

Alunos:

- João Vítor Morandi Lemos / 160010195
- José Aquiles Guedes de Rezende / 160010331

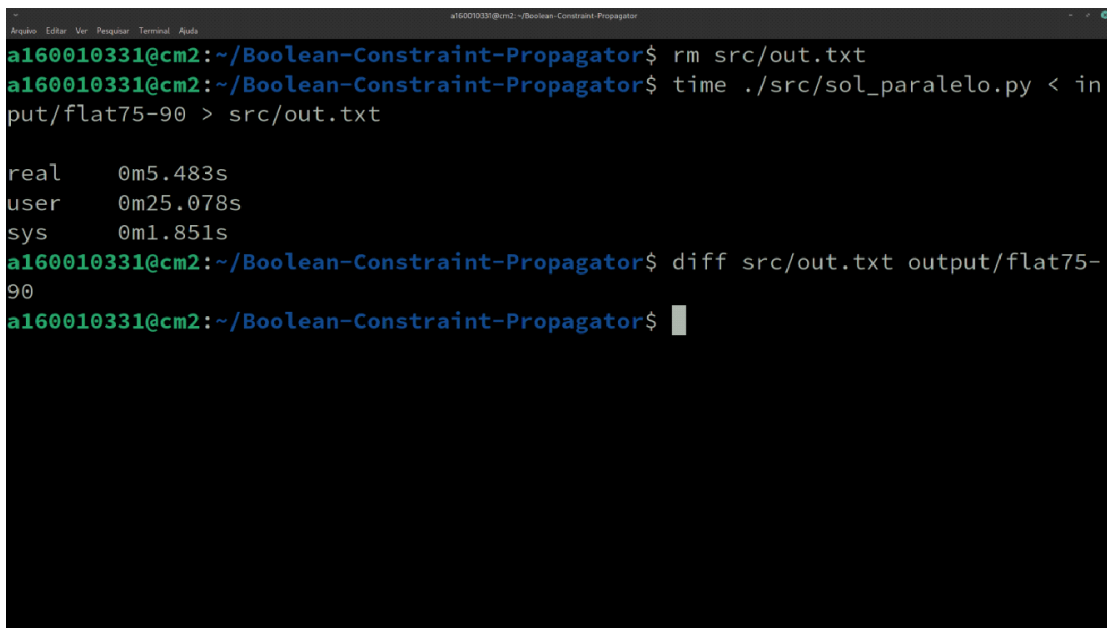
Desenvolvimento

Fizemos inicialmente uma solução sequencial (arquivo "sol.py"). Depois disso, focamos em usar paralelismo para tornar a solução mais eficiente (arquivo "sol_paralelo.py").

Reflexão

Quanto tempo a sua implementação demora para os casos fornecidos?

flat75-90 com 12 processadores: 5.483s



```
a160010331@cm2: ~/Boolean-Constraint-Propagator
a160010331@cm2:~/Boolean-Constraint-Propagator$ rm src/out.txt
a160010331@cm2:~/Boolean-Constraint-Propagator$ time ./src/sol_paralelo.py < input/flat75-90 > src/out.txt

real    0m5.483s
user    0m25.078s
sys     0m1.851s
a160010331@cm2:~/Boolean-Constraint-Propagator$ diff src/out.txt output/flat75-90
a160010331@cm2:~/Boolean-Constraint-Propagator$
```

flat175-34 com 12 processadores: 3m7.303s (solução do professor: 1m20.329s)

```

a160010331@cm2: ~/Boolean-Constraint-Propagator
bmc-ibm-3    flat30-36    prova      uf175-01    uf20-0123   uf20-0819   uf50-01
flat125-55   flat30-97    simples    uf200-01    uf20-018    uf20-0846   uf75-01
flat125-77   flat75-33    uf100-01   uf20-01     uf20-0216   uf20-095

a160010331@cm2:~/Boolean-Constraint-Propagator/input$ cd ..
a160010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < flat175-34 >out.txt
-bash: flat175-34: No such file or directory

real    0m0.004s
user    0m0.003s
sys     0m0.000s
a160010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < input/flat175-34 >out.txt
real    3m7.303s
user    23m21.267s
sys     0m32.993s
a160010331@cm2:~/Boolean-Constraint-Propagator$ rm out.txt
a160010331@cm2:~/Boolean-Constraint-Propagator$ time ./simple-gsat-io-generator < input/flat175-34 >out.txt
real    1m20.329s
user    0m26.611s
sys     0m0.293s
a160010331@cm2:~/Boolean-Constraint-Propagator$

```

Como é feita a otimização para paralelizar o algoritmo?

R: utilizamos processos no lugar de threads, que fazem uso de memória compartilhada, o que diminui um pouco a eficiência mas dessa forma o resultado sai corretamente.

Como o desempenho é alterado rodando o programa com 2, 4, 6, 8 e 12 threads?

flat75-90 com 3 processos: 10.015s

flat75-90 com 6 processos: 5.720s

flat75-90 com 12 processos: 5.483s

```

a160010331@cm2:~/Boolean-Constraint-Propagator$ rm src/out.txt
a160010331@cm2:~/Boolean-Constraint-Propagator$ time ./src/sol_paralelo.py 6 < input/flat75-90 > src/out.txt
real    0m5.720s
user    0m19.687s
sys     0m1.496s
a160010331@cm2:~/Boolean-Constraint-Propagator$ rm src/out.txt
a160010331@cm2:~/Boolean-Constraint-Propagator$ time ./src/sol_paralelo.py 3 < input/flat75-90 > src/out.txt
real    0m10.015s
user    0m17.588s
sys     0m1.783s
a160010331@cm2:~/Boolean-Constraint-Propagator$

```

O desempenho é modificado quando o arquivo de saída é redirecionado para /dev/null?

flat175-34 com 12 processadores: 3m7.303s

flat175-34 com 12 processadores redirecionando para /dev/null: 2m13.490s

```
al60010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < input/flat175-34 >/dev/null

real    2m12.672s
user    23m7.380s
sys     0m33.447s
al60010331@cm2:~/Boolean-Constraint-Propagator$ nano src/sol_paralelo.py
al60010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < input/flat175-34 >/dev/null

real    2m13.490s
user    23m17.656s
sys     0m33.236s
al60010331@cm2:~/Boolean-Constraint-Propagator$
```

E quando a impressão é comentada?

flat175-34 com 12 processadores: 3m7.303s

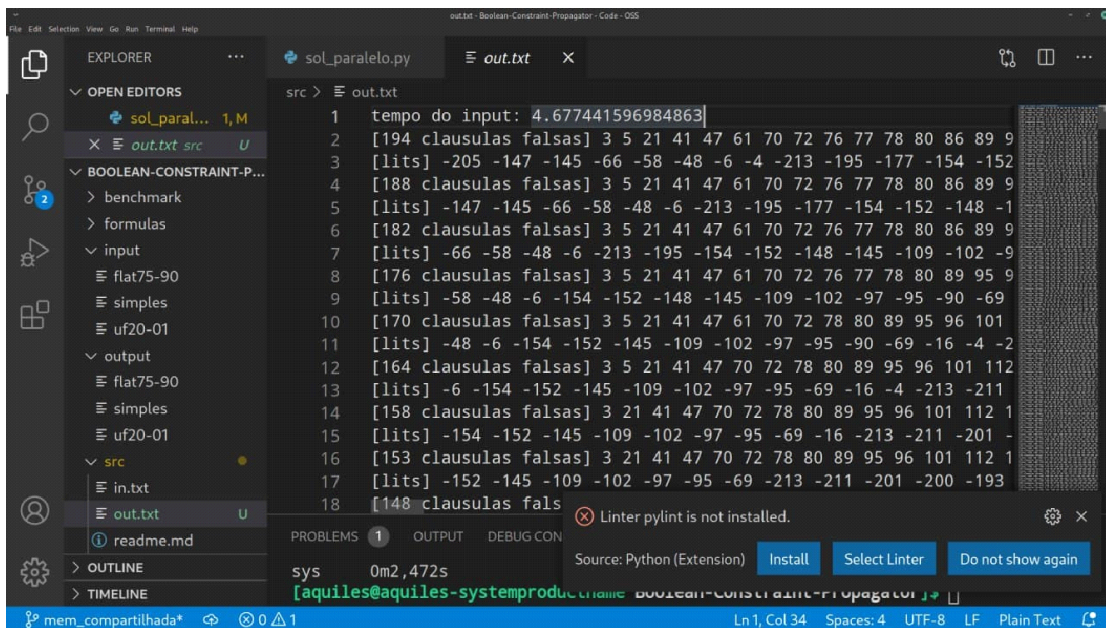
flat175-34 com 12 processadores com prints comentados: 2m12.250s

```
al60010331@chococino.naquadah.com.br's password:
=====
BEM VINDO
Máquinas disponíveis no momento:
cm2
cm4
=====
Last login: Sun Apr 11 19:04:17 2021 from 189.6.31.129
al60010331@chococino:~$ ssh cm2
al60010331@cm2's password:
Last login: Sun Apr 11 19:04:35 2021 from 164.41.127.67
al60010331@cm2:~$ cd Boolean-Constraint-Propagator/
al60010331@cm2:~/Boolean-Constraint-Propagator$ nano src/sol_paralelo.py
al60010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < input/flat
flat125-55 flat175-34 flat30-36 flat75-33
flat125-77 flat175-44 flat30-97 flat75-90
al60010331@cm2:~/Boolean-Constraint-Propagator$ time src/sol_paralelo.py < input/flat175-34

real    2m12.250s
user    23m14.500s
sys     0m32.925s
al60010331@cm2:~/Boolean-Constraint-Propagator$
al60010331@cm2:~/Boolean-Constraint-Propagator$
```

Qual o impacto da leitura do arquivo de entrada no tempo global?

flat75-90 com 12 processadores (tempo de leitura do input): 4.677s



```
src > out.txt
1 tempo do input: 4.677441596984863
2 [194 clausulas falsas] 3 5 21 41 47 61 70 72 76 77 78 80 86 89 9
3 [lits] -205 -147 -145 -66 -58 -48 -6 -4 -213 -195 -177 -154 -152
4 [188 clausulas falsas] 3 5 21 41 47 61 70 72 76 77 78 80 86 89 9
5 [lits] -147 -145 -66 -58 -48 -6 -213 -195 -177 -154 -152 -148 -1
6 [182 clausulas falsas] 3 5 21 41 47 61 70 72 76 77 78 80 86 89 9
7 [lits] -66 -58 -48 -6 -213 -195 -154 -152 -148 -145 -109 -102 -9
8 [176 clausulas falsas] 3 5 21 41 47 61 70 72 76 77 78 80 89 95 9
9 [lits] -58 -48 -6 -154 -152 -148 -145 -109 -102 -97 -95 -90 -69
10 [170 clausulas falsas] 3 5 21 41 47 61 70 72 78 80 89 95 96 101
11 [lits] -48 -6 -154 -152 -145 -109 -102 -97 -95 -90 -69 -16 -4 -2
12 [164 clausulas falsas] 3 5 21 41 47 70 72 78 80 89 95 96 101 112
13 [lits] -6 -154 -152 -145 -109 -102 -97 -95 -69 -16 -4 -213 -211
14 [158 clausulas falsas] 3 21 41 47 70 72 78 80 89 95 96 101 112 1
15 [lits] -154 -152 -145 -109 -102 -97 -95 -69 -16 -213 -211 -201 -
16 [153 clausulas falsas] 3 21 41 47 70 72 78 80 89 95 96 101 112 1
17 [lits] -152 -145 -109 -102 -97 -95 -69 -213 -211 -201 -200 -193
18 [148 clausulas falsas]
```

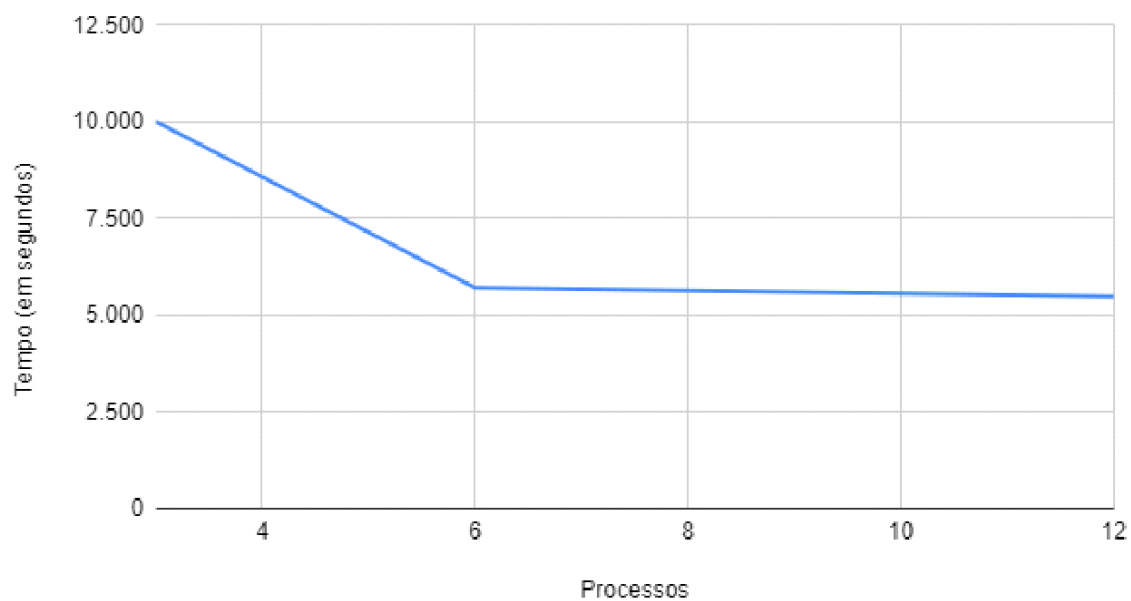
A partir de qual tamanho de fórmula a paralelização faz diferença?

R: a partir do arquivo flat75-90 já percebe-se o código com paralelismo rodando bem mais rápido do que o código sequencial

Entregando

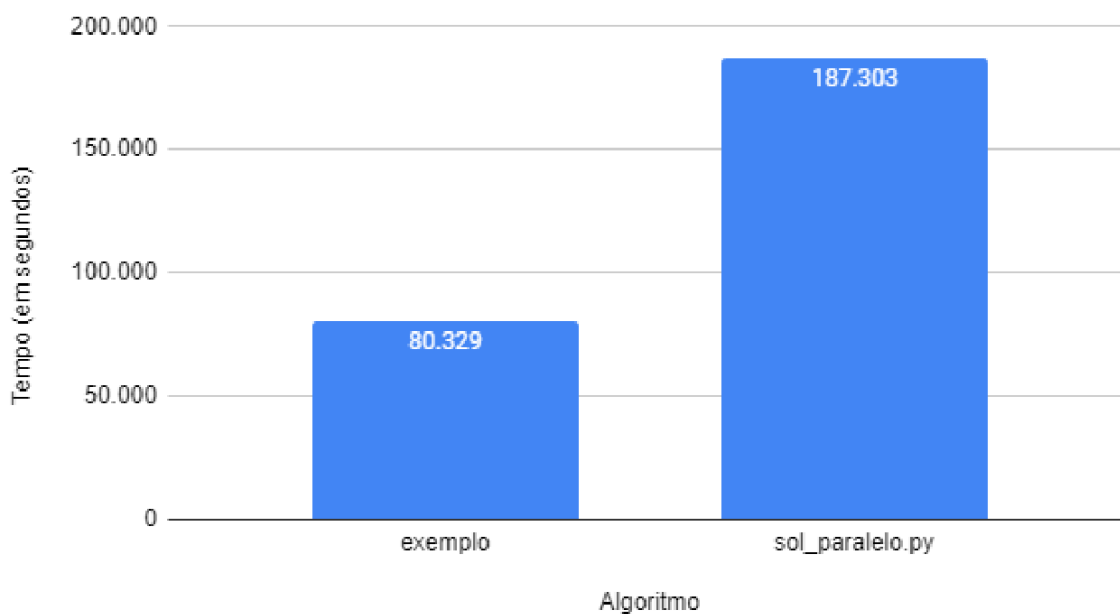
Gere um relatório simples contendo gráficos de desempenho para as diversas execuções

flat75-90



Coloque a comparação da sua solução com a implementação sequencial fornecida;

flat175-34



O maior problema acontece nas estruturas de controle (ex:semáforos)?

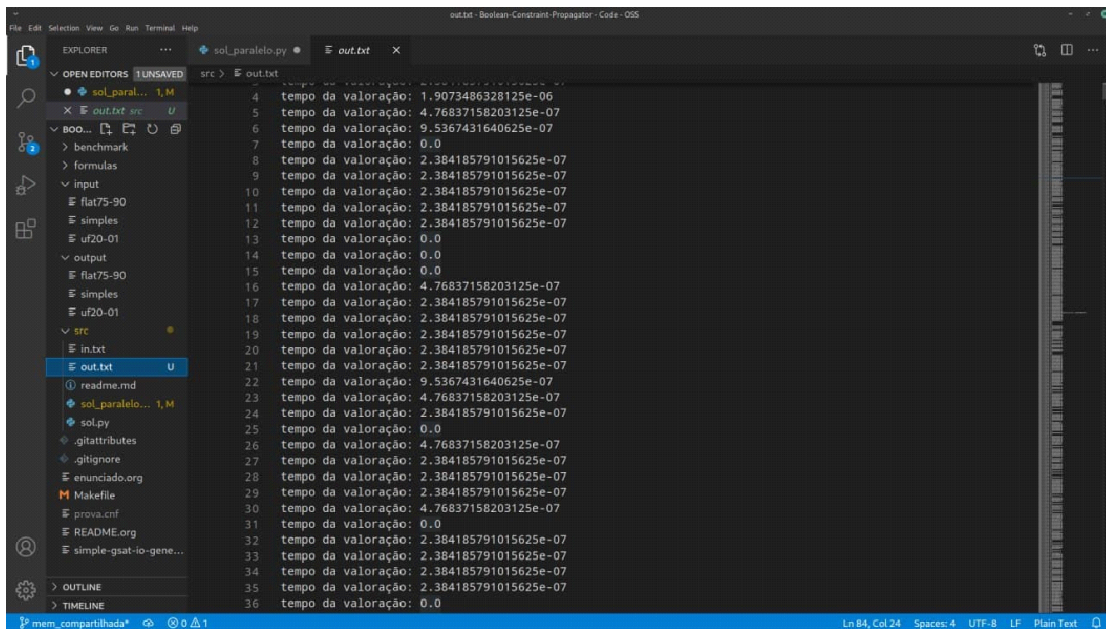
O tempo de leitura da entrada é perceptível?

R: sim, inclusive no caso analisado (flat75-90), o tempo de leitura foi o mais demorado do

código.

Isole o tempo de execução da função que avalia a verificação (descontando o tempo de leitura da fórmula e de cada valoração full e flips)

R: a cada valoração nosso código demora de 0 a 10^{-7} segundos



The screenshot shows a VS Code editor interface. On the left, the Explorer sidebar displays a project structure with folders like 'benchmark', 'formulas', 'input', 'output', 'src', and 'test'. The 'src' folder is expanded, showing files like 'in.txt', 'out.txt', 'readme.md', 'sol_paralelo.py', and 'sol.py'. The 'out.txt' file is selected and its contents are displayed in the main editor window. The file contains a list of execution times in scientific notation, such as '1.9073406328125e-06', '4.76837158203125e-07', and '0.0', each preceded by the text 'tempo da valoração:'. The status bar at the bottom indicates the current line and column as 'Ln 84, Col 24'.

Qual é a complexidade do seu verificador?

$O(n*m*o*\log*o)$

O seu algoritmo aproveita resultados parciais para avaliar flips mais rapidamente?

R: não, nosso algoritmo altera o valor do número escolhido no flip e depois verifica tudo novamente