Relatório

Aluno	Matrícula
Daniel Maike Mendes Gonçalves	16/0117003
Joberth Rogers Tavares Costa	16/0128013

1 - Explicação do Algoritmo

Para entender mais sobre o princípios de satisfatibilidade booleana:
https://pt.wikipedia.org/wiki/Problema_de_satisfatibilidade_booliana

- Primeira versão do algoritmo, a dupla trabalhou em uma versão sequencial da solução, em que fazia a leitura do número de variáveis e cláusulas, e após isso salvaria as cláusulas em um vector<vector> para que depois possa fazer as verificações. Depois de salvar as cláusulas o algoritmo começa a leitura de fulls e flips, em que a cada leitura ele faz o valor atual das variáveis e os compara com as cláusulas, salvando as que não foram satisfeitas e os literais em um map para que seja feita a contagem de suas aparições e antes de realizar o print da saída para cada full ou flip ele faz a ordenação desse map a partir de uma conversão para vector<pair<int, int>> e um comparador criado para ordenar com base em suas aparições e se empatar, o valor absoluto do maior número vem primeiro, após a impressão ele segue para o próximo full ou flip até alcançar o final.

- Segunda versão do algoritmo, seguindo a mesma lógica da primeira versão, as variáveis se tornaram globais para que sejam compartilhadas entre as threads, para poder separar entre as threads salvamos cada full e cada flip em um vector<map<int,int>> global em que se for full ele captura todos os valores e se for flip ele pega o map da iteração anterior e realiza apenas o flip da variável solicitada. Após terminada a leitura de todos os fulls e flips as threads iam pegando o próximo da lista que não foi resolvido até o final, e para cada resolução ele salvava em variáveis globais os resultados para poder realizar o print em ordem após a resolução de todos os fulls e flips. Essa segunda versão apresentou vários gargalos por salvar muito dado em memória, e em casos gigantescos ele quebrava por falta de memória.

- Terceira versão do algoritmo, seguimos a abordagem de tratar por blocos, separamos em blocos de tamanho 12000 fulls/flips e após isso criamos o tanto de threads determinadas e elas vão resolvendo um conjunto dentro desses blocos, por exemplo se forem 12 threads a primeira thread vai resolver o caso 0 e o caso 12, a segunda thread resolveria o caso 1 e o caso 13, sempre somando a quantidade de threads para não colidir. Após a resolução de um bloco, é esperado todas as threads com join, é realizado o print da solução do bloco sequencialmente e depois segue para o próximo bloco criando novas threads até que acabe.

Ambiente Executado

Para executar os testes foi usado as máquinas da chococinno particulamente para rodar toda a bateria de testes do problema.

2) Dados Coletados

OBS: Para realizar os testes envolvendo a saída jogada para /dev/null e executando o mesmo código com os prints comentados, foi testado usando apenas o time limit de 60 segundos.

2.1) Dados gerais

2.1.1) Qtd. de testes que ultrapassaram a baseline (1 thread)

Qtd. algortimos que ultrapassaram a baseline	
Contagem ultrapassada (TOTAL)	TOTAL
27	170
Contagem ultrapassada (timelimit 10)	TOTAL
2	34
Contagem ultrapassada (timelimit 30)	TOTAL
5	34
Contagem ultrapassada (timelimit 60)	TOTAL
1	34
Contagem ultrapassada (timelimit 120)	TOTAL
13	34
Contagem ultrapassada (timelimit 240)	TOTAL
6	34

2.1.2) Qtd. de testes que ultrapassaram a baseline (6 thread)

Qtd. algortimos que ultrapassaram a baseline	
Contagem ultrapassada (TOTAL)	TOTAL
54	170
Contagem ultrapassada (timelimit 10)	TOTAL
2	34
Contagem ultrapassada (timelimit 30)	TOTAL
4	34
Contagem ultrapassada (timelimit 60)	TOTAL
8	34
Contagem ultrapassada (timelimit 120)	TOTAL
19	34
Contagem ultrapassada (timelimit 240)	TOTAL
21	34

2.1.3) Qtd. de testes que ultrapassaram a baseline (8 thread)

Qtd. algortimos que ultrapassaram a baseline	
Contagem ultrapassada (TOTAL)	TOTAL
54	170
Contagem ultrapassada (timelimit 10)	TOTAL
4	34
Contagem ultrapassada (timelimit 30)	TOTAL
8	34
Contagem ultrapassada (timelimit 60)	TOTAL
5	34
Contagem ultrapassada (timelimit 120)	TOTAL
19	34
Contagem ultrapassada (timelimit 240)	TOTAL
18	34

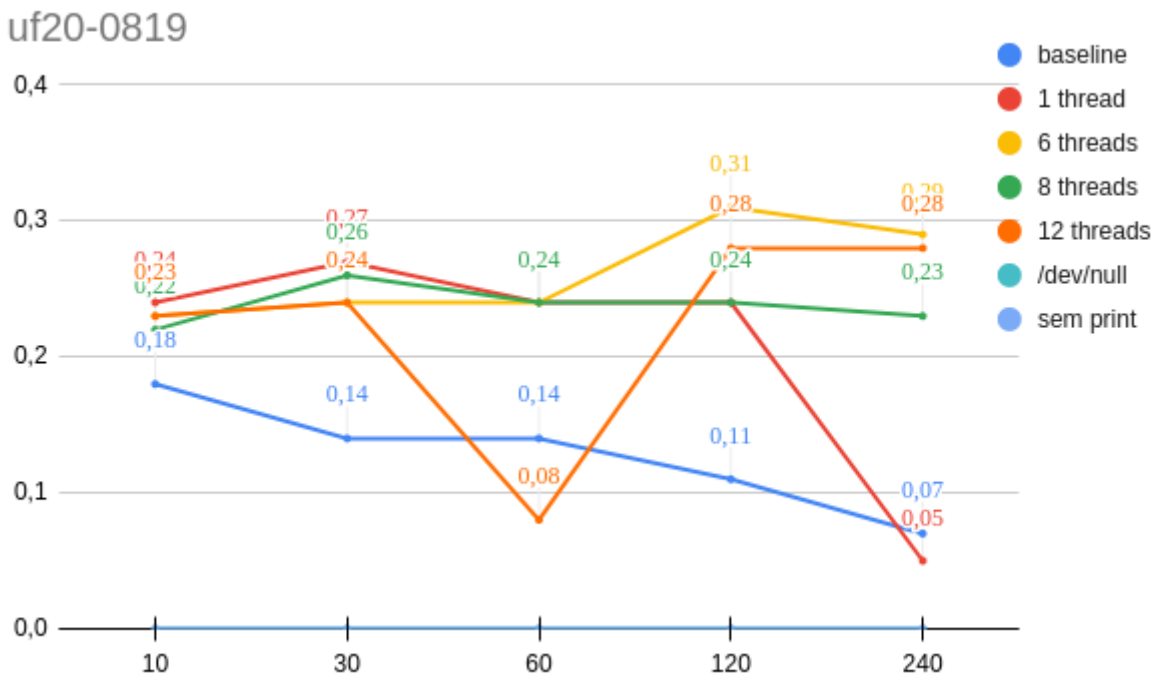
2.1.4) Qtd. de testes que ultrapassaram a baseline (12 thread)

Qtd. algortimos que ultrapassaram a baseline	
Contagem ultrapassada (TOTAL)	TOTAL
58	170
Contagem ultrapassada (timelimit 10)	TOTAL
2	34
Contagem ultrapassada (timelimit 30)	TOTAL
10	34
Contagem ultrapassada (timelimit 60)	TOTAL
9	34
Contagem ultrapassada (timelimit 120)	TOTAL
16	34
Contagem ultrapassada (timelimit 240)	TOTAL
21	34

Entradas Menores

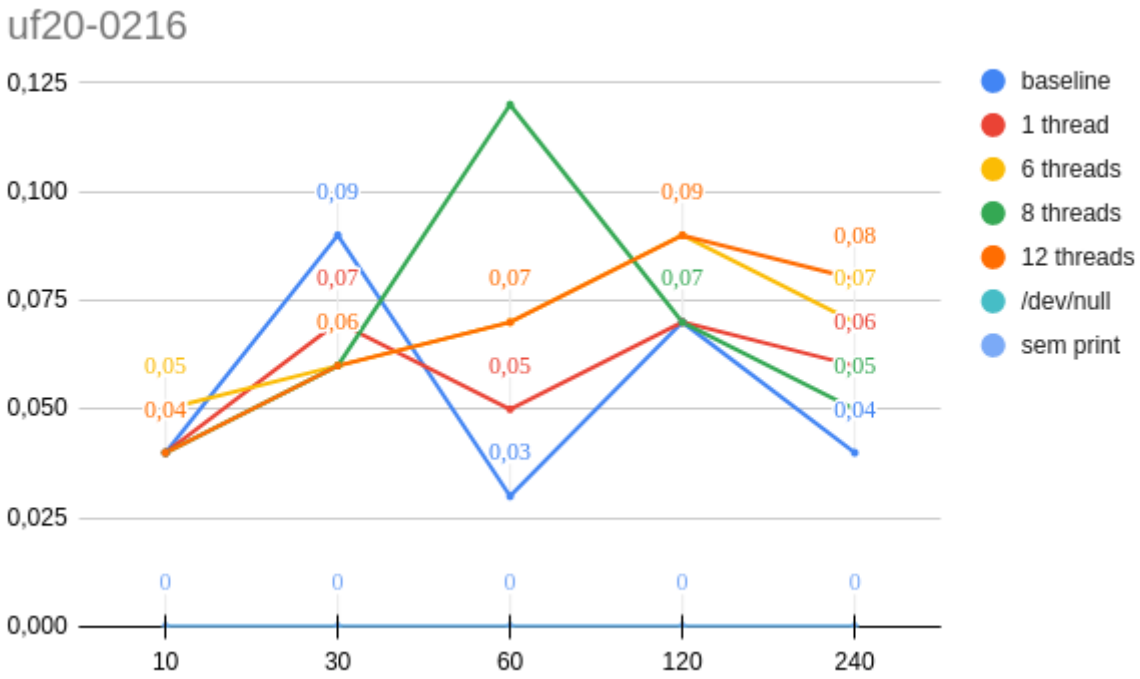
2.2.1) uf20-0819

Para essa medição é possível verificar que resultado conseguiu ser melhor que a baseline em apenas dois pontos durante as medições coletadas, são elas com 12 threads com timelimite de 60 segundos e com apenas 1 thread com timelimite de 240 segundos. Durante as medições que serão vistas mais abaixo, o algorítimo não se adequou muito bem a entradas muito pequenas, perdendo para o algorítimo do baseline.



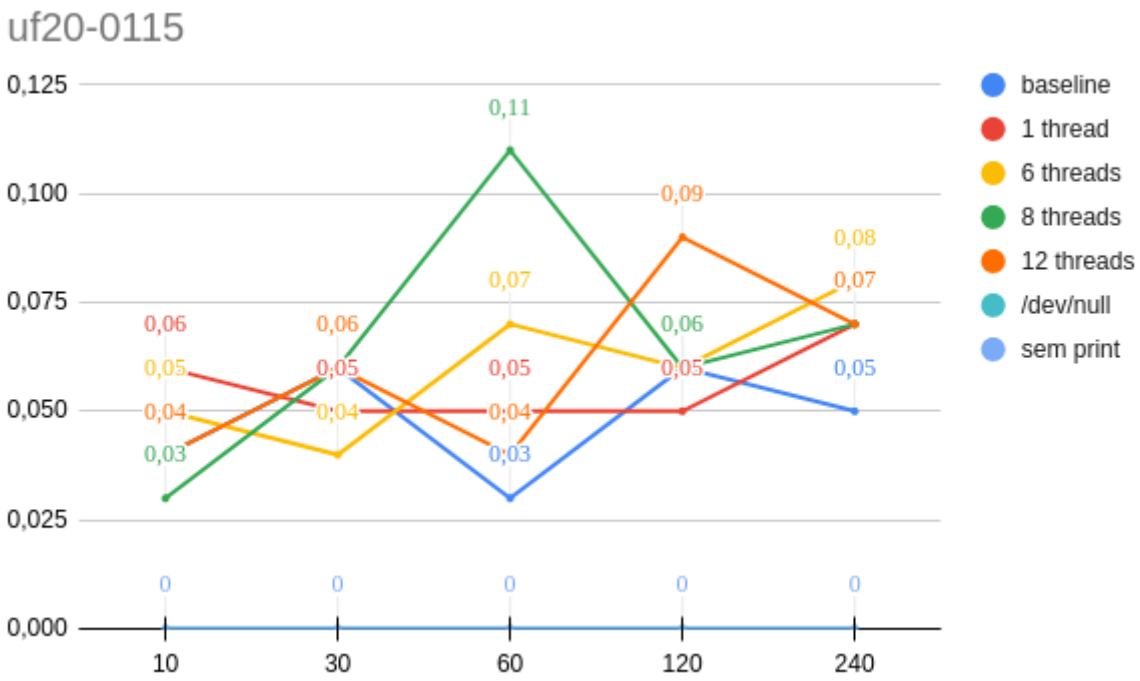
2.2.2) uf20-0216

Aqui houve medições discrepante em vários níveis com valores em diferentes patamares. Para esse arquivo de entrada, os valores que conseguiram bater o valores colhidos na baseline apenas com timelimit 10 e 60.



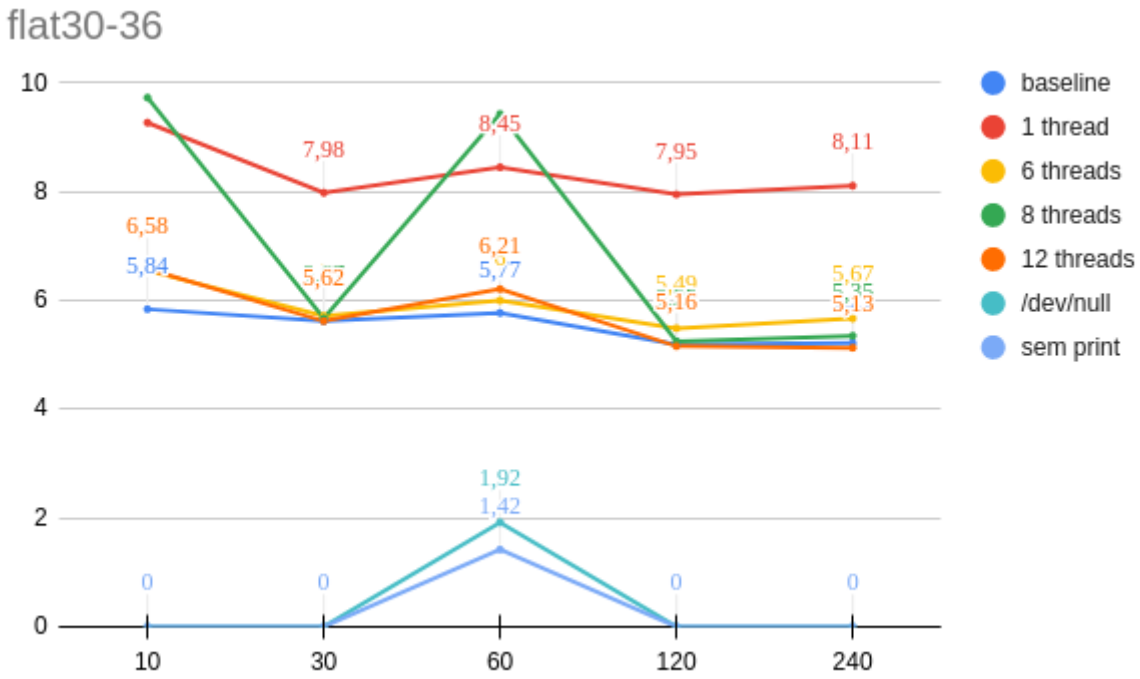
2.2.3) uf20-0115

Novamente houve resultados muito variados em diferentes timelimits. Como são entradas muito pequenas comparadas as outras, conseguiram se sair bem em relação a baseline apenas em timelimits muito baixos.



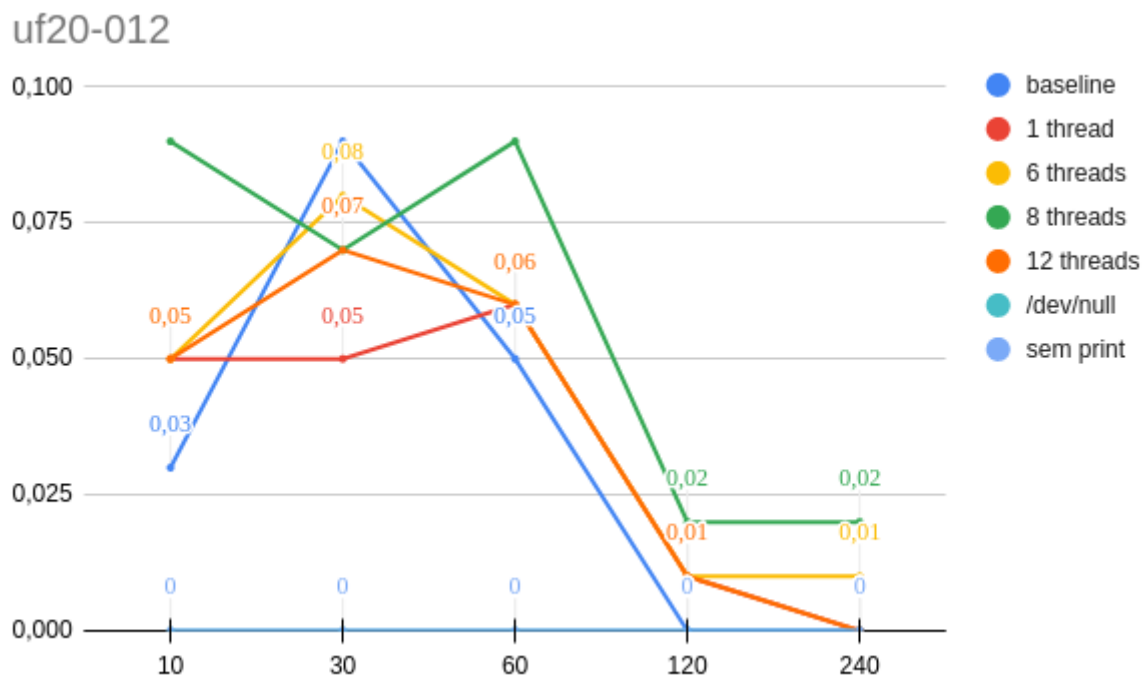
2.2.4) flat30-36

Mais um exemplo de arquivo com entradas pequenas, onde não foi possível ter resultados positivos em relação a baseline, o único resultado considerado foi com time limit 240 segundos e 12 threads.



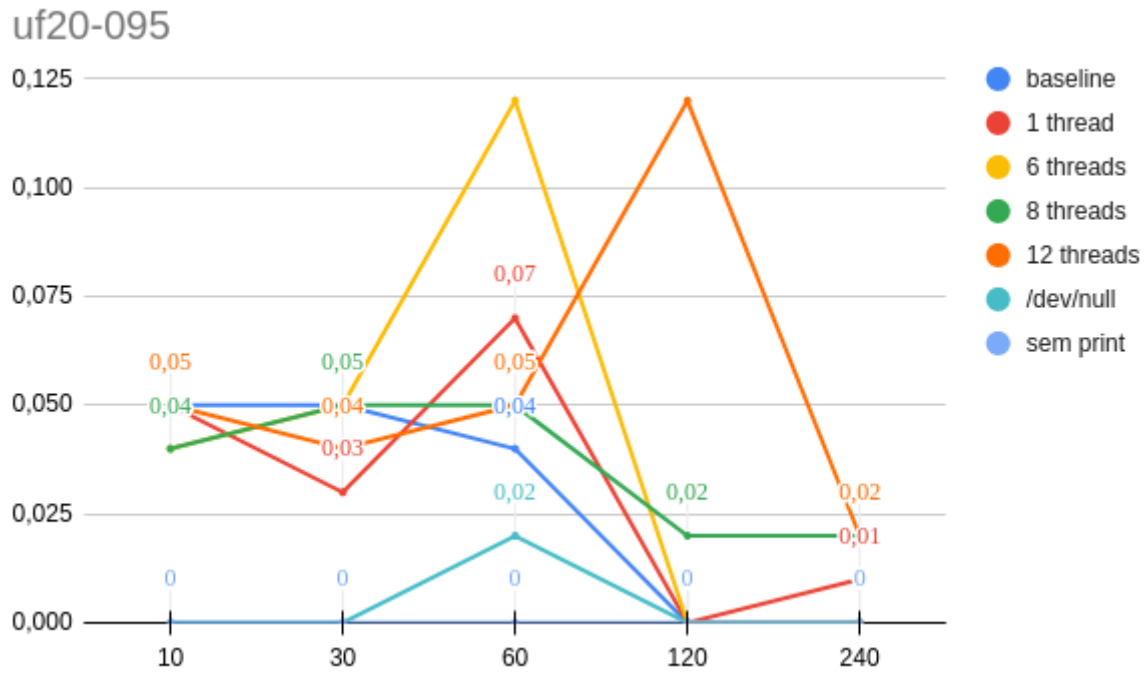
2.2.5) uf20-012

Apesar de ser um arquivo com poucas entradas comparado aos outros, é legal notar que de acordo com o aumento do time limit, os resultados foram tendendo cada vez mais ao tempo zero. Nesse caso, o algorítimo conseguiu ultrapassou a baseline apenas nos primeiros time limits, e depois foram tendo tempos piores.

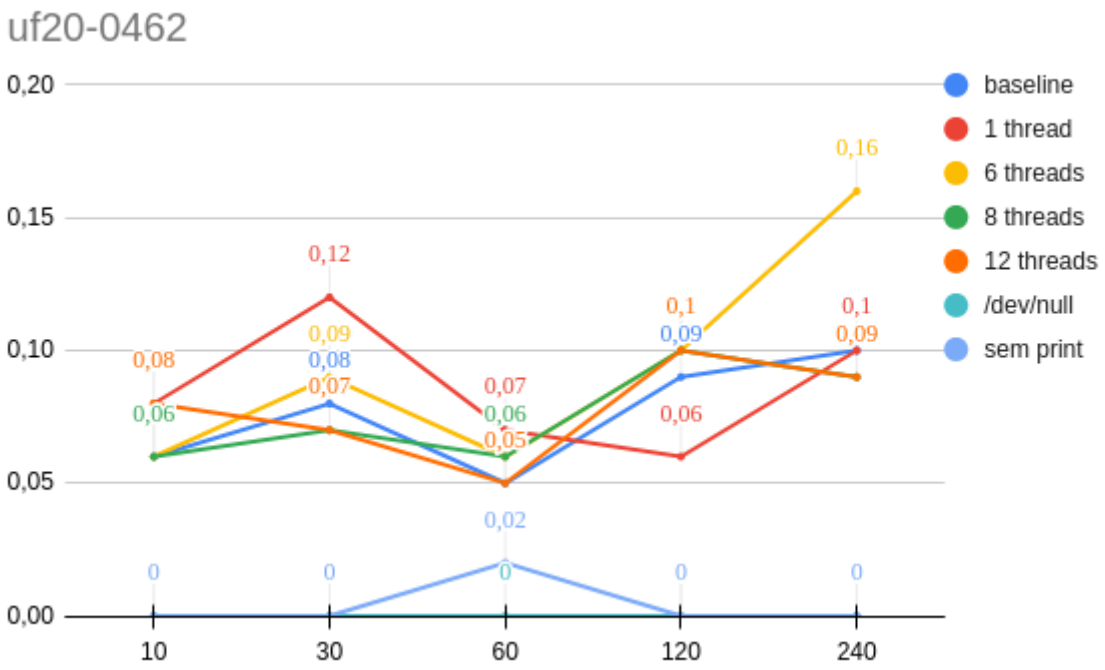


2.2.6) uf20-095

Novamente com um arquivo com poucas entradas comparado aos outros. Os resultados positivos para essa bateria de testes foram persistidos apenas em time timits menores que 60 segundos, sendo eles com 1, 8 e 12 threads respectivamente.

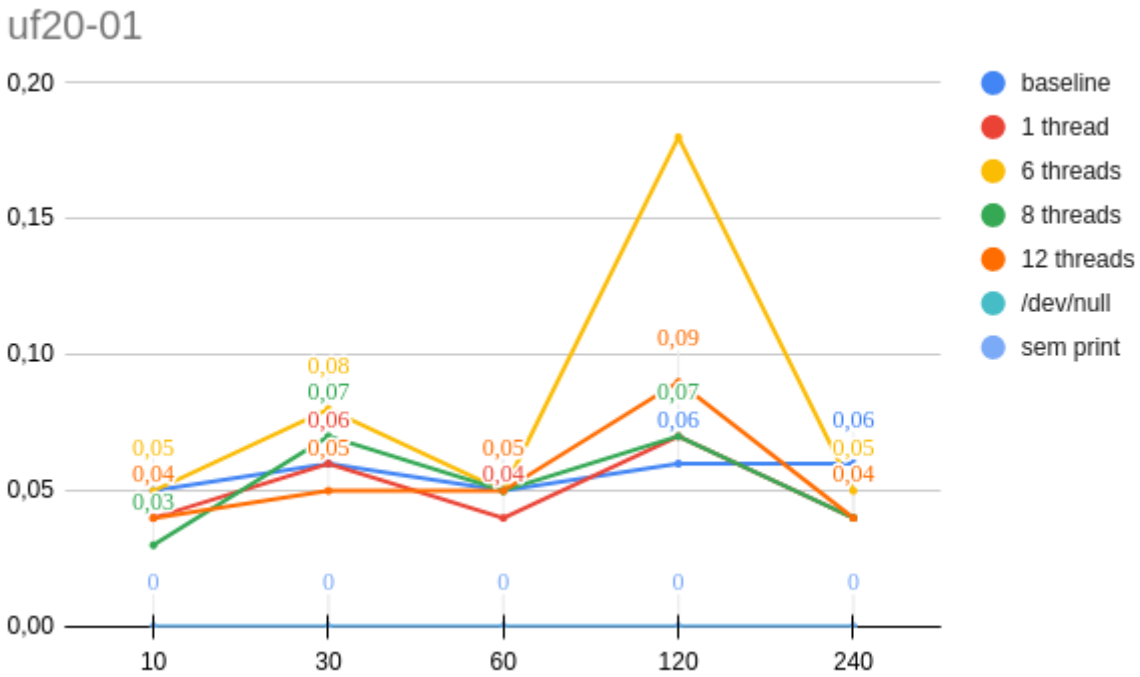


2.2.7) uf20-0462



2.2.8) uf20-01

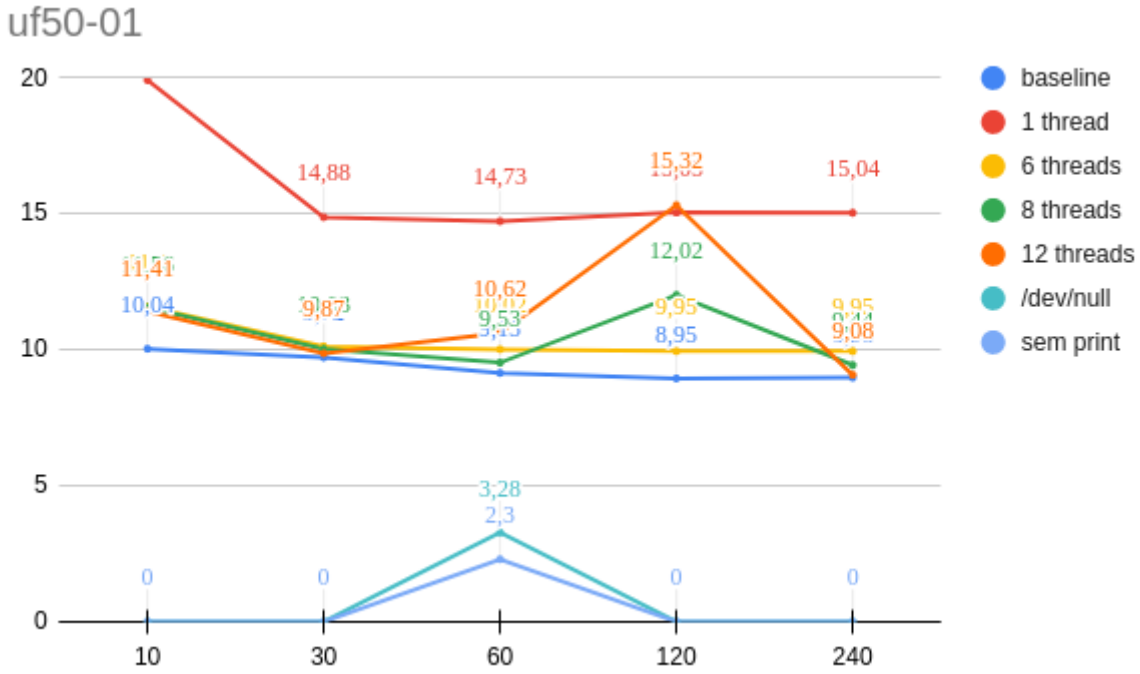
Um ponto legal de se notar é que a linha de baseline formou praticamente uma reta e o valor quase se manteve constante no decorrer dos testes com time limit. Como comparação houve pequenos valores que conseguiram chegar ao objetivo, que são algortimos de 8 e 12 threads no decorrer das evolução de time limits.



2.2.9) uf50-01

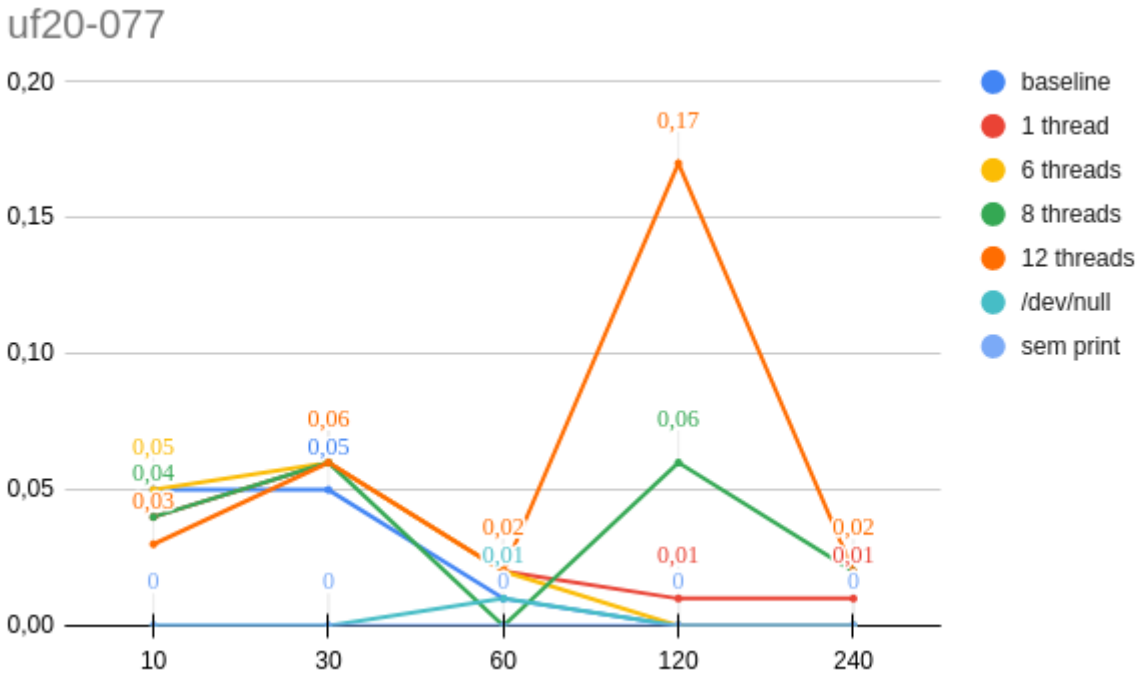
Para esses testes, é o tipo de teste onde não houve nenhuma resultado positivo, pois nenhum dos testes foram menores que o valor da baseline e também fica evidente que os testes com apenas uma thread não

chegou perto da baseline em nenhum dos testes no decorrer da coleta.



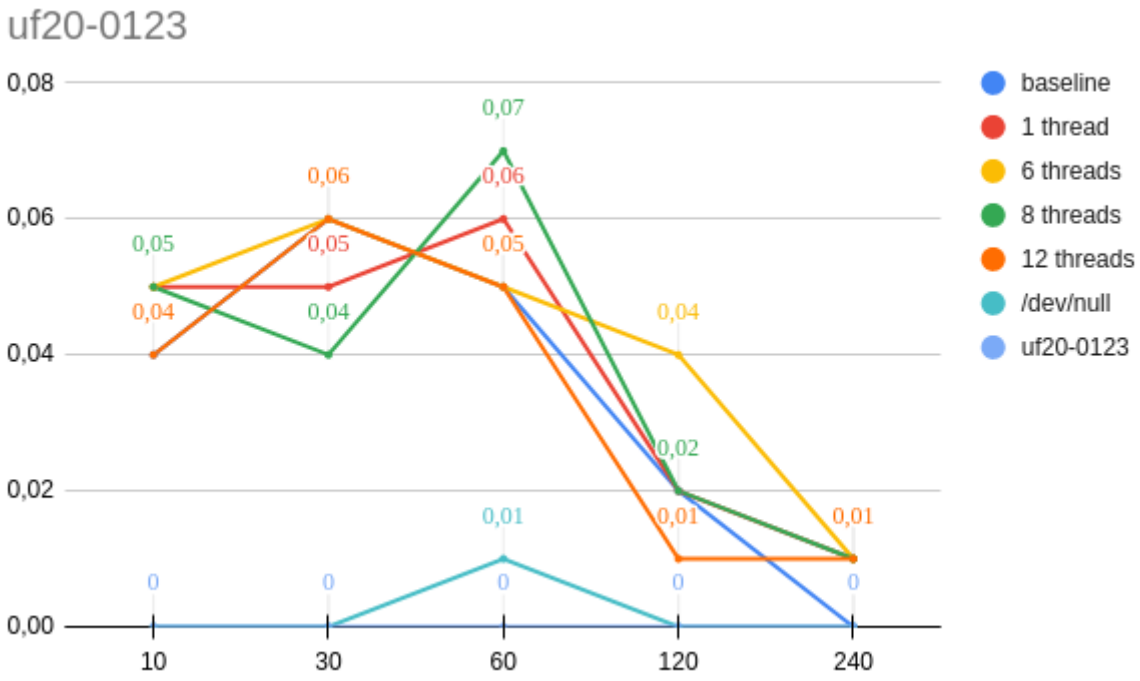
2.2.10) uf20-077

Para esse caso, é importante destacar que como sendo uma entrada pequena comparada a outros conjuntos de teste, quanto mais threads e entradas dentro do intervalo de teste mais discrepante o valor da baseline coletada.



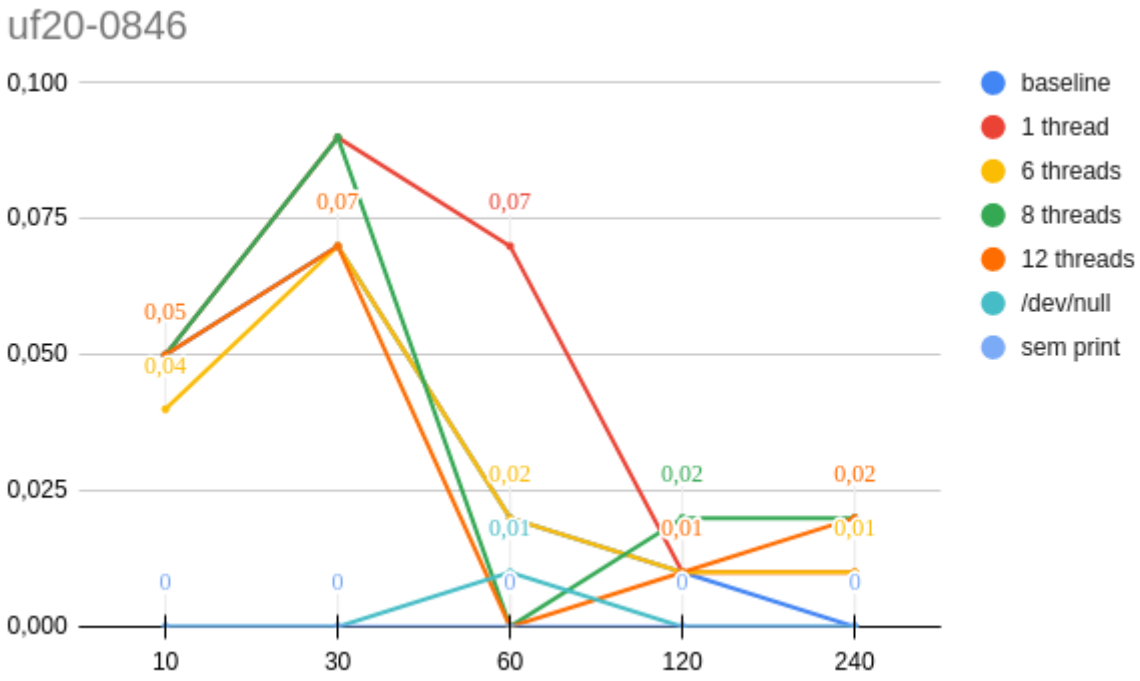
2.2.11) uf20-0123

Como os arquivos contém muito menos entradas que os outros do conjunto, é possível verificar que a maioria dos resultados não ultrapassaram e o valores são bem diferentes do valor estipulado pela baseline, mas mesmo assim é possível verificar que há resultados que conseguiram ser melhores que o valor base.



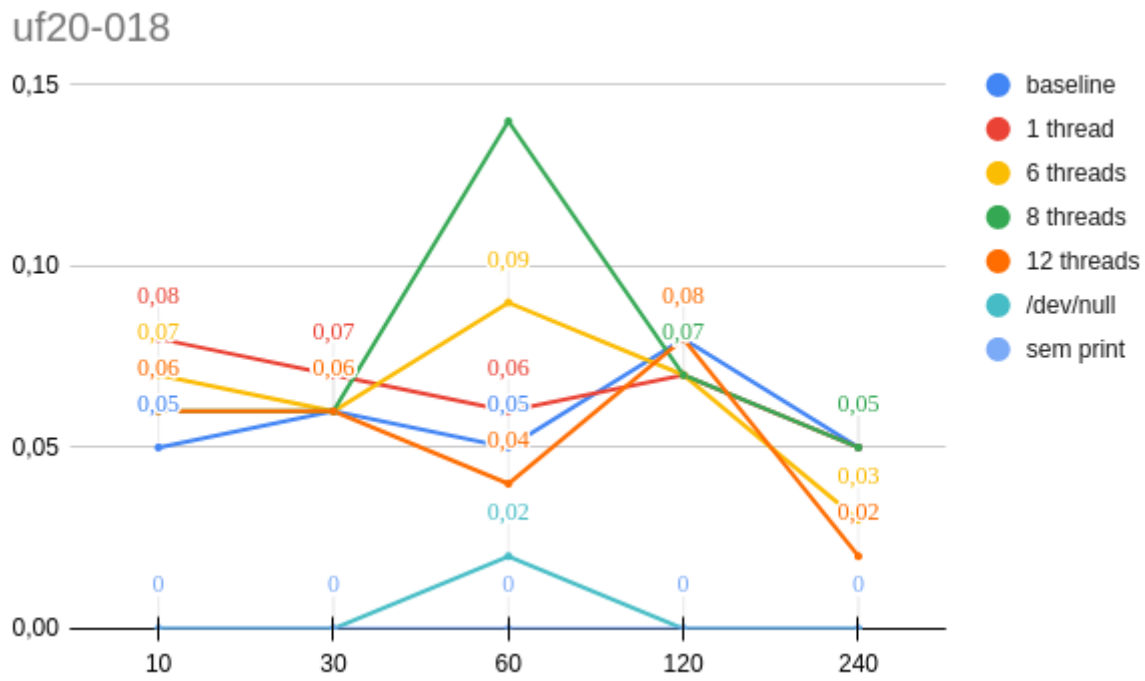
2.2.12) uf20-0846

Os resultados encontrados para esse tipo de entrada foram mais eficazes nos time limits de 10 segundos. Nos outros testes eles se manterem ou próximos ou superiores no tempo em referência a baseline.



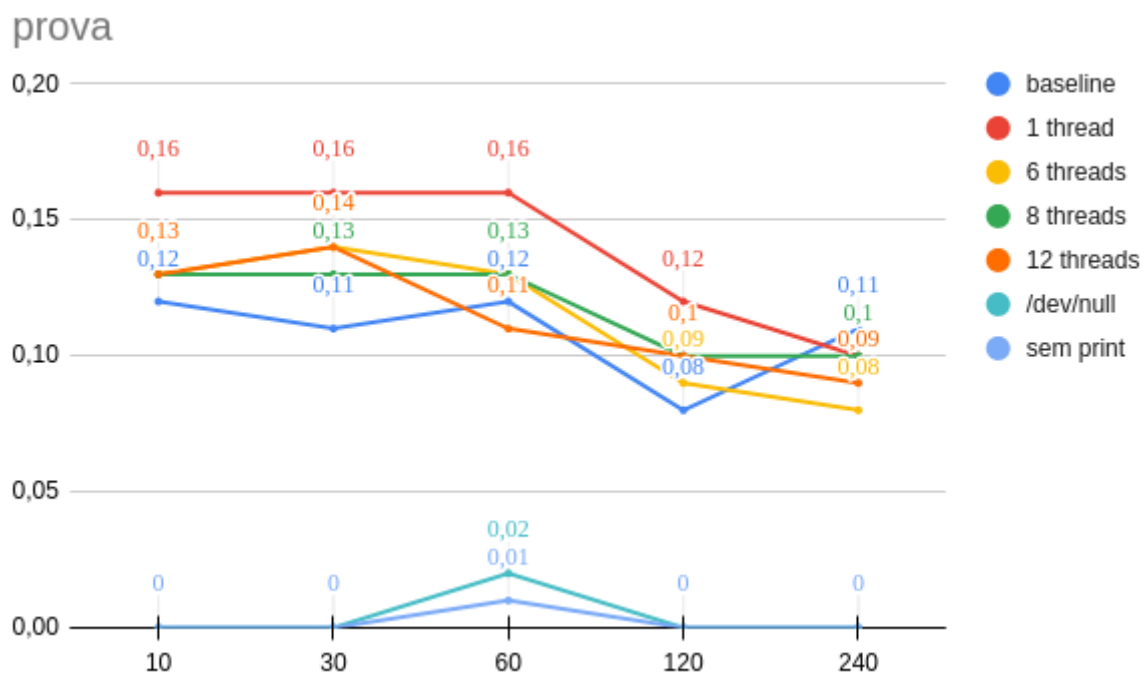
2.2.13) uf20-018

Já para esse conjunto de testes, os resultados envolvendo uma thread ainda continuaram ruins ao ponto de relação, já no gráfico fica evidente até o time limit 60 segundo, que os resultados conseguiram prosperar apenas a partir do time limit de 60 segundos sendo em 60 segundos, apenas 12 threads conseguiu ser melhor, já para os outro time limits (120, 240) todos resultados obtiveram resultado satisfatório.



2.2.14) prova

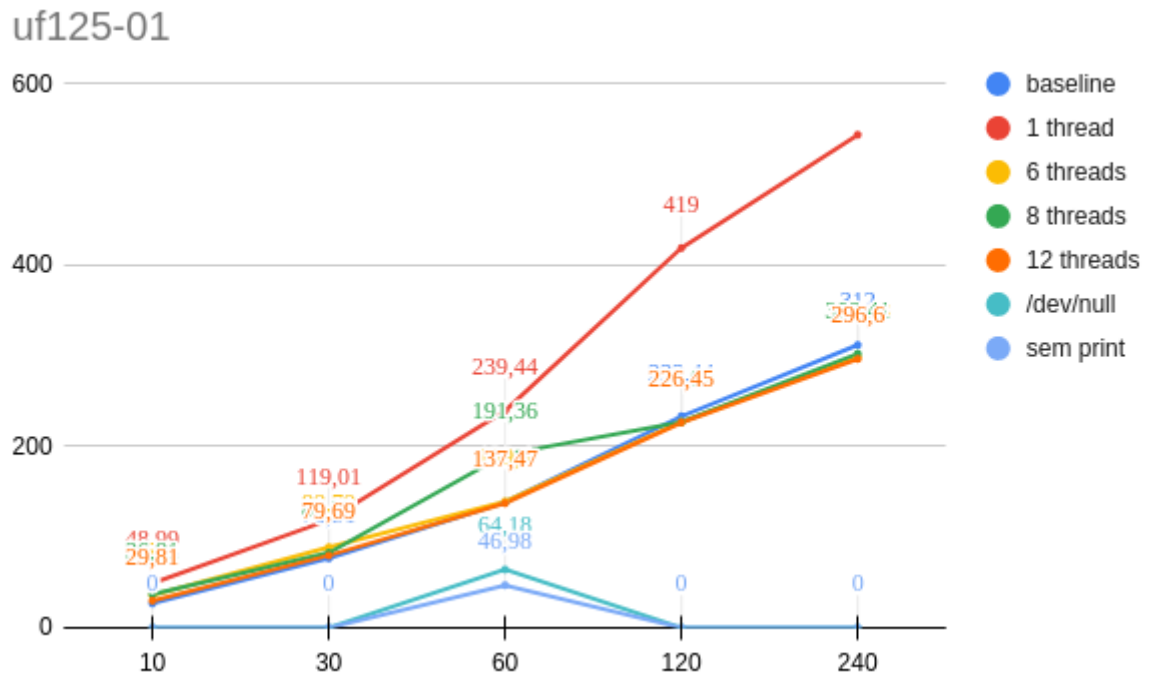
Para essa medição é possível verificar, que os resultados mais evidentes são com time limit 240. Onde todos os algoritmos conseguiram bater o tempo da baseline.



Entradas maiores

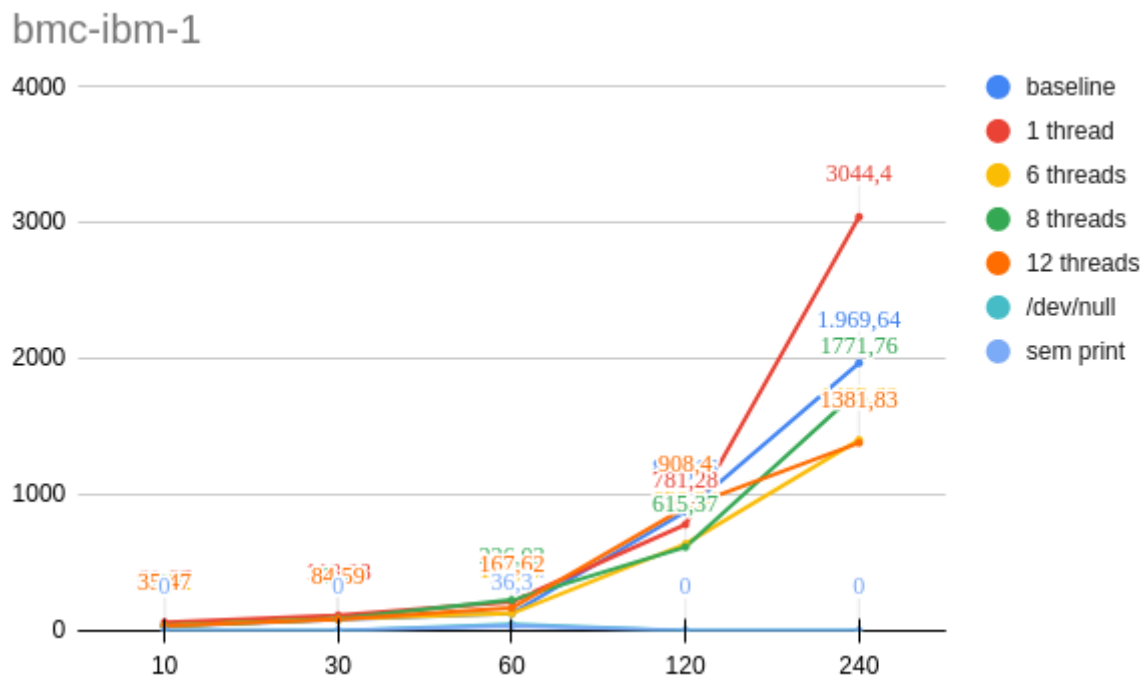
2.2.15) uf125-01

Nessa medição fica muito mais evidente que houve resultados bem parecidos com a baseline tirando a medição com 1 thread. Apesar de ter algumas medições que foram menores ainda sim a distância foi pouca da baseline atual para esse arquivo de teste.



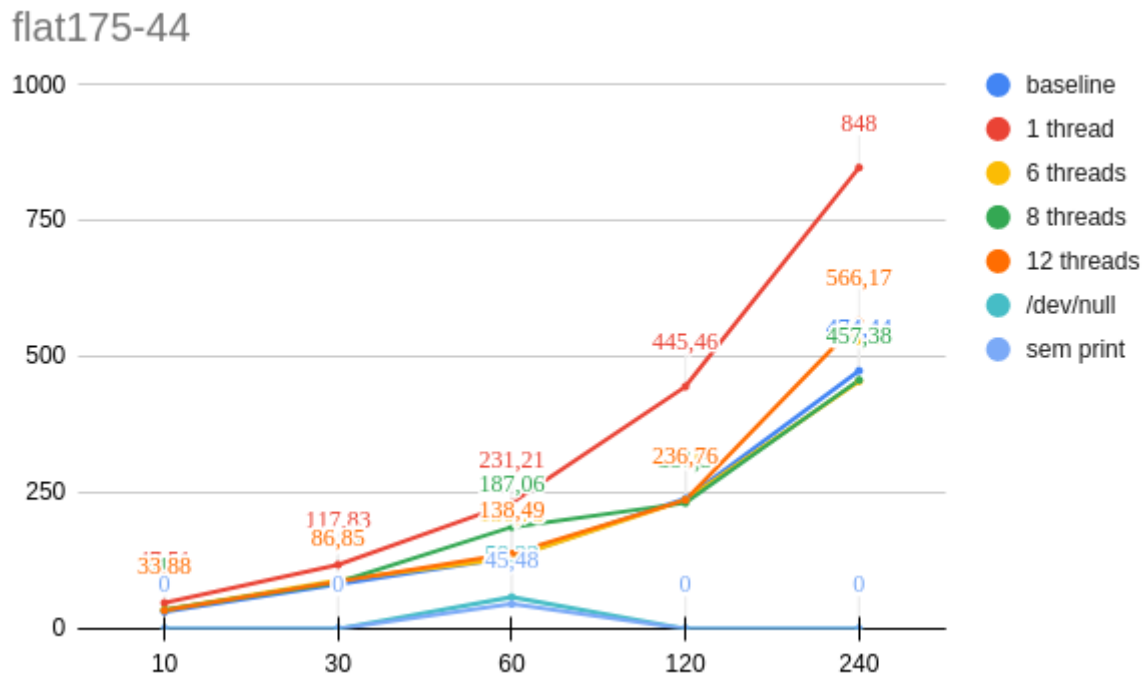
2.2.16) bmc-ibm-1

Os arquivos IBMs contém entradas medianas e tempos de execução medianos também, no bmc-ibm-1 os tempo de execução em relação a baseline foram bem parecidos, onde a partir do time limit 120 os tempos foram mais discrepantes, logo tendo um tempo maior que de baseline apenas para medições referentes apenas uma thread.



2.2.17) flat175-44

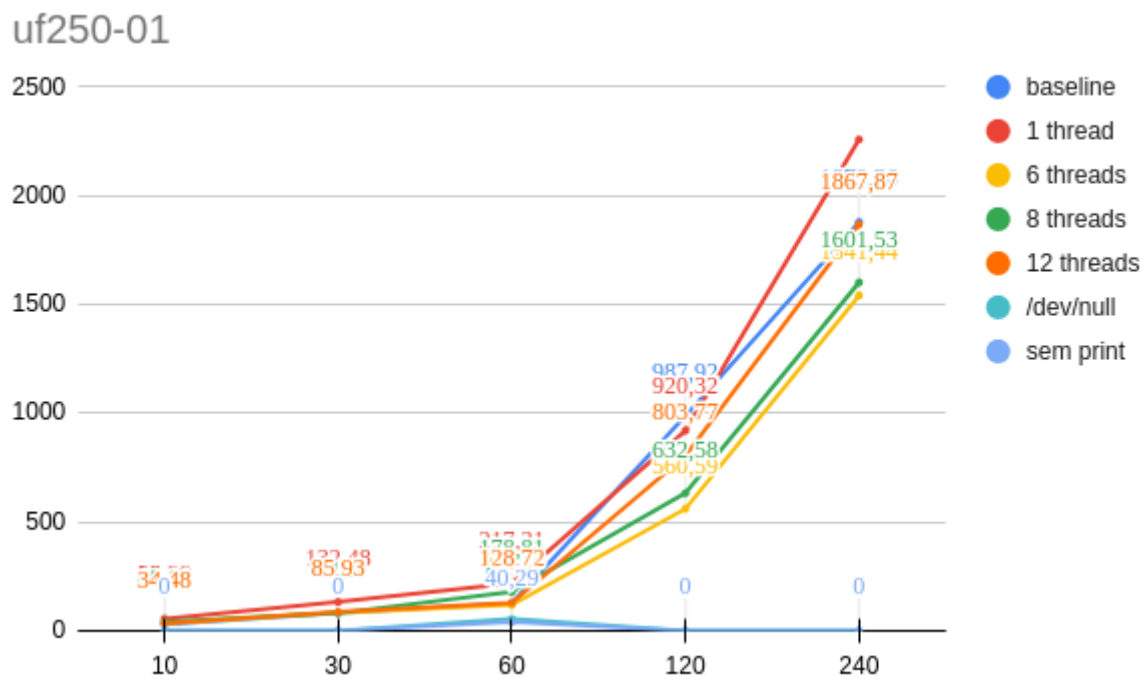
Nesse gráfico é possível ver um crescimento linear de cada uma das medições. Mas é possível notar que apenas os algorítimos executados com 6, 8 e 12 threads conseguiu alcançar com valores muito proximo a baseline para diferentes time limits, como é possível verificar também o algoritimo usando apenas uma thread não consegue se aproximar o valor da baseline atual sendo o mais distante do valor desejado.



2.2.18) uf250-01

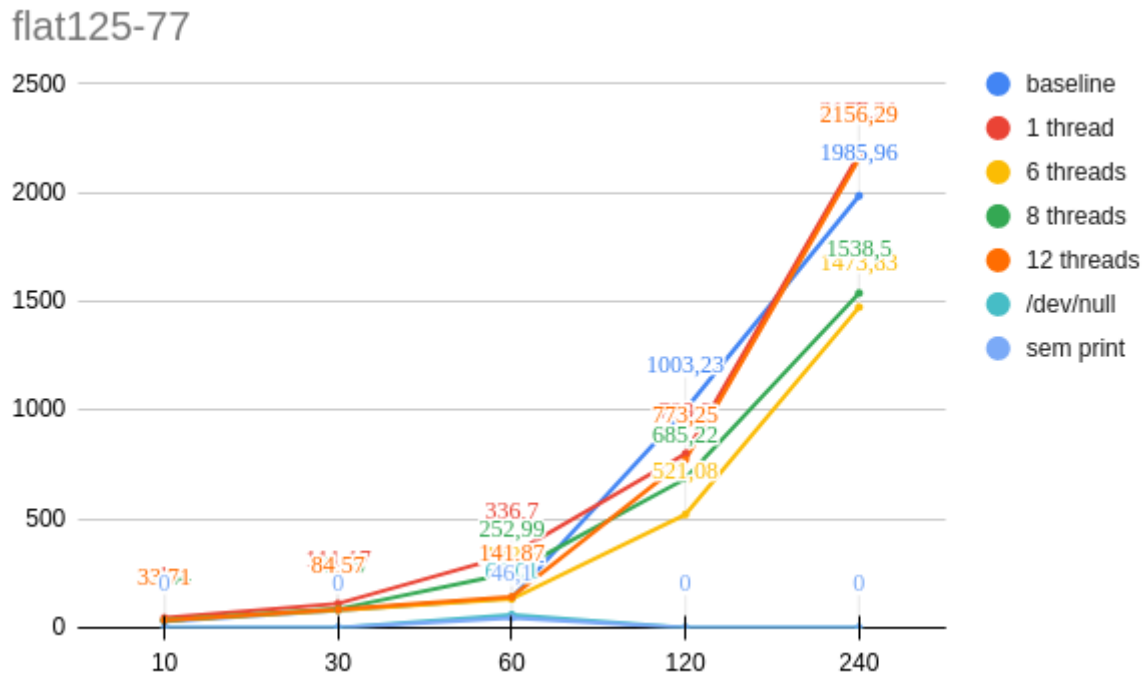
É possível visualizar que o resultados começam bem próximos, por mais que alguns dos resultados não tenham sido esperados. Com o crescimento do time limit os resultados começaram a tender um valor

melhor que o valor base, ocorrendo a partir do time limit de 60 segundos.



2.2.19) flat125-77

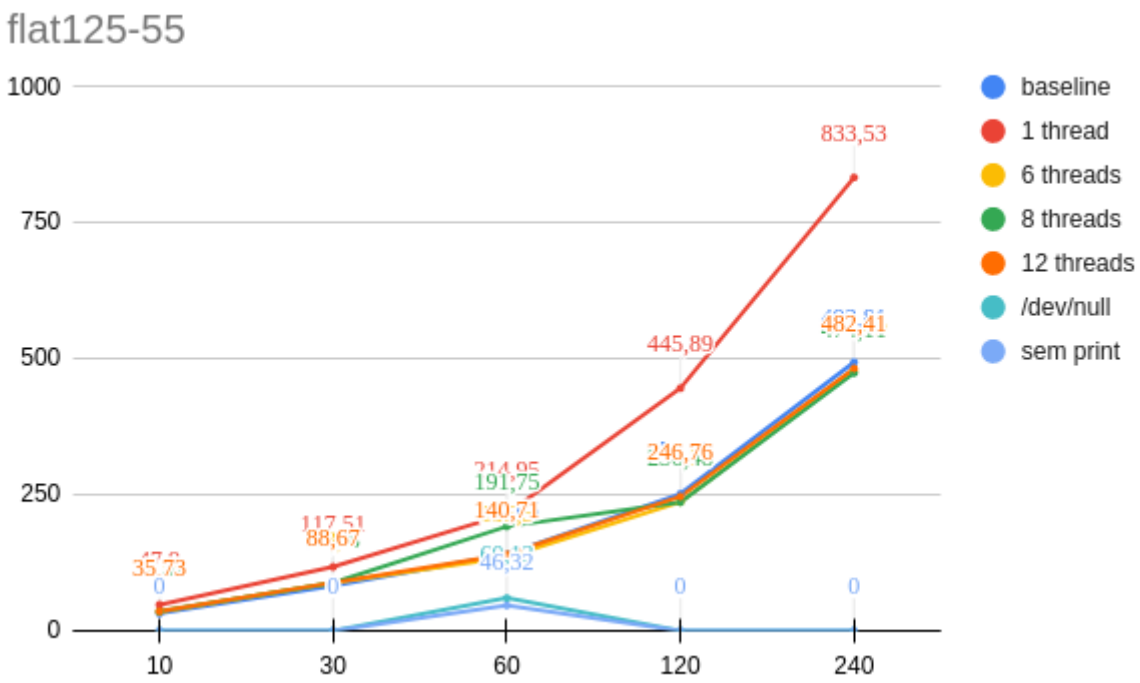
Para essa bateria de testes é possível ver que apesar dos resultados estarem bem perto da linha de baseline nos primeiros time limits, apenas a partir do time limit de 60 segundos que houve resultados mais visíveis de algorítimos que foram mais rápidos que o algorítimo sequencial.



2.2.20) flat125-55

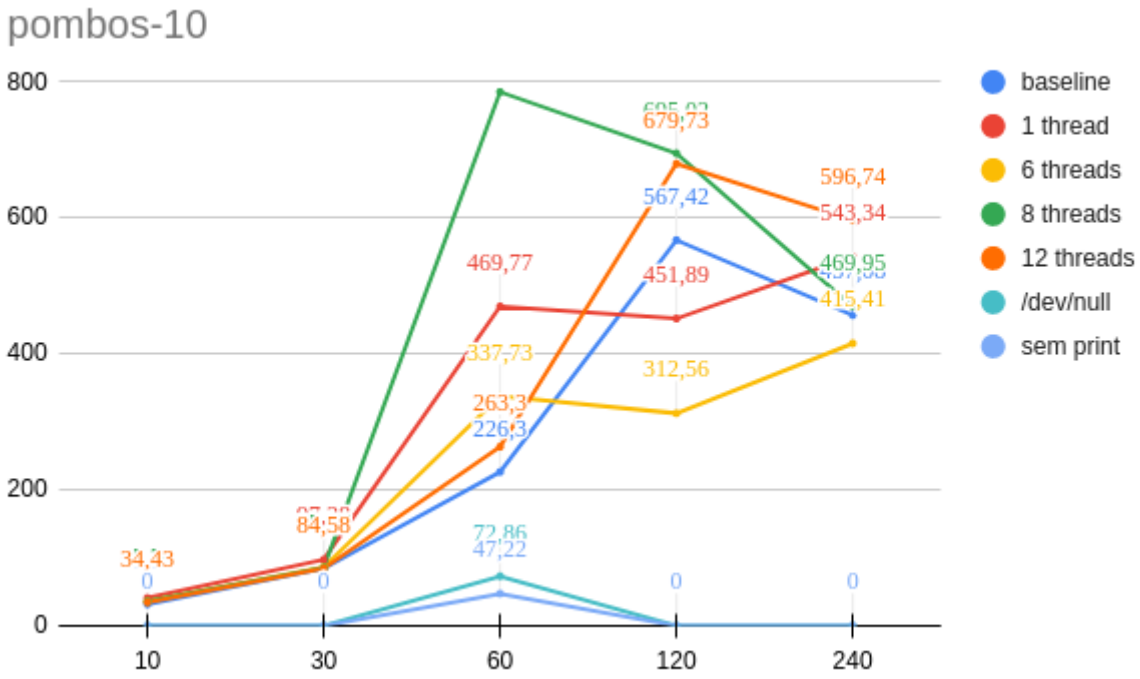
Para esses testes, apesar de não ser muito evidente os houve resultados acima de 60 segundo de time limits, mas com resultados muito próximo do de baseline, onde os algorítimos de 8 e 12 threads foram os

que mais se sobressaíram para esse intervalo.



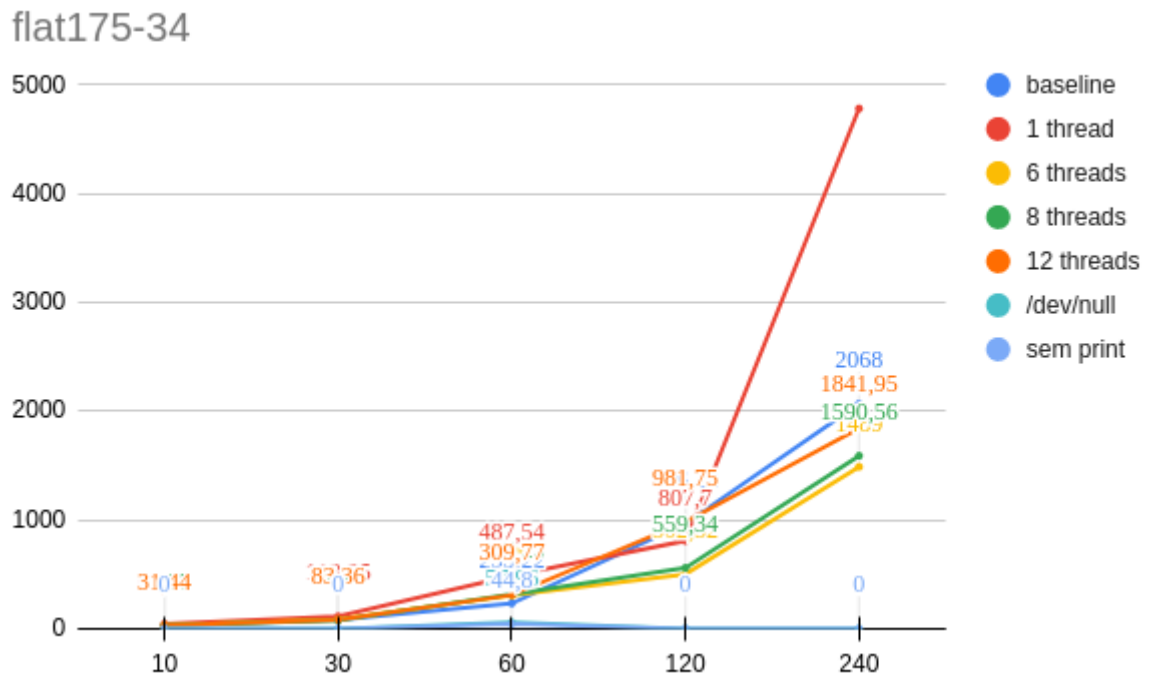
2.2.21) Pombos-10

Esse arquivo de testes é um dos maiores dentro do conjunto e durante os testes foram os que mais demoraram a computar. Na bateria de testes, não foi possível ter resultados tão bons relacionado a linha de base, os únicos resultados evidentes foram com time limits de 120 segundo com 6 e 12 threads.



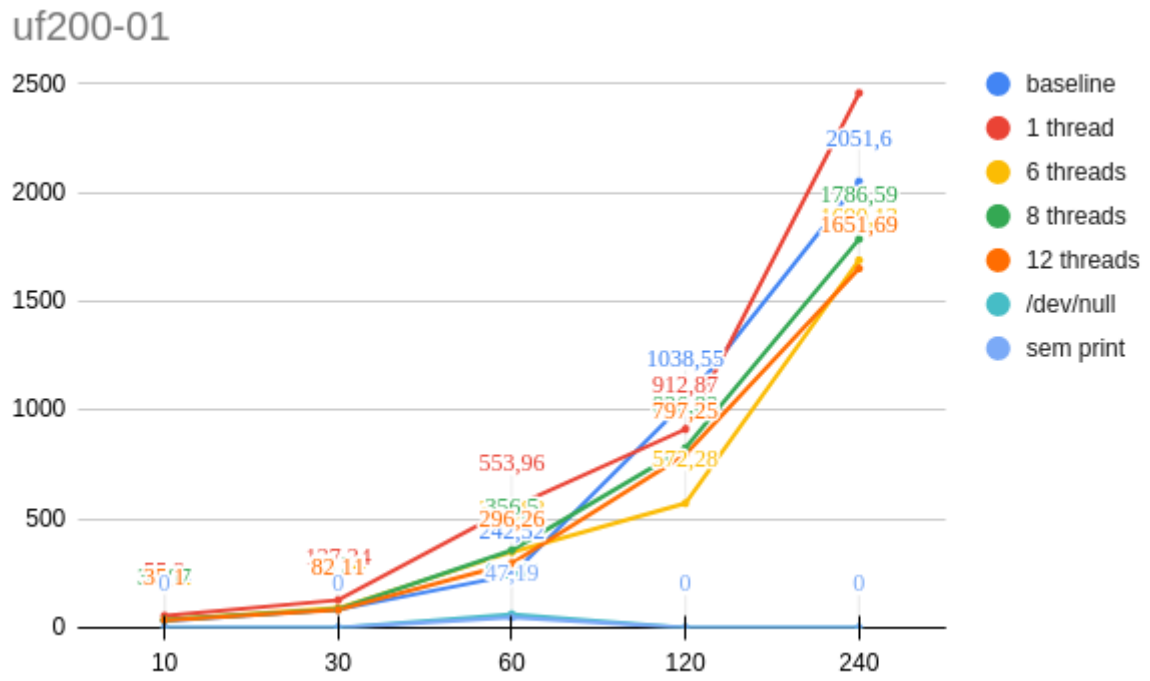
2.2.22) flat175-34

Novamente é possível ver uma linearidade durante o crescimento do time limit, há apenas um valor muito discrepante dessa linearidade que é o teste com 1 thread para 240 segundos de time limit, de resto os valores foram bem perto do valor de baseline, mas é possível verificar que os testes com 6, 8 e 12 thread a partir de 120 segundos de time limit.



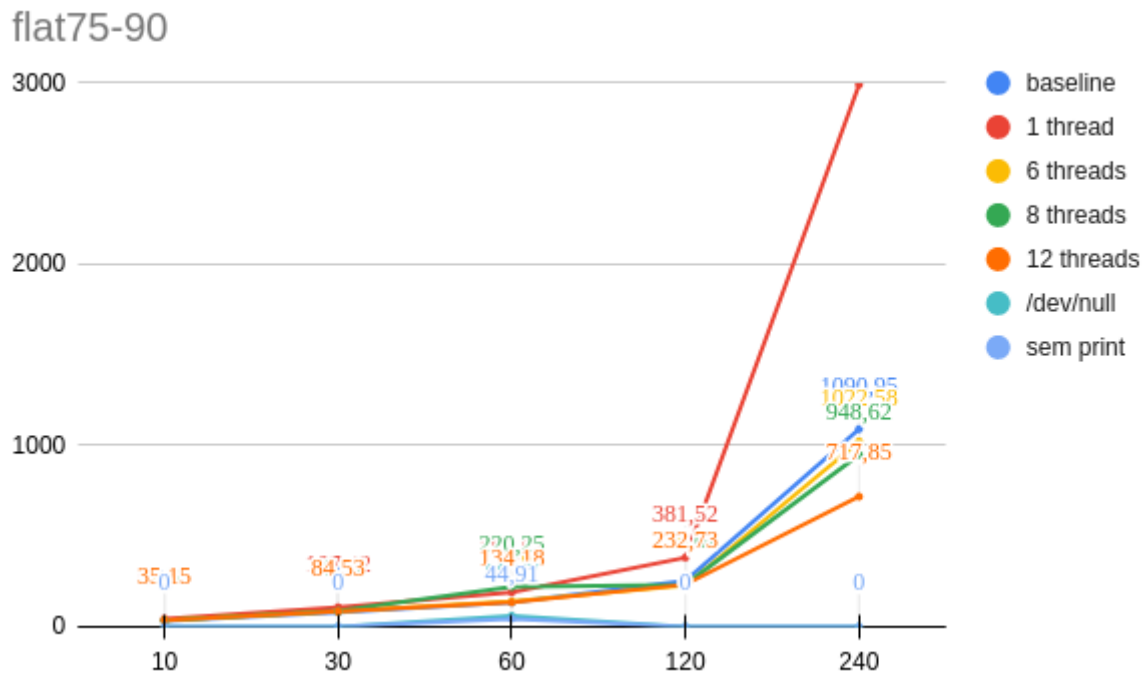
2.2.23) uf200-01

Nestes testes houve uma constância novamente, e é notável que a partir do time limit 120 segundo se sobressaiu melhor que a baseline, tirando o teste com 1 thread.



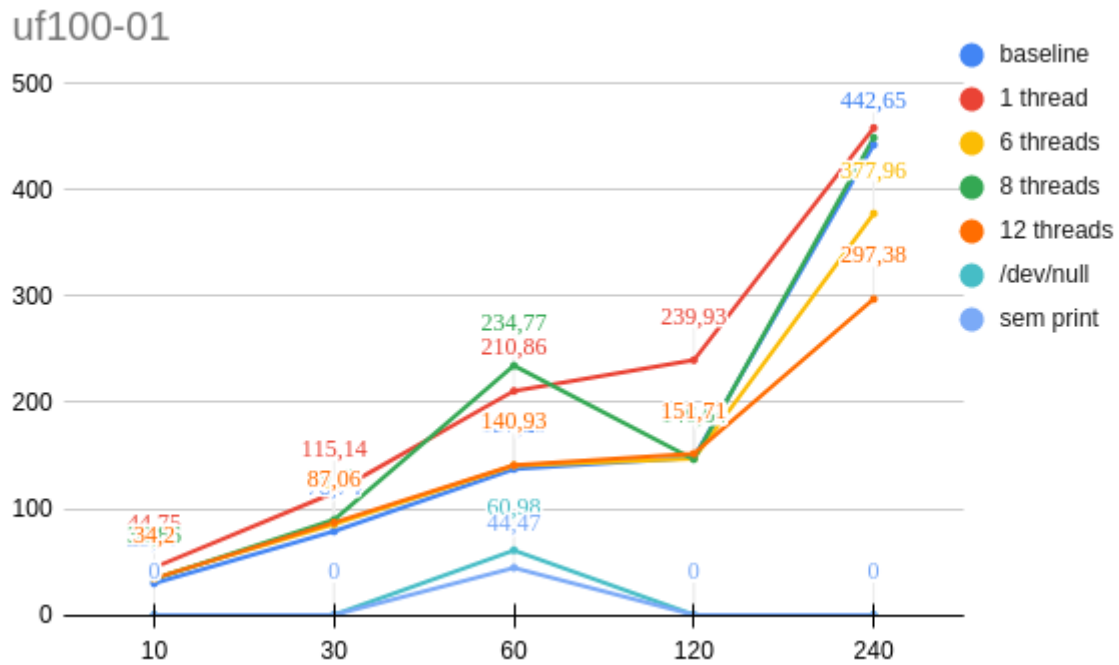
2.2.24) flat75-90

Mais uma vez testes as coletas chegaram em um valor muito proximo da baseline, mas não há apenas houve sucesso nas ultimas coletas com time limit 240 segundos, com 6, 8 e 12 threads, e para um thread o resultado foi muito maior que o valor de baseline estipulado.



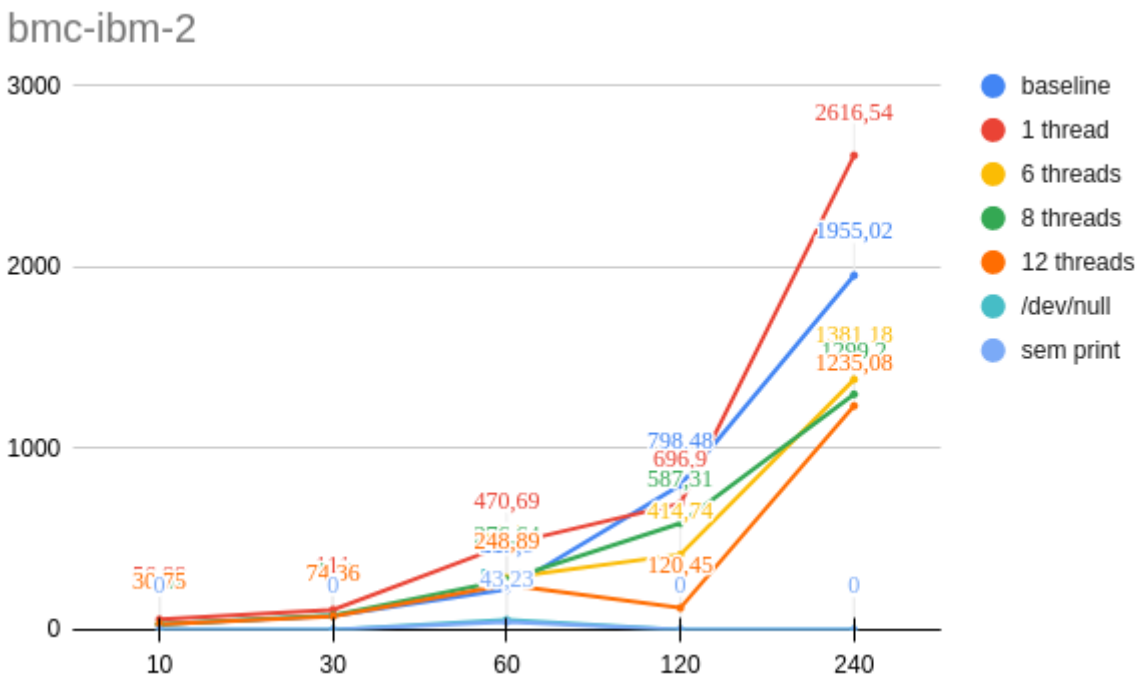
2.2.25) uf100-01

Os testes para essas coletas chegaram em um valor muito proximo da baseline, mas não há apenas houve sucesso nas ultimas coletas com time limit 240 segundos, com 6 e 12 threads, e para um thread o resultado foi muito mais proximo da baseline.

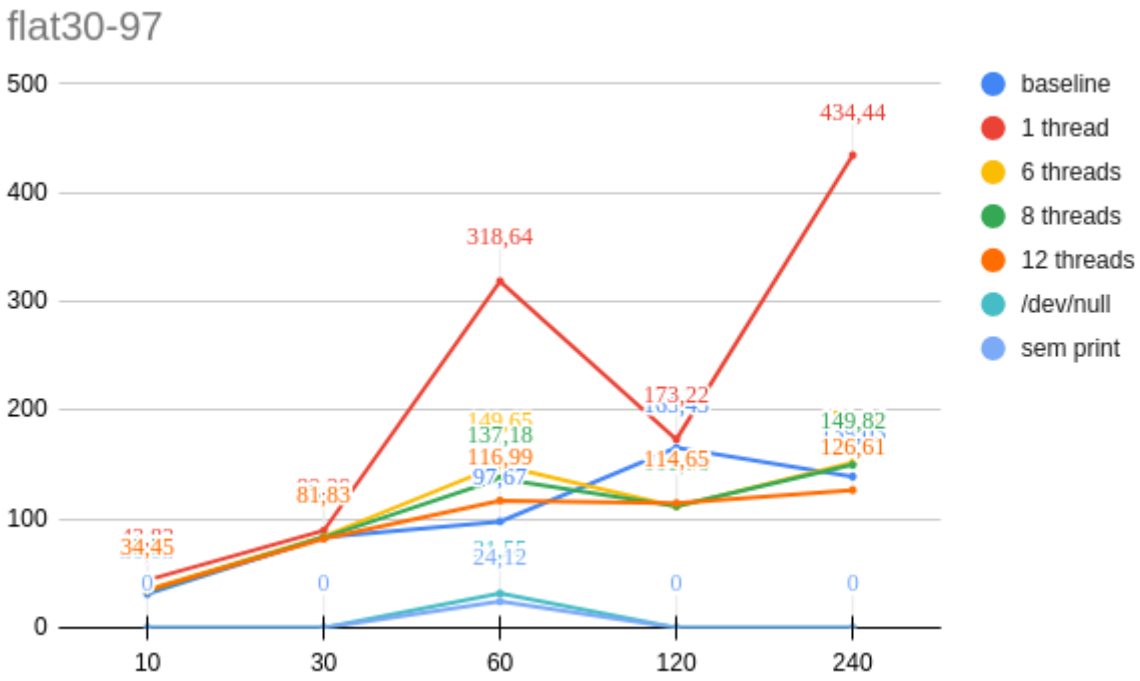


2.2.26) bmc-ibm-2

Como na primeira coleta de entrada para entradas do tipo "ibm", no começo das coletas,os resultados se manterem muito próximo da baseline, e apenas depois de time limits grandes com entradas maiores que fica perceptível os resultados melhores que a linha de base estipulada.

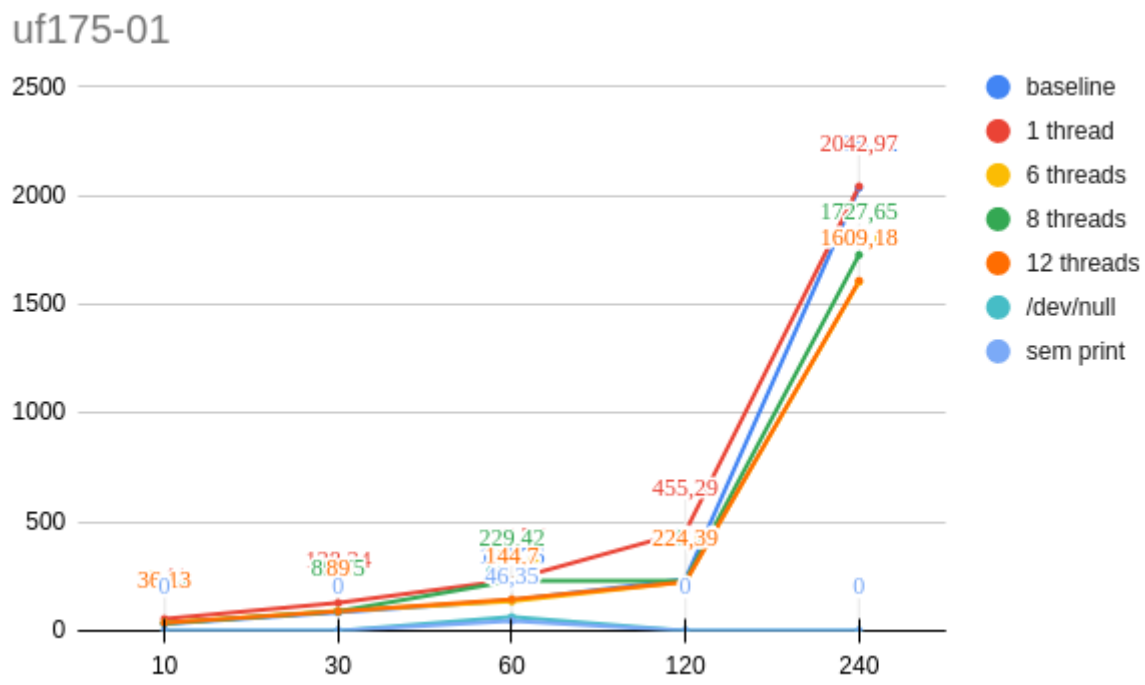


2.2.27) flat30-97



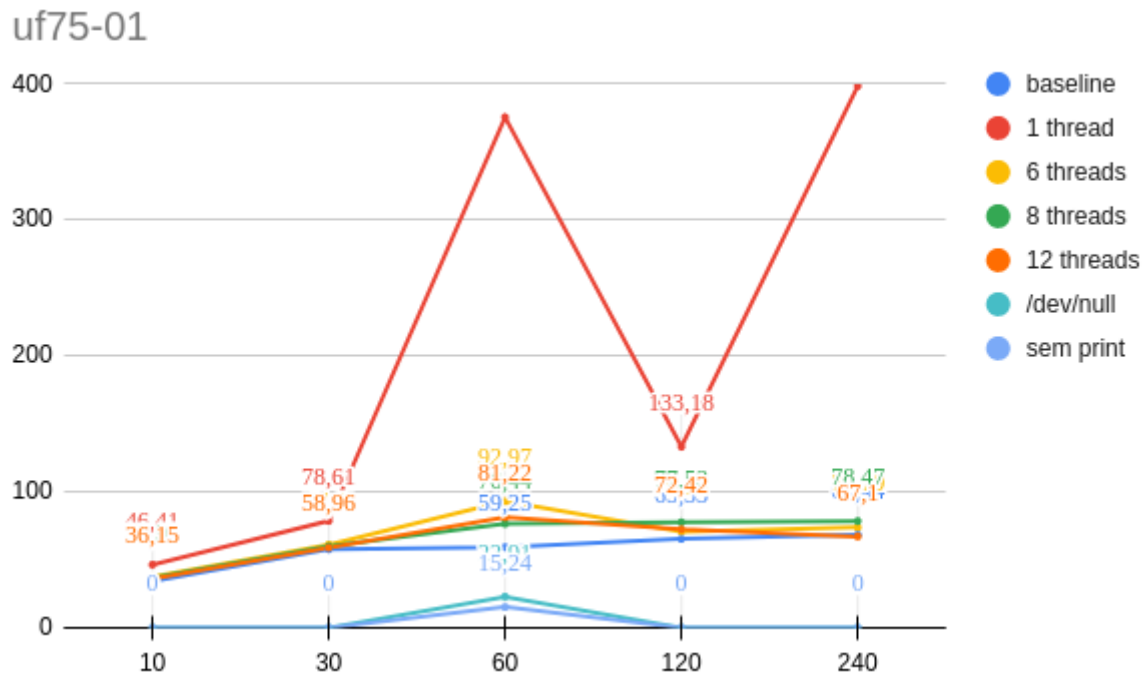
2.2.28) uf175-01

Os testes para essas coletas chegaram em um valor muito proximo da baseline, mas não há apenas houve sucesso nas ultimas coletas com time limit 240 segundos, com 6 e 12 threads, e para um thread o resultado foi muito mais proximo da baseline.



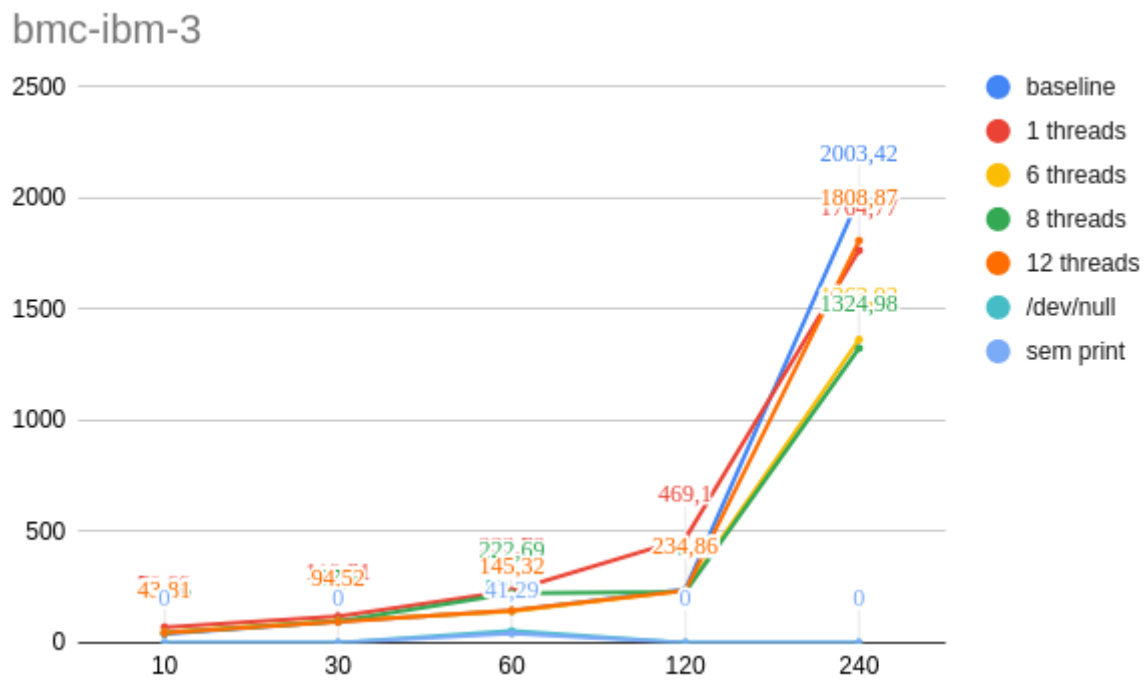
2.2.29) uf75-01

Para esse teste, os testes envolvendo 1 thread foram os piores dentro do intervalo de teste. Os outros se mantiveram perto da linha de base, e apena na ultima iteração com 240 segundos de timeline gerado e 12 threads que obteve um resultado positivo em relação a baseline.



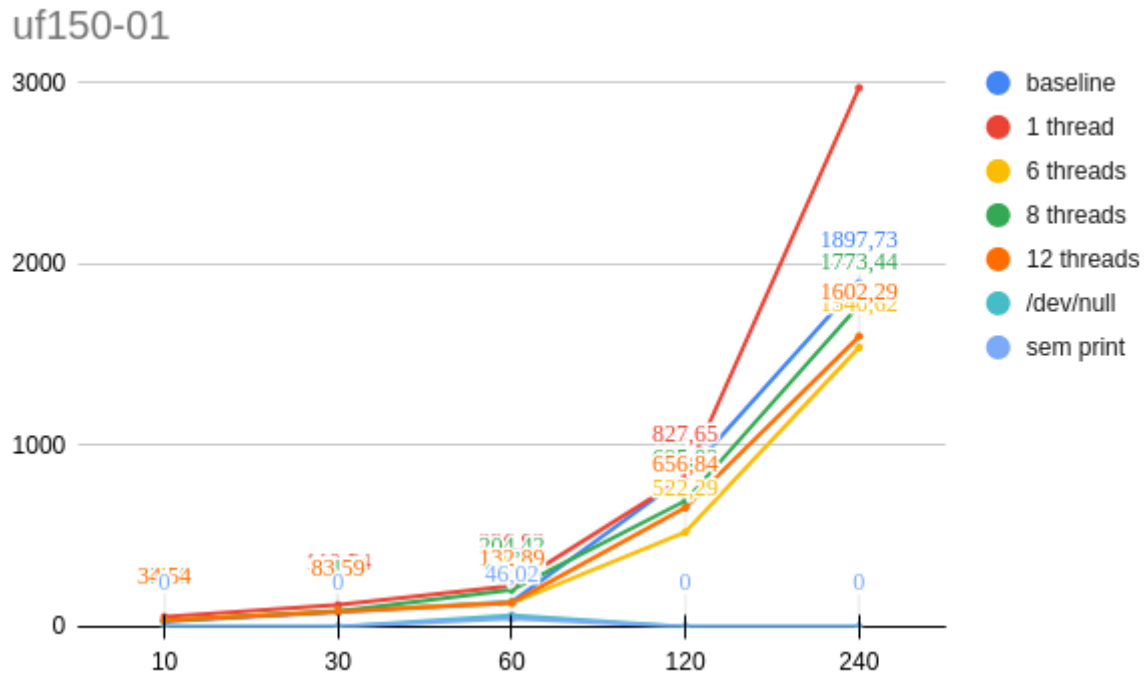
2.2.30) bmc-ibm-3

Novamente aqui temos resultados dos testes envolvendo arquivos ibm com entradas maiores. Nele é possível verificar que apenas time limits gerados acima de 120 segundos tiveram bons resultados em relação a linha de base.



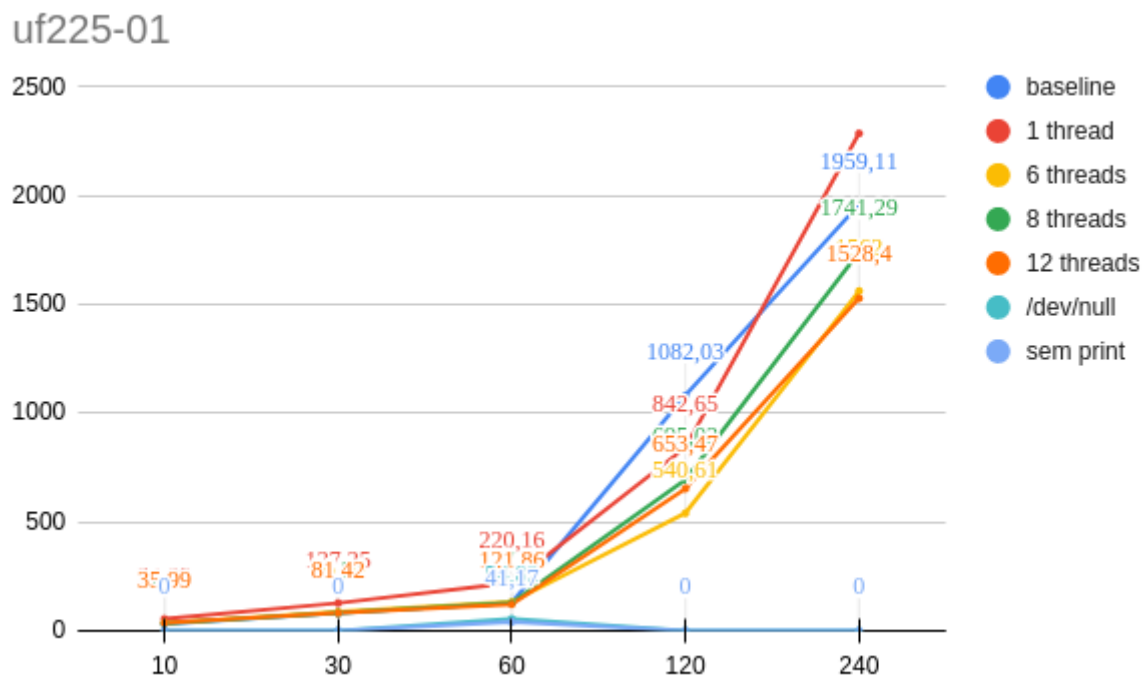
2.2.31) uf150-01

Nessa medições é possível verificar que apenas time limits gerados acima de 60 segundos tiveram bons resultados em relação a linha de base.



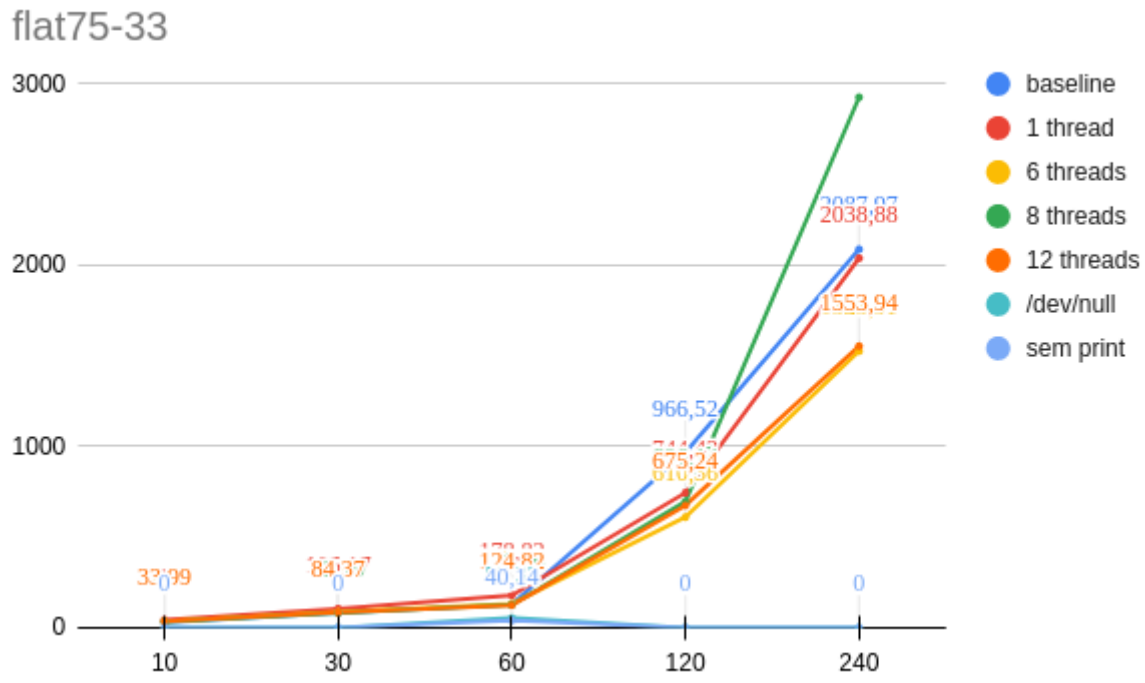
2.2.32) uf225-01

Nessa medições é possível verificar que apenas time limits gerados acima de 30 segundos tiveram bons resultados em relação a linha de base. Tendo discrepância apenas para uma thread no time limit 240.



2.2.33) uf75-33

Para esse teste, os testes envolvendo 8 thread foram os piores no ultimo intervalo de teste dos casos. Os outros se mantiveram perto da linha de base, e apena na ultima iteração com 120 segundos de timeline gerado e 12 threads que obteve um resultado positivo em relação a baseline.



Conclusão

É possível verificar através das análises dos gráficos que para entradas menores o algorítimo mais recente desenvolvido pela dupla tem menor desempenho, pois este usa números muito grandes para cada bloco na hora de processar os fulls e flips, o que pode ser um empecilho na hora de computar entradas pequenas. Já

para entradas maiores como é o caso dos bcm-ibm, é possível verificar que temos uma linha de tendência para essas entradas que tendem a ser uma parabola, geralmente nos primeiros timelimits os valores ficam bem próximos da baseline, mas a partir do timelimit 120 segundos, fica bastante evidente que com 6,8 e 12 thread os algoritmos sobressaem bem melhor que o da baseline. Logo resolvemos escolher esse tipo de abordagem de análise para servir como prova de que para entradas maiores o estrátegia tende a ser melhor e foi possível comprovar isso através dos testes. É importante destacar também que o conceito de bloco para nosso algoritmo é fixo contendo 12000 flips para ser processado a cada iteração dentro da thread.