7efdbbd5b940>
```

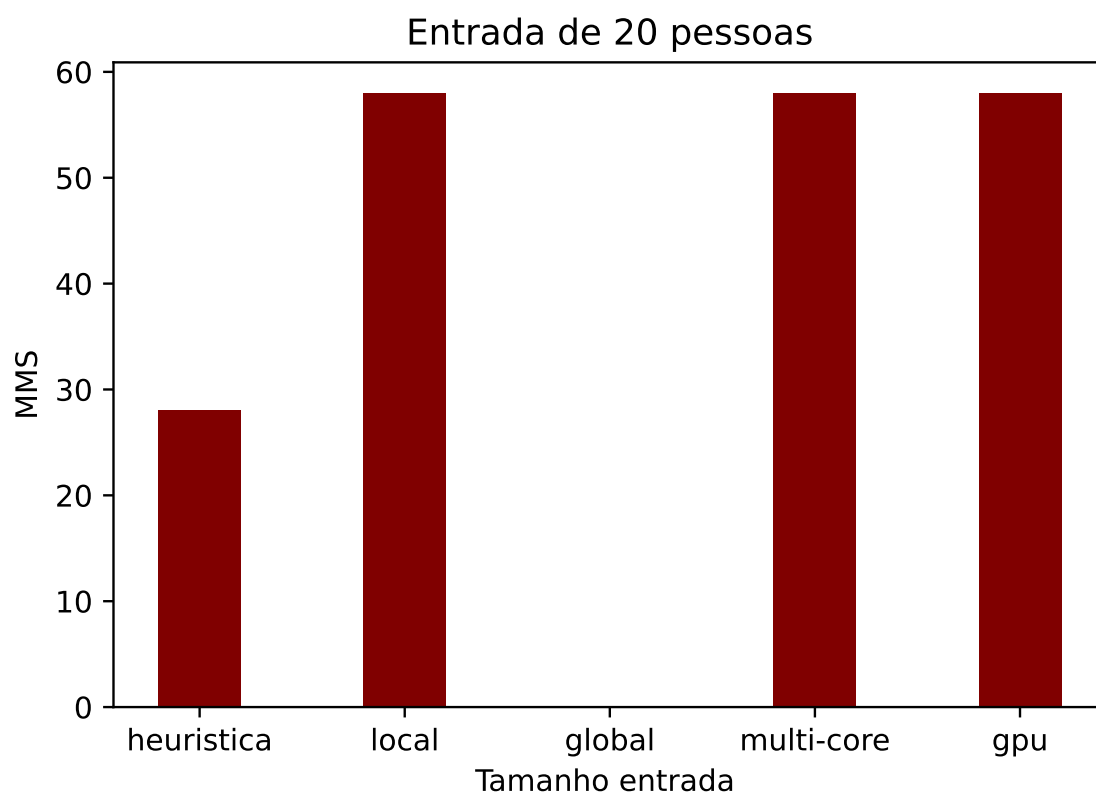


A partir dos gráficos pode-se concluir que os algoritmos de busca local de busca global possuem valores de tempo maiores do que a heurística, a qual não sofre fortes alterações. Um fato curioso que pode ser observado é que a busca local inicia com um tempo maior e logo antes da entrada de 10 - por volta da entrada de 7- é ultrapassada rapidamente pela busca global, cuja variação é extremamente alta. Por realizar a busca no resultado apresentado e não em todos os possíveis, a busca local já possui um tempo maior, mas factível na máquina utilizada e possui um resultado melhor que o da heurística. Por ser um algoritmo que utiliza recursão e percorre todos os caminhos possíveis para certificar que possui o melhor, a implementação da busca global cresce rapidamente. O único esforço da solução heurística é o de ordenar, por isso o tempo não sofre grandes alterações.

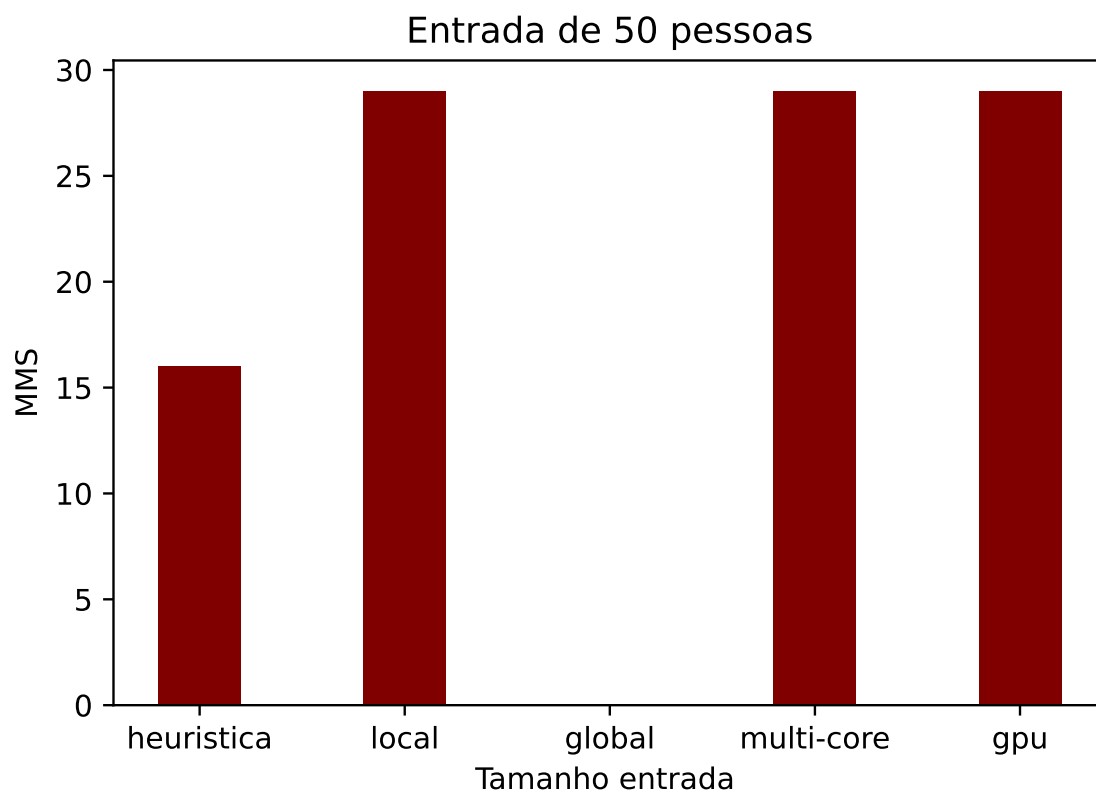
A partir do gráfico é possível notar que para entradas muito grandes, o código paralelizado com o openmp e como o programa que utiliza gpu tiveram um tempo de execução menor que a busca local serial. É possível analisar também que o tempo do programa que utiliza a GPU se mantém quase constante em 0.5 segundos, estando maior que os outros nas entradas menores e menor que os outros nas entradas maiores - maior somente que a heurística. Esse resultado mostra que utilizar a gpu é boa para entradas maiores. O código paralelo do openmp mostrou uma diferença local inicial somente em entradas bem grandes como a de 100 pessoas. Acredito que isso ocorreu devido ao uso de critical, ferramenta que aumenta bastante o tempo de execução quando utilizada.

2.2 Qualidade da solução

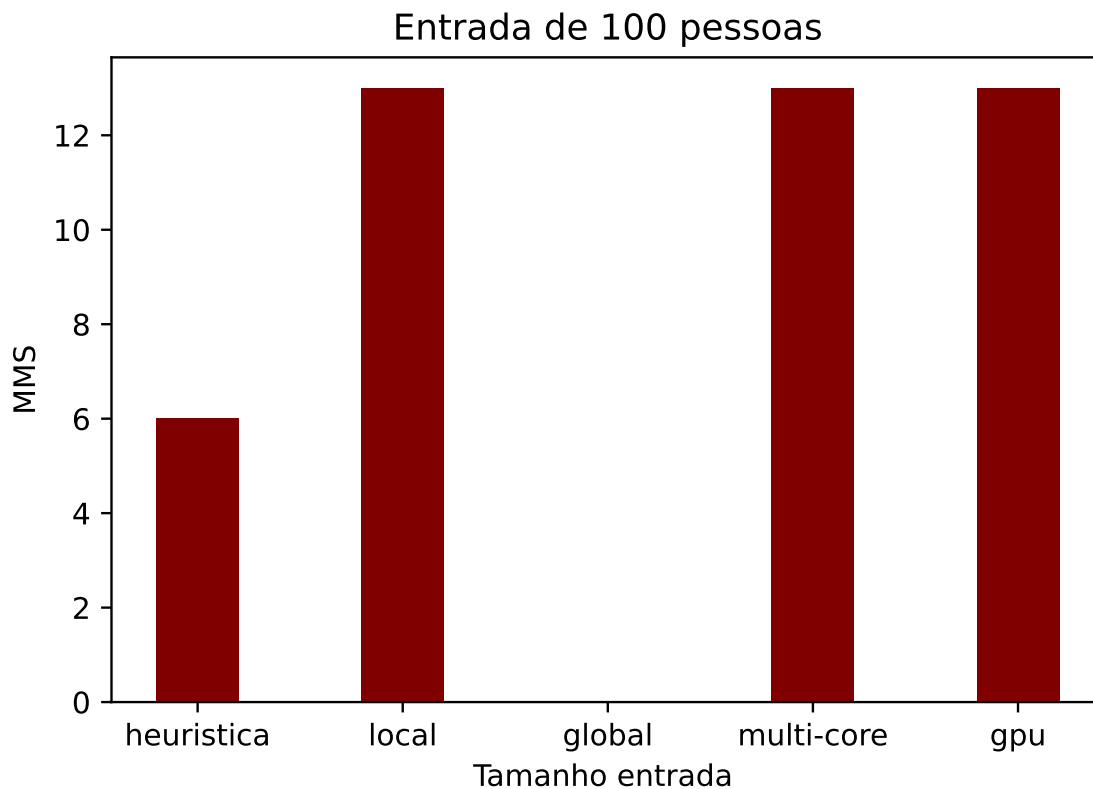
Text(0, 0.5, 'MMS')



`Text(0, 0.5, 'MMS')`



`Text(0, 0.5, 'MMS')`



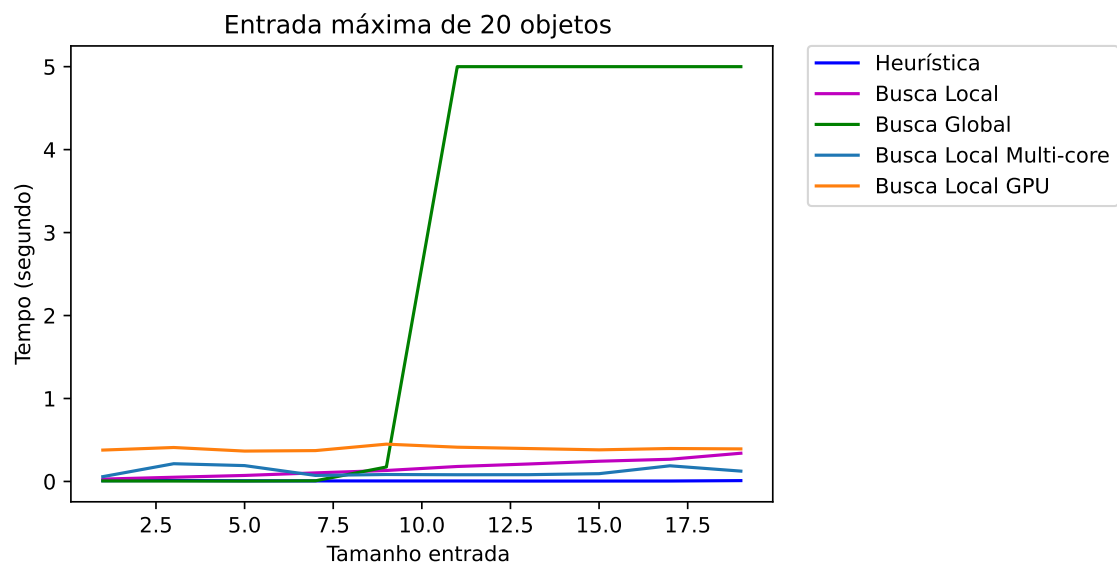
3 Efeito número de objetos

Nessa seção será realizada a análise do impacto de uma entrada com diferentes números de objetos nos 3 diferentes algoritmos implementados no projeto. A capacidade da máquina foi sendo testada, aumentando cada vez mais o input de com o número objetos. A variação do número de pessoas foi baixa, para que a mudança dessa variável não fosse importante para o tempo e qualidade da solução, e esses se mantivessem em função do delta objetos. A busca global não conseguiu um tempo factível depois de um número de entrada de 10 pessoas. Dessa forma, esse algoritmo aparecerá somente até o valor de entrada 10 - para as demais entradas foi assumido um valor limite.

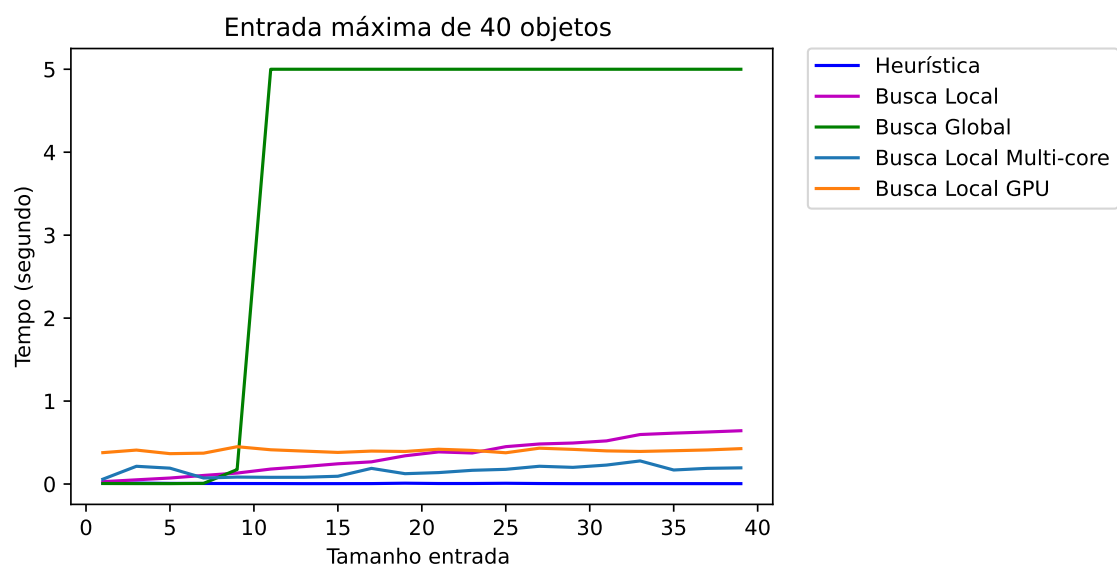
3.1 Tempo

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

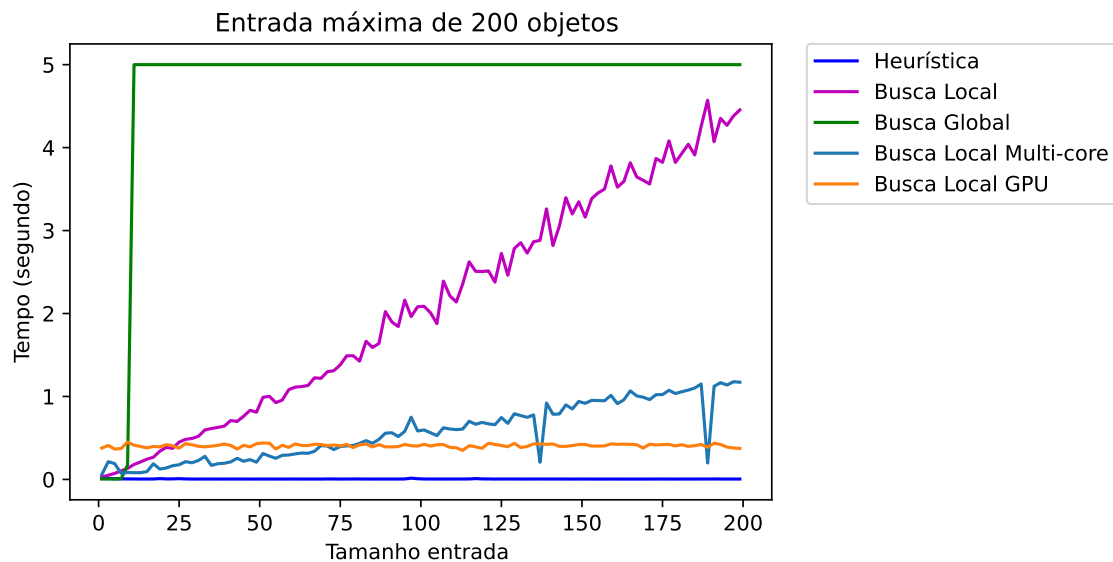
<matplotlib.legend.Legend at 0x7efdbbbc5850>



<matplotlib.legend.Legend at 0x7efdbb5ab400>



<matplotlib.legend.Legend at 0x7efdbb54f700>

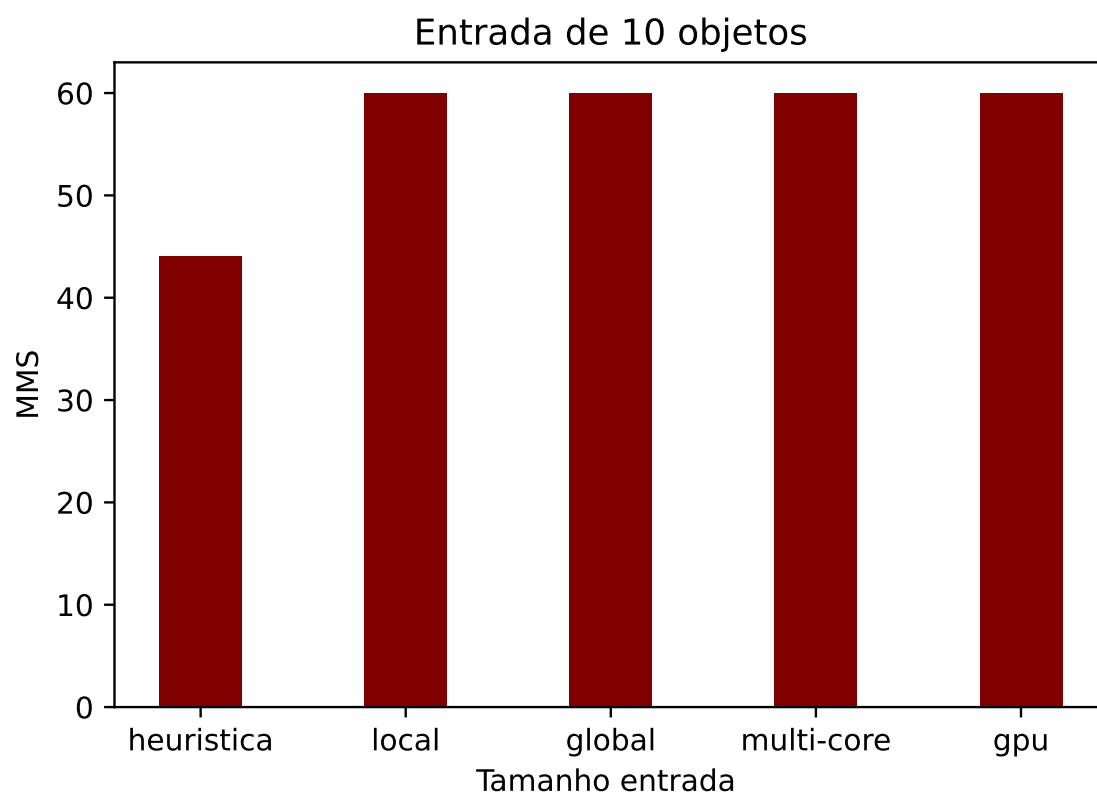


Como analisado no gráfico do tempo da seção anterior, a busca local e a busca global possuem uma variação significativamente maior do que a heurística. Nas entradas pequenas as 3 implementações possuem tempos parecidos, mas há uma relação direta com o aumento da quantidade de objetos na entrada e com o aumento do tempo da busca global e local, enquanto o tempo da heurística não altera muito. Novamente é perceptível a rápida mudança no tempo da busca global, que inicia inferior ao da busca local e depois aumenta drasticamente.

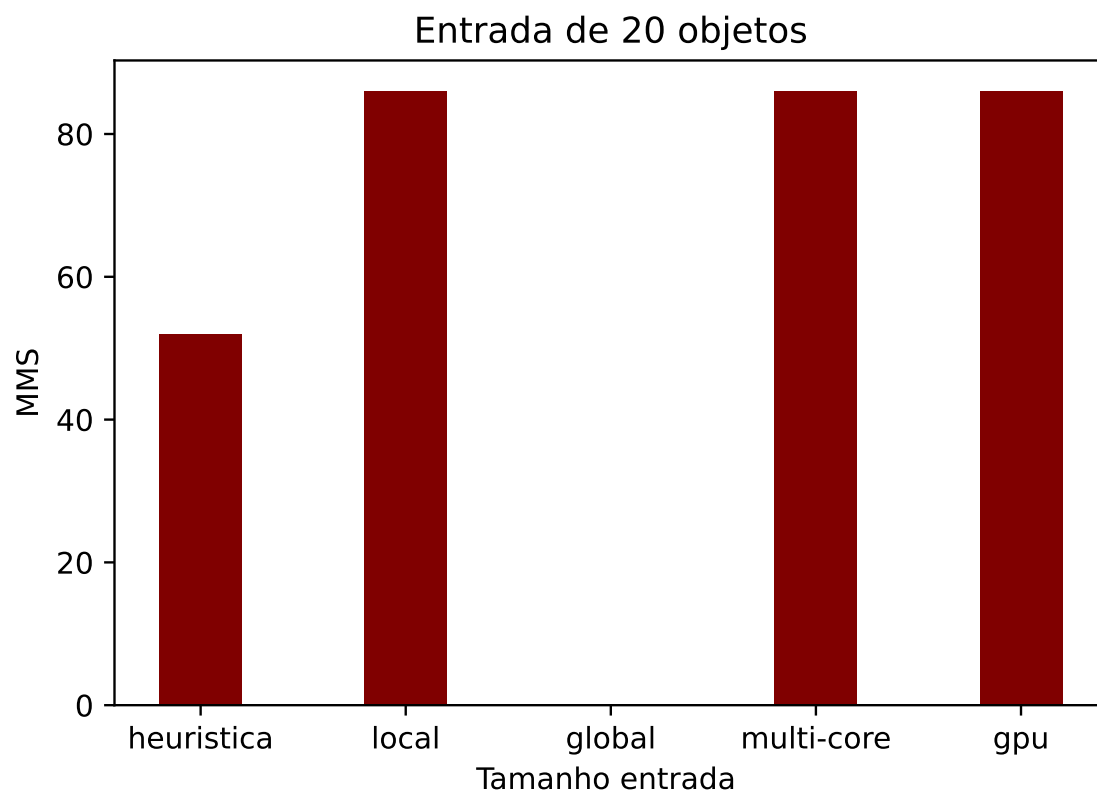
3.2 Qualidade da solução

3.2.1 Gráficos

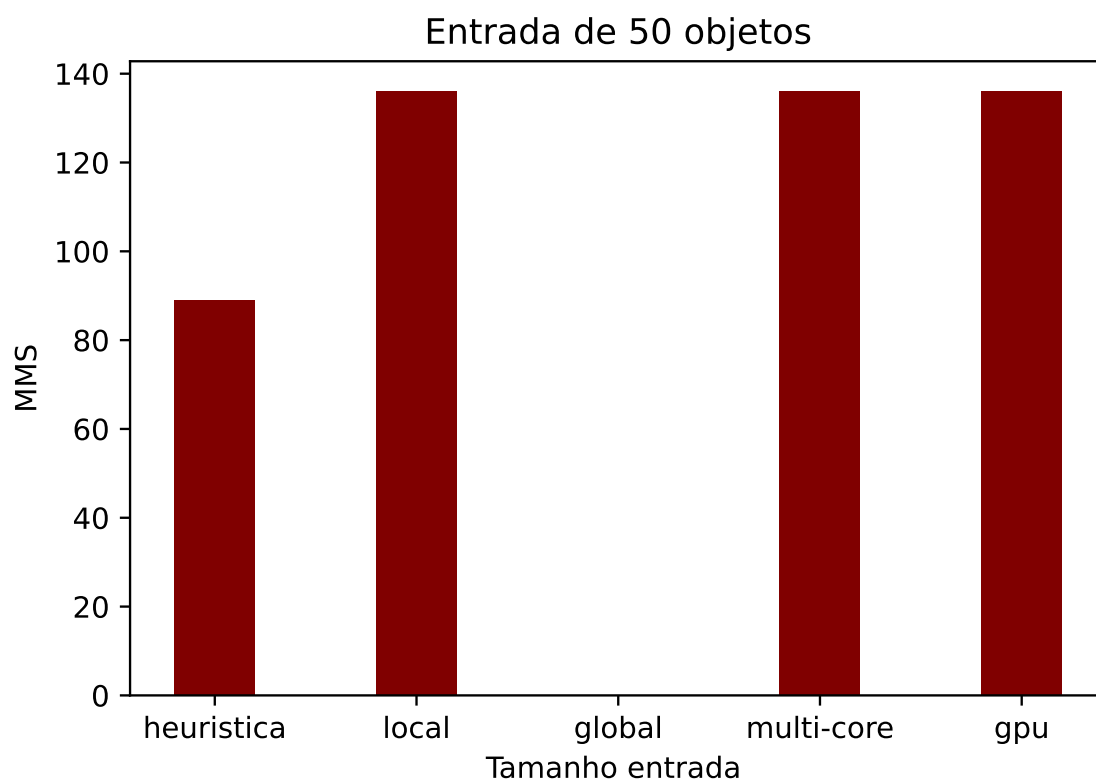
`Text(0, 0.5, 'MMS')`



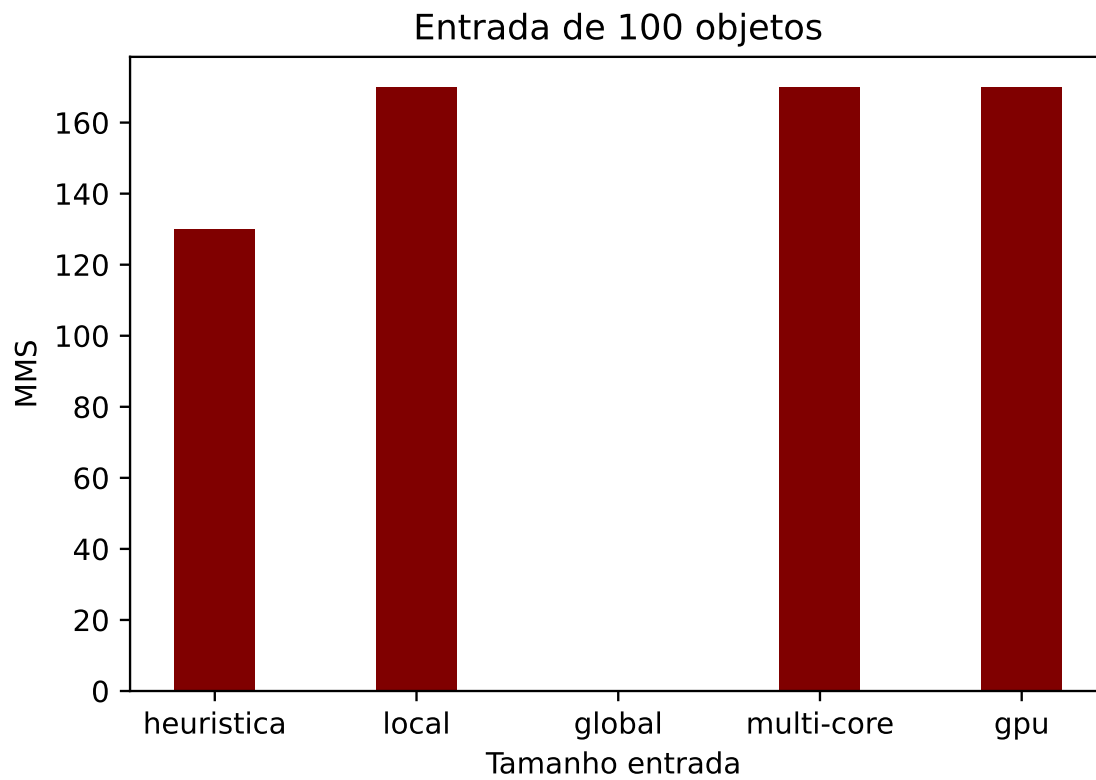
`Text(0, 0.5, 'MMS')`



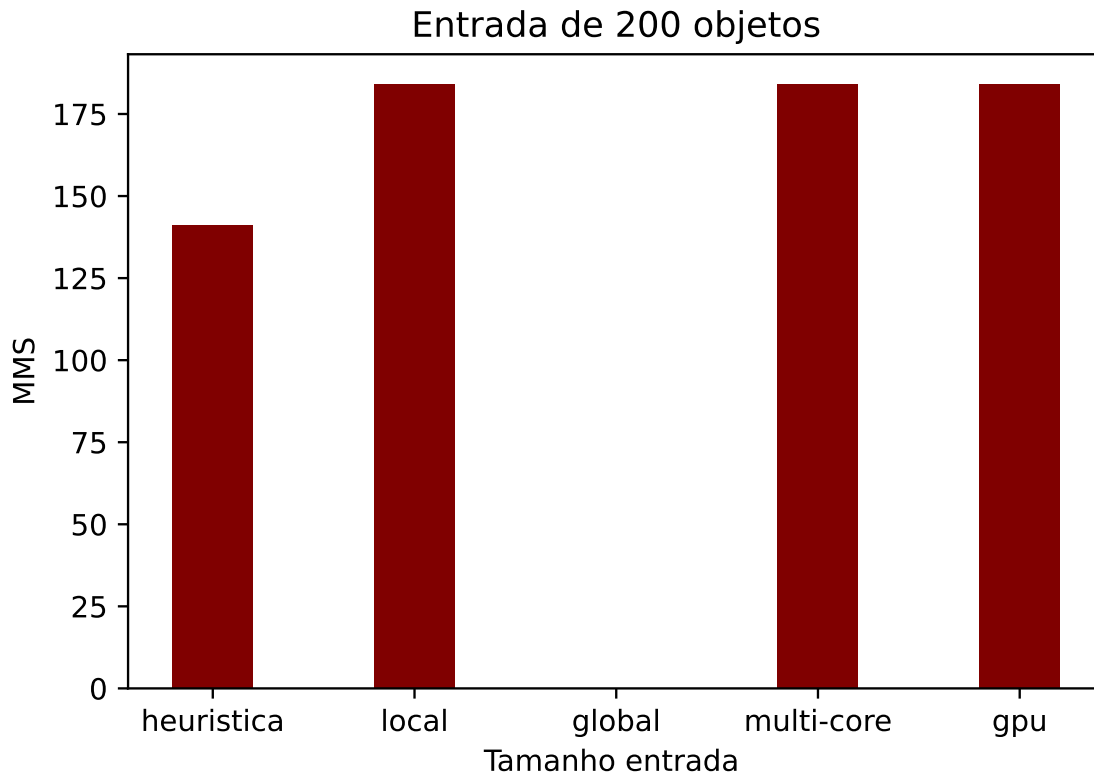
`Text(0, 0.5, 'MMS')`



Text (0, 0.5, 'MMS')



Text (0, 0.5, 'MMS')



No último gráfico de qualidade da solução pode ser observado que com entradas pequenas os três algoritmos possuem resultados parecido. Contudo, com o aumento do número de objetos a diferença entre a busca local heurística vai se tornando cada vez maior. No primeiro gráfico de qualidade da solução pode ser observado que a busca global estava crescendo com variação parecida da busca local, ou seja, pode-se ver que essas duas implementações, se possível avaliar, tivessem resultados mais parecidos do que os da busca local e da heurística.

4 Conclusão

As implementações analisadas no relatório possuem algumas características diferentes. Para entradas pequenas os algoritmos apresentam soluções com qualidades muito parecidas, e o tempo da heurística já inicia um pouco menor. Dessa forma esse algoritmo parece ser eficiente para entradas pequenas, uma vez que é mais rápido e possui saídas com alta qualidade. Na medida que a entrada aumenta o tempo da busca local e global aumenta, e a qualidade da solução da heurística cai. Assim sendo, para entradas cada vez maiores, resultados cada vez maiores são tidos a partir da busca local. Consequentemente, as implementações podem ser utilizadas para diferentes finalidades, cada uma podendo ser escolhida a partir das preferências e importâncias de determinado projeto.