Relatório Final

Matheus Pellizzon

Supercomputação - 2021.1

O problema

O problema `Maximin Share` é um problema *NP-difícil*. Sua resolução visa encontrar a distribuição mais igualitária possível de `N` objetos para `M` pessoas. É possível que determinadas pessoas acabem com mais objetos que outras ou também com maior valor acumulado que outras, uma vez que os objetos são indivisíveis. Esse problema, portanto, visa definir qual o *menor* valor que uma pessoa deveria aceitar. Basicamente, queremos maximizar o valor da pessoa que terá o menor valor. Esse é o chamado MMS.

Para exemplificar o problema, temos 6 objetos e 3 pessoas. Cada objeto possui um valor associado a ele: 20, 11, 9, 13, 14, 37. Uma possível distribuição seria:

- Pessoa 1: itens 37 - valor total: 37;
- Pessoa 2: itens 20, 11 - valor total: 31;
- Pessoa 3: itens 14, 9, 13 - valor total: 36.

Tendo em vista que cada linha representa os objetos que cada pessoa recebeu, o MMS é 31 (pessoa do meio). No entanto, é possível distribuir esses objetos de modo a obter um melhor MMS:

- Pessoa 1: itens 37 - valor total: 37;
- Pessoa 2: itens 20, 14 - valor total: 34;
- Pessoa 3: itens 13, 11, 9 - valor total: 33.

O MMS equivale a 33 (última pessoa), ou seja, essa distribuição é melhor que a outra, pois é mais igualitária.

Entradas e saídas

Por padrão, todas as entradas seguem o formato:

```
N M
v1 ... vN
```

Sendo `N` o número de objetos, `M` o número de pessoas e `vi` o valor do objeto `i`.

O formato da saída desse programa é como segue:

```
MMS
objetos da pessoa 1
...
objetos da M
```

Sendo `MMS` o valor do grupo menos valioso e `objetos da pessoa j` a lista de índices dos objetos que a pessoa `j` recebeu nessa distribuição.

Soluções

Heurística

Cada pessoa deve receber ao menos `N/M` objetos (arredondado para baixo). Para a realização da distribuição, basta ordenar os itens por valor, decrescente, e atribuí-los para cada pessoa sequencialmente. Ao chegar na última pessoa, é preciso voltar para a primeira e continuar a distribuição até acabar os objetos. Para obter a MMS dessa solução basta verificar o valor total recebido pela última pessoa.

Exemplificando com o exemplo inicial:

- 3 pessoas
- 6 objetos de valores: 20, 11, 9, 13, 14, 37.

Ordenando os objetos por valor: 37, 20, 14, 13, 11, 9.

Distribuição:

- Pessoa 1: itens 37, 13 - valor total: 50;
- Pessoa 2: itens 20, 11 - valor total: 31;
- Pessoa 3: itens 14, 9 - valor total: 23.

Portanto, o MMS para essa solução é 23.

Busca Local

A solução local consiste em algumas etapas. Inicialmente, é preciso distribuir os objetos para as pessoas de forma aleatória. Em seguida, é preciso selecionar a pessoa P com o menor valor total e verficar para cada objeto se ele pode ser doado da pessoa que o tem para a pessoa P. Um objeto pode ser doado se o valor total do doador menos o valor do item doado for maior que o valor total da pessoa P.

Caso um objeto possa ser doado, faça a doação, recalcule o MMS e repita esse processo até não existir mais doações possíveis.

Exemplificando com o exemplo inicial:

- 3 pessoas
- 6 objetos de valores: 20, 11, 9, 13, 14, 37.

Fazer uma distribuição aleatória:

- Pessoa 1: itens 20, 13 - valor total: 33;
- Pessoa 2: itens 9, 11 - valor total: 20;
- Pessoa 3: itens 14, 37 - valor total: 51.

P é a pessoa 2, pois tem o menor valor total (20). Verificando se algum item pode ser doado para ela:

- Objeto 1:

    - valor: 20;
    - dono: 1;
    - valor do dono: 33.

  Se for doado para P: - valor do antigo dono: 13; - valor do novo dono: 40.

  Como 13 é menor que 40, essa troca não pode ocorrer.

- Objeto 5:

    - valor: 14;
    - dono: 3;
    - valor do dono: 51.

  Se for doado para P: - valor do antigo dono: 37; - valor do novo dono: 34.

  Como 37 é maior que 34, essa troca pode ocorrer.

Então o novo MMS é 33, dada a nova distribuição:

- Pessoa 1: itens 20, 13 - valor total: 33;
- Pessoa 2: itens 9, 11, 14 - valor total: 34;
- Pessoa 3: itens 37 - valor total: 37.

As verificações de troca devem ser feitas novamente até não ser possível. Com a distribuição atual não é possível realizar mais trocas, portanto essa seria uma possível resolução local. O próximo passo consiste em repetir todas as etapas algumas vezes e escolher qual resolução apresenta o maior MMS.

Busca Local em CPU e GPU

As etapas da busca local são as mesmas. O paralelismo tem o objetivo de acelerar o processo de geração de resoluções possíveis e escolher a melhor.

Busca exaustiva

Utilizando recursão, é possível gerar todas as distribuições possíveis dados N itens e M pessoas, obtendo M^N distribuições. Basta escolher como resolução a distribuição em que o MMS é máximo.

Para 3 itens de valores 11, 37, 20 e 2 pessoas (representadas por 0 e 1) as seguintes distribuições são possíveis:

Item 37	Item 20	Item 11	Valor 0	Valor 1
0	0	0	68	0
0	0	1	57	11
0	1	0	48	20
0	1	1	37	31
1	0	0	11	57
1	0	1	20	48
1	1	0	11	57

Item 37	Item 20	Item 11	Valor 0	Valor 1
1	1	1	0	68

O MMS nessa situação é 31. É possível perceber que existem casos "espelhados", que acabam consumindo recursos desnecessariamente.

A descrição acima dá a entender que quanto mais itens e mais pessoas mais possibilidades serão processadas e maior tempo será consumido para execução. Para contornar essa desvantagem, é possível determinar um valor de corte para parar a execução caso a distribuição seja "ruim". O método consiste em ordenar os itens por valor (decrescente) e, durate a distribuição, verificar se a pessoa atual possui valor maior que o valor de corte. Caso a condição seja verdade, essa distribuição já irá levar a um cenário não ótimo, então pode ser pulada.

Exemplificando:

Valor de corte: (37 + 20 + 11)/2 = 34:

Item 37	Item 20	Item 11	Valor 0	Valor 1
0	0	-	57	0

obs: "-" ainda não distribuído.

Portanto, nem é necessário realizar a distribuição do item 11.

Testes

Para avaliar o desempenho das soluções anteriores foram geradas entradas variando `N` (e `M` fixo) e outras com `M` variando (e `N` fixo). Essa decisão foi feita para facilitar a visualização e interpretação dos resultados. Submetendo cada um dos algoritmos aos mesmos testes garante que os resultados finais serão justos em relação ao tempo de execução. Além disso, permitem comparar os MMSs obtidos para cada algortimo, a fim de determinar qual o melhor método para resolução do `Maximin Share`.

Vale ressaltar que os algoritmos de busca local, sejam eles sequencial, paralelo em CPU e GPU, repetem a busca 100000 vezes. Isso ocorre com a finalidade de obter uma solução local que equivale a um ponto de máximo global.

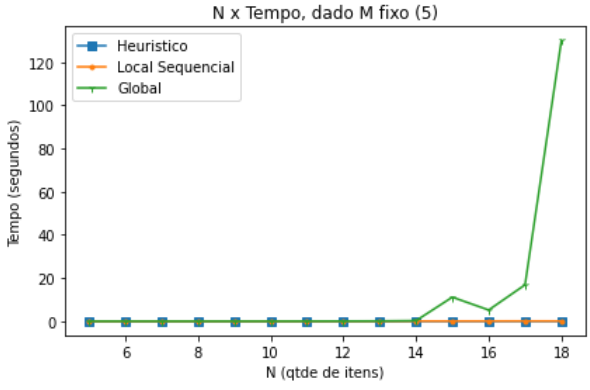
Os testes foram realizados em uma máquina com CPU de 4 *cores* (Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz) e uma GPU com 384 CUDA cores (GeForce 940MX).

Testes executados em todas as soluções, com M fixo:

N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_Global	T_Global
14	5	92	0.007312	130	0.059702	130	0.236005
15	5	146	0.019012	184	0.058811	184	11.148053
16	5	88	0.016759	125	0.070490	125	5.194191
17	5	117	0.018752	159	0.083992	159	16.863090
18	5	163	0.015947	204	0.075088	204	130.027999

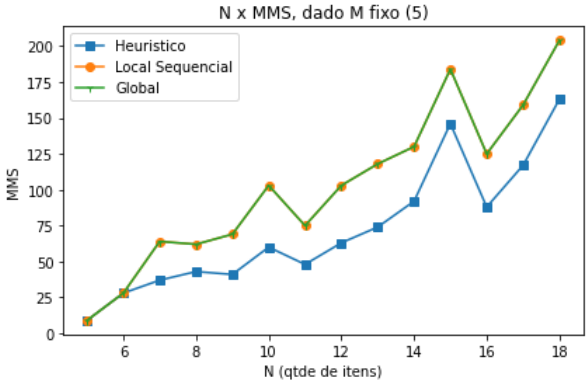
Os resultados obtidos para um M fixo podem ser observados na tabela anterior. O que mais chama atenção nesse resultados é o tempo de execução da solução global, que pula de aproximadamente 16.86 segundos para 130.03 segundos.

A visualização do crescimento do tempo de execução dessa solução pode ser observada no gráfico abaixo. O tempo de execução da global, mesmo utilizando a técnica de *branch and bound*, se comporta como uma exponencial. Caso fosse viável executar testes em maiores quantidades, seria possível observar que a busca do melhor MMS entre muitos cenários, passaria de minutos para horas, de horas para dias, e assim em diante.



Também é possível observar as diferentes soluções em relação ao MMS da solução. A busca global possui MMS ótimo, como esperado. A busca local, no entanto, alcança os mesmos valores em tempos muito mais razoáveis.

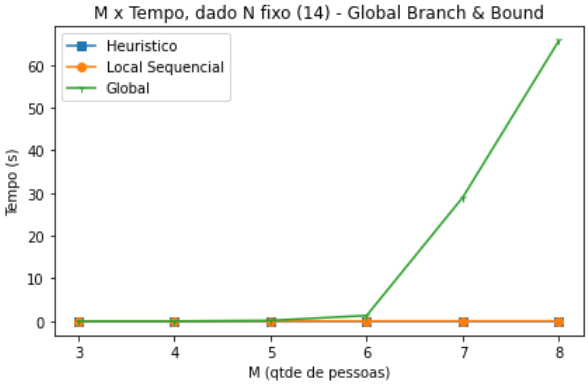
Como a estratégia delimitada envolve repetir a busca local um determinado número de vezes vezes (100000 vezes, como descrito no início dos testes), a probabilidade de se obter um ponto de máximo local que equivale ao máximo global é alto. Ao mesmo tempo que apresenta MMS ótimo, o tempo e os recursos consumidos pela local não chegam perto da global. Isso pode ser visto principalmente em entradas com muitos itens ou muitas pessoas, onde é viável executar a busca local mas não é viável executar a exaustiva.



Testes executados em todas as soluções, com N fixo:

N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_Global	T_Global
14	4	133	0.005054	166	0.054012	166	0.009035
14	5	119	0.006722	156	0.052893	156	0.169022
14	6	126	0.004374	156	0.052282	156	1.337289
14	7	48	0.004261	94	0.060653	94	29.037224
14	8	77	0.004367	98	0.060246	98	65.731313

Variando o número de pessoas, é possível observar o mesmo resultado, que já era esperado dada a conclusão anterior. Mesmo utilizando uma estratégia para "cortar" algumas etapas da busca exaustiva, o tempo de execução volta a se comportar, eventualmente, como uma exponencial.



É possível perceber a diferença entre realizar "cortes" na busca ou não realizá-los. A marca dos 30 segundos é atingida com 7 pessoas e 14 objetos (~ 678.22 bilhões de possibilidades) para o algoritmo que possui a estratégia de *branch and bound* (acima). Já para o algoritmo global que deve passar por todos os cenários (abaixo), os 50 segundos são atingidos com 5 pessoas e 12 itens (~ 244.14 milhões de possibilidades). Esses dados permitem concluir que é possível melhorar a performance da busca global. No entanto, a estratégia delimitada não é suficiente para superar os tempos reduzidos da busca local.

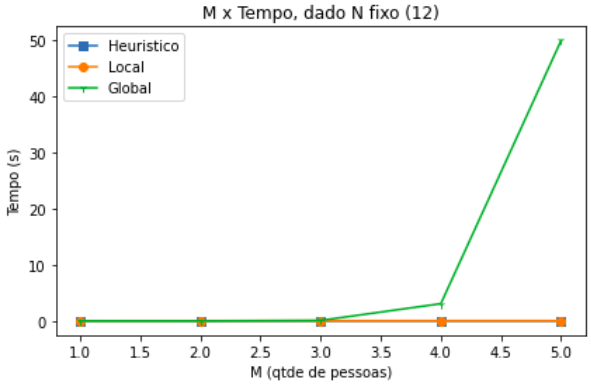
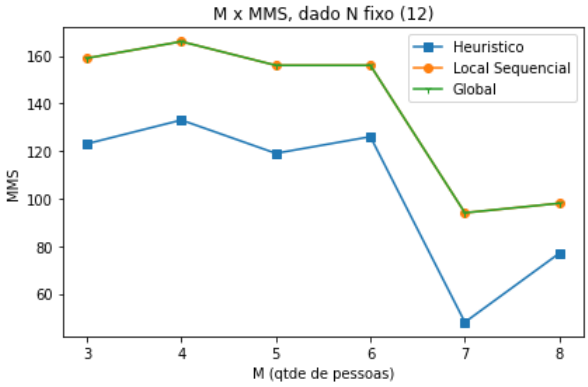


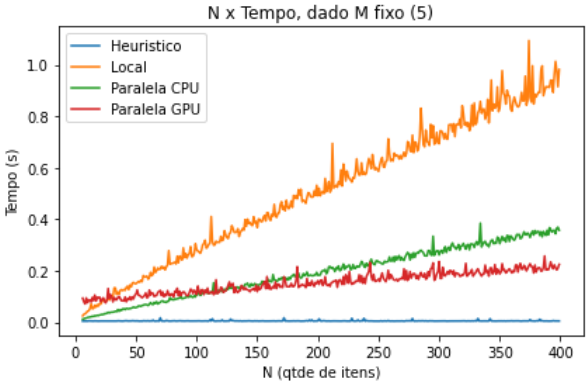
Gráfico retirado do relatório intermediário, em que o algoritmo da busca global não possuia a estratégia delimitada e deveria percorrer todos os M^N cenários para atingir o melhor MMS.

O mesmo caso obervado anteriormente para se repete para o MMS também. Os valores de MMS para as entradas geradas são bem próximas entre as diferentes soluções.



Testes executados na solução heurística e local, com M fixo:

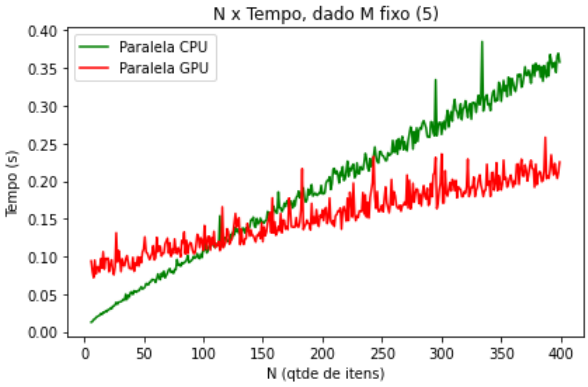
N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_CPU	T_CPU	MMS_GPU	T_GPU
6	5	56	0.005048	63	0.025669	63	0.012694	63	0.094023
7	5	34	0.005059	40	0.031266	40	0.013675	40	0.080435
8	5	40	0.005313	57	0.033896	57	0.016210	57	0.071809
9	5	21	0.005027	37	0.037290	37	0.016708	37	0.095054
10	5	61	0.004998	99	0.044056	99	0.019048	99	0.076727



Apesar do tempo de execução da busca local crescer quase que linearmente, enquanto o tempo da heurística permanece praticamente constante, o MMS da local é melhor que o heurístico.

Retomando os cenários da busca global, demorariam milhares de horas para encontrar a solução ótima para 5 pessoas e 90 itens, por exemplo. A busca local se destaca nesse aspecto; é possível obter uma solução ótima (ou pelo menos bem próxima) que funciona para entradas grandes e é executada em um tempo muito menor que da global.

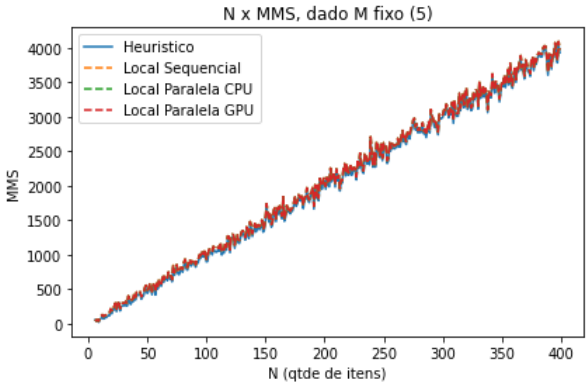
Além da comparação entre local e heurística anterior, vale ressaltar as melhorias (em relação ao tempo) obtidas nas buscas locais com paralelismo em CPU ou GPU. Como esperado, o MMS é o mesmo, independente da versão do algoritmo (paralela em CPU ou GPU e sem paralelismo).

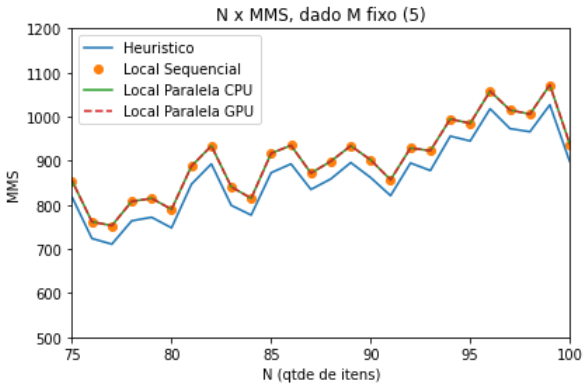


Para a paralela em CPU, o tempo de execução diminui com base na quantidade de threads disponíveis. Como a máquina utilizada para a execução desses testes possui 4 *cores*, era esperado que o tempo fosse reduzido em quase 1/4 (no mundo ideal). No entanto, o que realmente é observado é uma redução de aproximadamente 2/3 da local sequencial. Mesmo assim, ainda não é possível igualar ao tempo da heurística, dada a natureza das operações e complexidade do problema.

Outro fato relevante para esses casos remete ao custo fixo de realizar operações em GPU. Existe um overhead para a parte sequencial do programa. Assim, é possível obervar que o desempenho da GPU é pior que a implementação sequencial e paralela em CPU para entradas reduzidas (nesse teste, menos que 100 objetos). Isso leva a conclusão de que não é necessário paralelizar um programa só porque existe essa possibilidade.

Para entradas suficientemente grandes, o paralelismo em GPU começa a superar ambas as locais, devido a quantidade de *threads* que a GPU possui. No entanto, como a GPU executa a parte paralela do programa em *chunks*, a melhora no tempo de execução é limitada pela quantidade de *chunks* que estarão em execução simultaneamente, além do tempo fixo para alocação de memória, por exemplo. Mesmo assim não é melhor que a heurística (em relação ao tempo de execução).



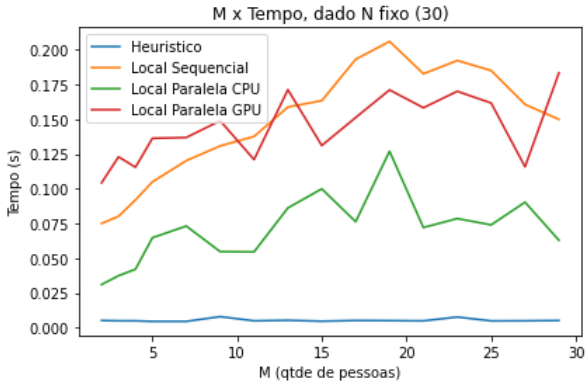


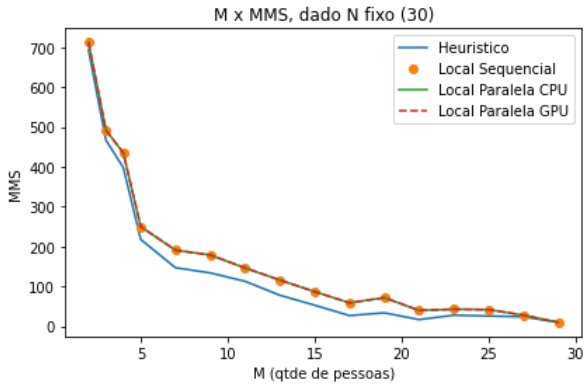
Ainda em relação ao MMS, no gráfico anterior, não é muito óbvia a diferença entre as duas soluções, dada a escala utilizada. A diferença no MMS obtido pela local é significante, quando comparada com o MMS obtido pela heurística.

Testes executados na solução heurística e local, com N fixo:

N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_CPU	T_CPU	MMS_GPU	T_GPU
30	2	693	0.005122	712	0.074950	712	0.031032	712	0.104218
30	3	467	0.004926	492	0.080139	492	0.037363	492	0.122924
30	4	398	0.004899	435	0.091913	435	0.042023	435	0.115439
30	5	218	0.004504	250	0.104916	250	0.064698	250	0.136261
30	7	147	0.004502	191	0.120150	191	0.073120	191	0.136836

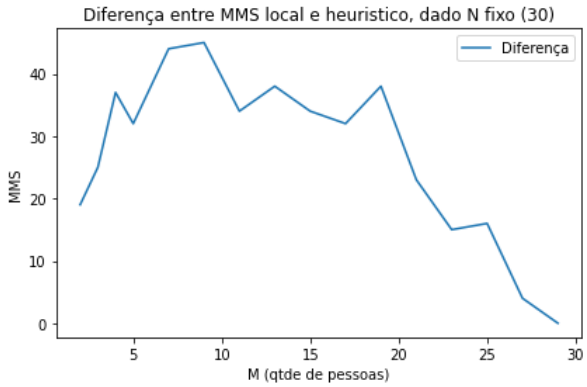
Como visto para os testes com M fixo, o tempo que a busca local leva para ser executada cresce com o tamanho das entradas.





Nesses testes com N fixo, o mesmo padrão percebido anteriormente se repete. Na maioria dos casos a variação de MMS entre as soluções locais e heurística são relevantes.

Vale ressaltar que a busca local encontra uma distribuição de "máximo local", por causa da distribuição aleatória dos itens para as pessoas. Novamente, dada a estratégia adotada, é provável chegar a uma resposta ótima para uma distribuição. No entanto ainda é possível que o ponto ótimo não seja encontrado. No caso médio, no entanto, a busca local acaba sendo a melhor solução, quando consideramos tanto o seu tempo de execução (se comparada a global) e a qualidade do resultado (se comparada a heurística).



Testes executados na solução heurística e local, com itens com valores semelhantes:

Por causa do resultado anterior, uma nova análise pode ser realizada.

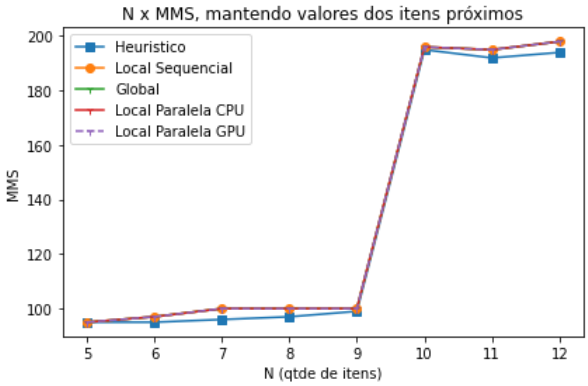
A busca heurística não é ruim, como aparenta ser dados os testes anteriores. Para casos em que os itens possuem valores muito parecidos, o resultado da heurística é equiparável tanto ao da local quanto da global.

Nos testes seguintes tanto N quanto M foram variados, e os itens possuem valores próximos.

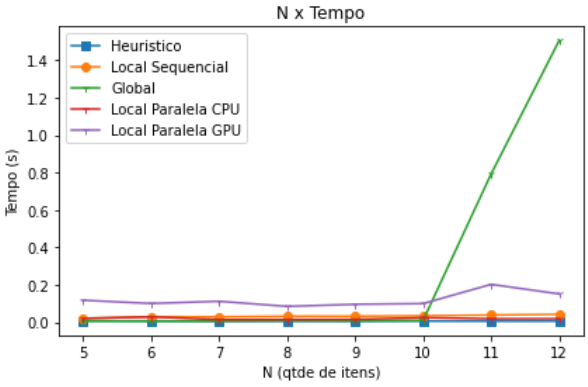
Nos gráficos, um dos eixos foi fixado como o número de itens para facilitar a visualização dos resultados, visto que o intuito dessa análise é a qualidade das diferentes soluções quanto ao MMS obtido.

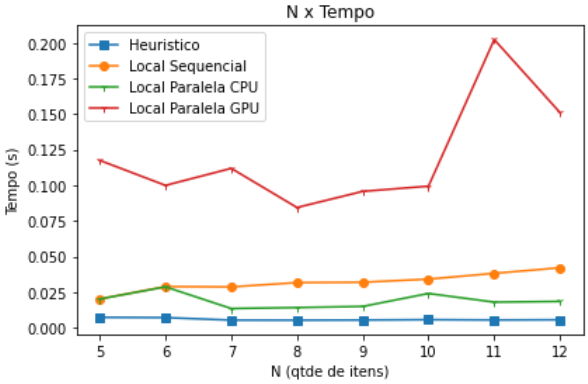
N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_Global	T_Global	MMS_CPU	T_CPU	MMS_GPU	T_GPU
5	5	95	0.007352	95	0.020313	95	0.006079	95	0.020206	95	0.117534
6	5	95	0.007180	97	0.028957	97	0.005253	97	0.028748	97	0.099971
7	5	96	0.005382	100	0.028733	100	0.005766	100	0.013552	100	0.111922
8	5	97	0.005341	100	0.031724	100	0.008856	100	0.014226	100	0.084448

N	M	MMS_Heuristico	T_Heuristico	MMS_Local	T_Local	MMS_Global	T_Global	MMS_CPU	T_CPU	MMS_GPU	T_GPU
9	5	99	0.005446	100	0.031996	100	0.007143	100	0.015029	100	0.095767



Nesse caso, há um pulo no valor do MMS simplesmente porque cada pessoa passou a receber pelo menos dois 2 itens. O valor bem próximo, se não igual, entre o MMS das soluções fica em evidência.





Como esperado, o tempo da heurística e da local continuam inferiores ao da global (1° gráfico acima), e o tempo das locais é marginalmente superior ao da heurística (2° gráfico acima). A local em GPU não performa bem nesse teste dado o custo fixo de utilizá-la, característica abordada anteriormente.

Conclusão

Tendo os testes e resultados acima como base, a conclusão que pode ser tirada é que nenhuma solução é melhor que a outra. Tudo depende do cenário (as entradas). Dados cenários diversos, com entradas grandes, que dificilmente seriam executados na global, ou com objetos de valores muito diferentes, que remetem a "fraqueza" da heurística, a busca local sequencial e suas versões em paralelo se destacam.

Comparando as locais, em específico as paralelas, é possível concluir que tudo depende do cenário. Não adianta paralelizar com GPU uma entrada pequena.