

Universidade Federal de Mato Grosso do Sul

Campus Ponta Porã

Análise de Algoritmos I

Trabalho Prático II

Problema da Mochila Booleana

Aluno: Daniel de Leon Bailo da Silva

Professor: Eduardo Theodoro Bogue

Maio
2019

Sumário

Resumo	1
1 Introdução	2
2 Knapsack Top-Down	3
2.1 Resultados	3
3 Knapsack Bottom-Up	5
3.1 Resultados	5
4 Análise dos Resultados	7
4.1 Minimizando Resultados	8
4.2 Top-Down	8
4.2.1 Ganhos	9
4.3 Bottom-Up	9
4.3.1 Ganhos	10
5 Resultados Obtidos	11
5.1 Resultados Finais	12

Resumo

Este trabalho consiste em mostrar os resultados obtidos a partir da execução do algoritmo da *Mochilha Booleana* ou *Knapsack 0/1*, em suas versões dinâmicas. Feito isso, dada as instâncias para aplicar os algoritmos, foi comparado o tempo de execução para cada instância nas suas versões dinâmicas, *Top-Down* e *Bottom-Up*.

Considerar o seguinte ambiente para a obtenção dos resultados:

- Processador: Intel Core™ i5-8250U
 - Número de núcleos 4
 - Número de threads 8
 - Frequência baseada em processador 1.60 GHz
 - Frequência turbo max 3.40 GHz
- Memória: 8GB RAM

Este trabalho foi armazenado num repositório *GitHub* para melhor controle do versionamento do código.

<https://github.com/danbailo/T2-Analise-Algoritmos-I>

1 Introdução

Visto que na *Programação Dinâmica* os problemas podem ser abordados de duas formas, o *Top-Down* e o *Bottom-Up*, devemos saber que entre eles existem vantagens e desvantagens quando comparados um com o outro.

Top-Down

Simplemente uma recursão normal com a adição de uma tabela *memoization*.

Bottom-Up

- Prepare uma tabela com o tamanho do número de estados do problema.
- Comece a preencher a tabela através de casos triviais.
- Preencha a tabela de acordo com a ordem topológica do problema.

Vantagens e Desvantagens

- Top-Down
 - Transformação natural através da recursão.
 - Apenas computa um subproblema se ele for necessário.
 - Mais lento se ocorrerem muitas chamadas a subproblemas devido ao overhead recursivo.
- Bottom-Up
 - Mais rápido se muitos subproblemas são computados.
 - Pode ter uma economia de espaço em alguns casos (não precisar criar uma tabela com todos os subproblemas).
 - Não é tão intuitivo.
 - Computa todos os subproblemas.

2 Knapsack Top-Down

Código do algoritmo *Knapsack 0/1 Top-Down* que foi utilizado para obter os resultados onde o mesmo foi escrito na linguagem de programação *Python*.

```
1 def Knapsack(number_items, weight_max, values_items,
2   weight_items):
3     if number_items == 0 or weight_max == 0: return 0
4     if weight_items[number_items-1] > weight_max:
5       return Knapsack(number_items-1, weight_max, values_items,
6   weight_items)
7     if mem[number_items][weight_max] is not False:
8       return mem[number_items][weight_max]
9     temp = max(Knapsack(number_items-1,
10   weight_max-weight_items[number_items-1], values_items,
11   weight_items)+values_items[number_items-1], Knapsack(
12   number_items-1, weight_max, values_items, weight_items))
13   mem[number_items][weight_max] = temp
14   return temp
```

Algoritmo 1: Top-Down

2.1 Resultados

Após aplicar o algoritmo [Top-Down](#) nas instâncias propostas pelo professor, os seguintes resultados foram obtidos.

Tempo total para executar todas instâncias:	26.8087 segundos
Média de tempo de execução por instância:	0.8647 segundos
Maior tempo gasto por uma das instâncias:	1.8696 segundos
Menor tempo gasto por uma das instâncias:	0.2439 segundos
Instância com maior tempo de execução:	s025.kp
Instância com menor tempo de execução:	s008.kp

Tabela 1: Resultados Top-Down para 1 execução

Amplitude:	1.6256 segundos
Erro:	0.1070 segundos
Variância:	0.3437 segundos
Desvio Padrão:	0.5863 segundos
Desvio Absoluto:	0.2471 segundos

Tabela 2: Estatísticas

Nos gráficos representados pelas figuras 1 e 2, podemos ver o tempo que cada instâncias demorou para antingir tal resultado e qual resultado foi obtido, respectivamente. Na figura 3 conseguimos vizualizar o tempo de execução para cada instância.

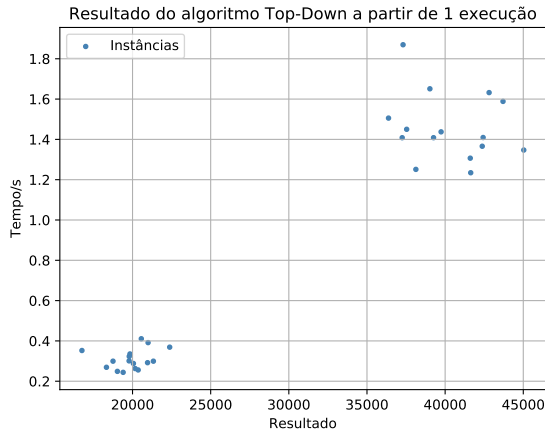


Figura 1: Tempo/Resultado

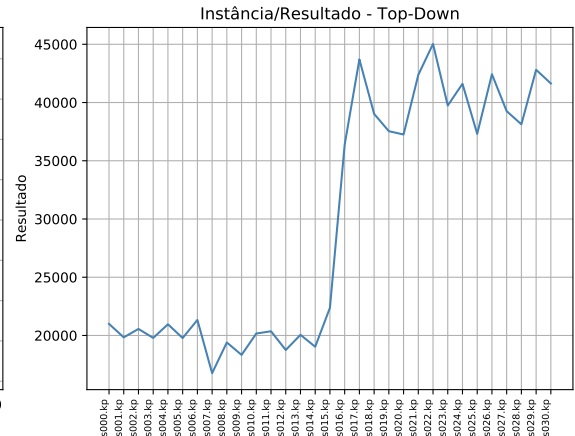


Figura 2: Resultado/Instância

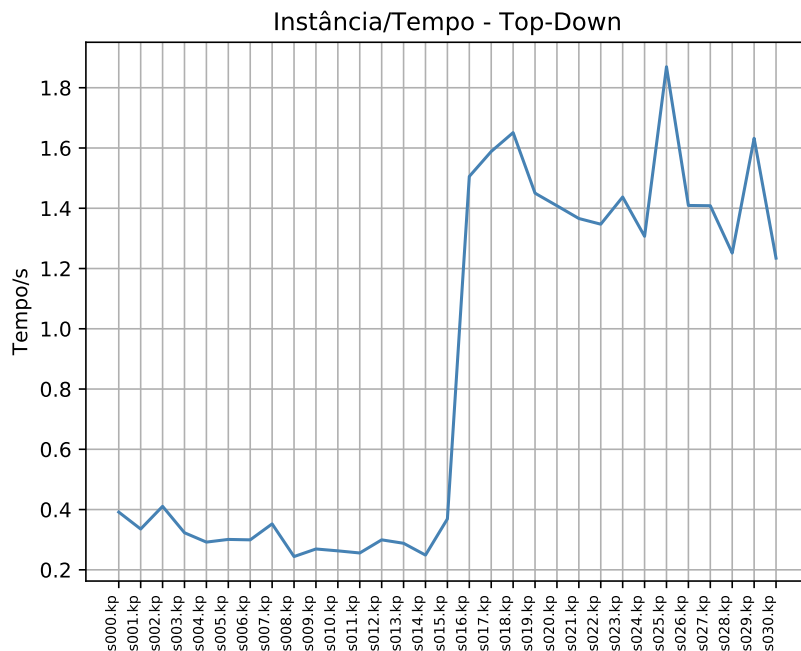


Figura 3: Gráfico do Tempo/Instância

3 Knapsack Bottom-Up

Código do algoritmo *Knapsack 0/1 Bottom-Up* que foi utilizado para obter os resultados onde o mesmo foi escrito na linguagem de programação *Python*.

```
1 def Knapsack(number_items, weight_max, values_items,
2   weight_items):
3     K = [[0 for x in range(weight_max + 1)] for x in range(
4       number_items + 1)]
5     for i in range(number_items + 1):
6       for w in range(weight_max + 1):
7         if i == 0 or w == 0: K[i][w] = 0
8         elif weight_items[i-1] <= w:
9           K[i][w] = max(values_items[i-1]+K[i-1][w-
10            weight_items[i-1]], K[i-1][w])
11         else: K[i][w] = K[i-1][w]
12     return K[number_items][weight_max]
```

Algoritmo 2: Bottom-Up

3.1 Resultados

Após aplicar o algoritmo [Bottom-Up](#) nas instâncias propostas pelo professor, os seguintes resultados foram obtidos.

Tempo total para executar todas instâncias:	18.1289 segundos
Média de tempo de execução por instância:	0.5848 segundos
Maior tempo gasto por uma das instâncias:	1.1784 segundos
Menor tempo gasto por uma das instâncias:	0.1937 segundos
Instância com maior tempo de execução:	s018.kp
Instância com menor tempo de execução:	s014.kp

Tabela 3: Resultados Bottom-Up para 1 execução

Amplitude:	0.9847 segundos
Erro:	0.0680 segundos
Variância:	0.1390 segundos
Desvio Padrão:	0.3728 segundos
Desvio Absoluto:	0.1371 segundos

Tabela 4: Estatísticas

Nos gráficos representados pelas figuras 1 e 2, podemos ver o tempo que cada instâncias demorou para antingir tal resultado e qual resultado foi obtido, respectivamente. Na figura 3 conseguimos vizualizar o tempo de execução para cada instância.

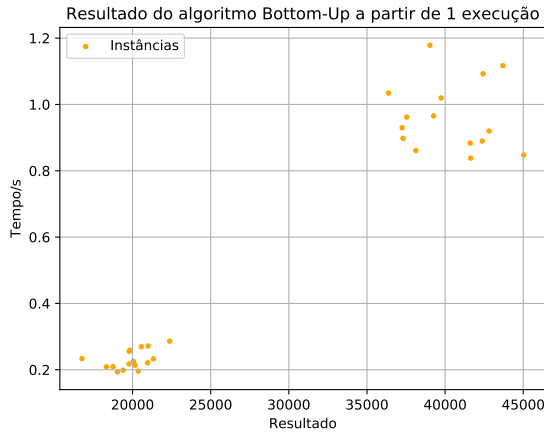


Figura 4: Tempo/Resultado

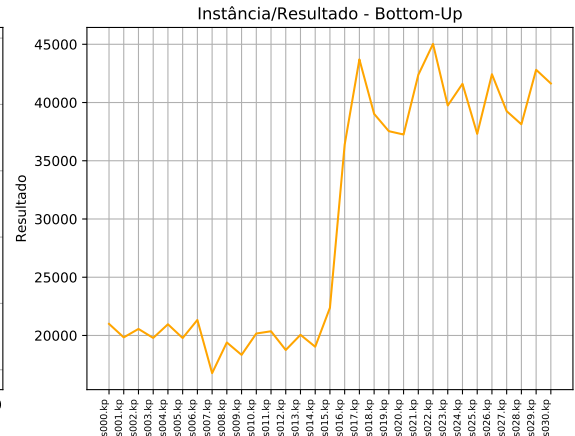


Figura 5: Resultado/Instância

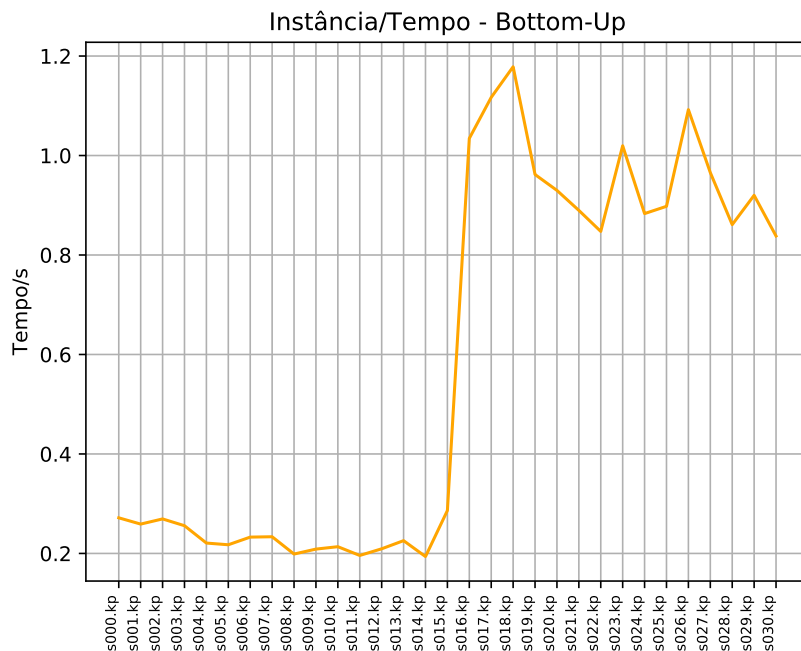


Figura 6: Gráfico do Tempo/Instância

4 Análise dos Resultados

Como podemos ver abaixo, na figura 7, que é responsável por mostrar os resultados atingidos para cada instância, os gráficos estão sobrepostos e isso se deve pelo fato de que, independente dos algoritmos serem escritos e abordarem ideias diferentes, o objetivo final é o mesmo.

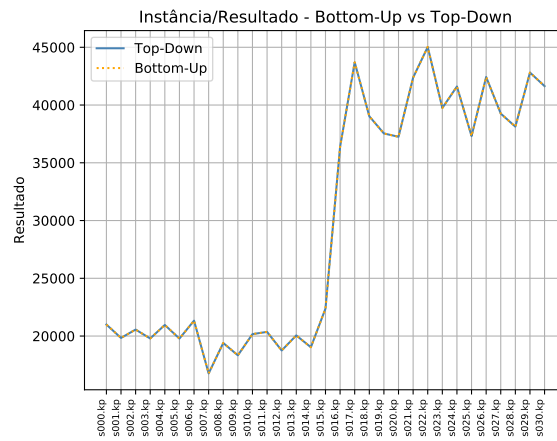


Figura 7: Gráfico do Resultado/Instância

Como podemos ver, o tempo necessário para atingir os mesmos resultados são bem maiores quando comparamos o [Knapsack Top-Down](#) ao [Knapsack Bottom-Up](#), computacionalmente falando. Porém, era de se esperar um resultado como esse, como vimos na página 2 as características de cada um.

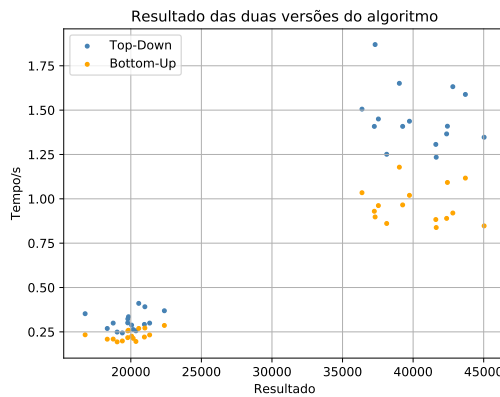


Figura 8: Gráfico do Tempo/Resultado

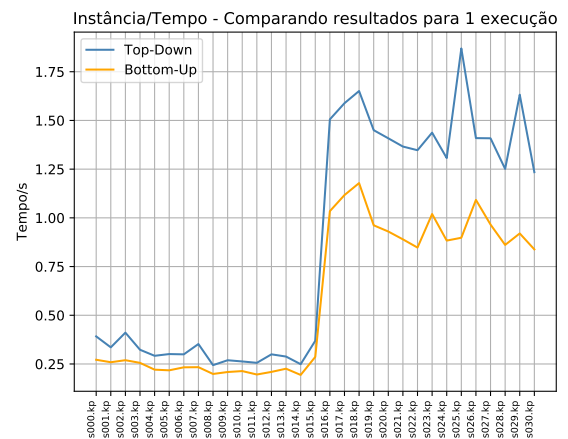


Figura 9: Gráfico do Tempo/Instância

4.1 Minimizando Resultados

A fim de comprovar o comportamento dos gráficos, decidi executar **100 vezes** as instâncias com o objetivo de minimizar a variação do tempo de execução de cada instância.

Para cada instância é armazenado seu tempo de execução individual, logo, foi necessário a cada execução somar o tempo de cada instância para obter o tempo gasto para esse algoritmo para uma única execução. Feito isso, bastou repetir o mesmo processo para todas iterações. Assim, obtive o tempo total de execução para os dois algoritmos. O tempo médio é em relação ao tempo que uma única instância gastou em 100 execuções.

4.2 Top-Down

Tempo total de execução: 39 minutos

Tempo médio gasto por cada instância no total: 1 minuto e 25 seg

Tabela 5: Tempo total

Após coletar os dados de todas as execuções, foi se calculado a média total para cada item e obtido um resultado mais expressivo a fim de minimizar a variação.

Tempo de execução: 23.4056 segundos

Média de execução por instância: 0.7550 segundos

Maior tempo gasto por uma das instâncias: 1.3737 segundos

Menor tempo gasto por uma das instâncias: 0.2419 segundos

Instância com maior tempo de execução: s027.kp

Instância com menor tempo de execução: s014.kp

Tabela 6: Resultados a partir de 100 execuções

Amplitude: 1.1317 segundos

Erro: 0.0905 segundos

Variância: 0.2457 segundos

Desvio Padrão: 0.4957 segundos

Desvio Absoluto: 0.1431 segundos

Tabela 7: Estatísticas

4.2.1 Ganhos

Ao compararmos os resultados das tabelas 1 e 2 com as tabelas 6 e 7, podemos ver qual é a diferença entre os seguintes resultados obtidos:

p/ **Tempo de execução:** 3.4031 segundos
p/ **Média de execução por instância:** 0.1097 segundos
p/ **Maior tempo gasto por uma das instâncias:** 0.4959 segundos
p/ **Menor tempo gasto por uma das instâncias:** 0.0020 segundos

Tabela 8: Diferença de 1 execução para 100 execuções

p/ **Amplitude:** 0.4939 segundos
p/ **Erro:** 0.0165 segundos
p/ **Variância:** 0.098 segundos
p/ **Desvio Padrão:** 0.0906 segundos
p/ **Desvio Absoluto:** 0.1039 segundos

Tabela 9: Estatísticas

4.3 Bottom-Up

Tempo total de execução: 29 minutos
Tempo médio gasto por cada instância no total: 55.9276 segundos

Tabela 10: Tempo total

Após coletar os dados de todas as execuções, foi se calculado a média total para cada item e obtido um resultado mais expressivo a fim de minimizar a variação.

Tempo de execução: 17.3375 segundos
Média de execução por instância: 0.5592 segundos
Maior tempo gasto por uma das instâncias: 0.9883 segundos
Menor tempo gasto por uma das instâncias: 0.1929 segundos
Instância com maior tempo de execução: s027.kp
Instância com menor tempo de execução: s014.kp

Tabela 11: Resultados a partir de 100 execuções

Amplitude: 0.7953 segundos
Erro: 0.0635 segundos
Variância: 0.1209 segundos
Desvio Padrão: 0.3478 segundos
Desvio Absoluto: 0.1188 segundos

Tabela 12: Estatísticas

4.3.1 Ganhos

Ao compararmos os resultados das tabelas 3 e 4 com as tabelas 11 e 12, podemos ver qual é a diferença entre os seguintes resultados obtidos:

p/ **Tempo de execução:** 0.7914 segundos
p/ **Média de execução por instância:** 0.0255 segundos
p/ **Maior tempo gasto por uma das instâncias:** 0.1900 segundos
p/ **Menor tempo gasto por uma das instâncias:** 0.0008 segundos

Tabela 13: Diferença de 1 execução para 100 execuções

p/ **Amplitude:** 0.1894 segundos
p/ **Erro:** 0.0045 segundos
p/ **Variância:** 0.0181 segundos
p/ **Desvio Padrão:** 0.0250 segundos
p/ **Desvio Absoluto:** 0.0182 segundos

Tabela 14: Estatísticas

Feito isso, o objetivo de minimizar a variação foi atingida...

5 Resultados Obtidos

Ao analisar as tabelas 8 e 13 que mostram a diferença dos resultados atingidos, podemos comparar com o seguinte grafico

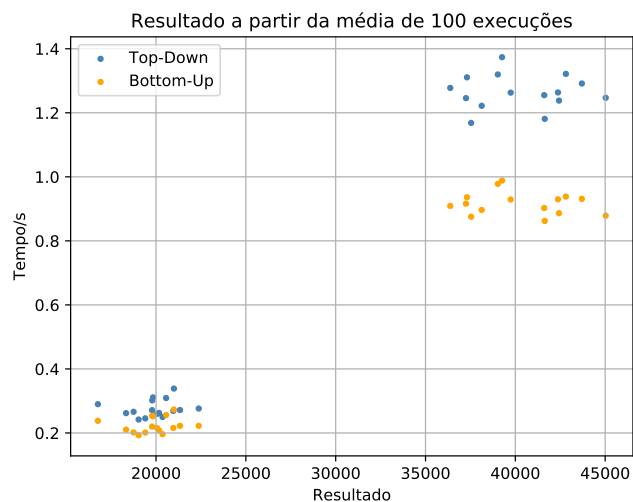


Figura 10: Gráfico do Tempo/Resultado

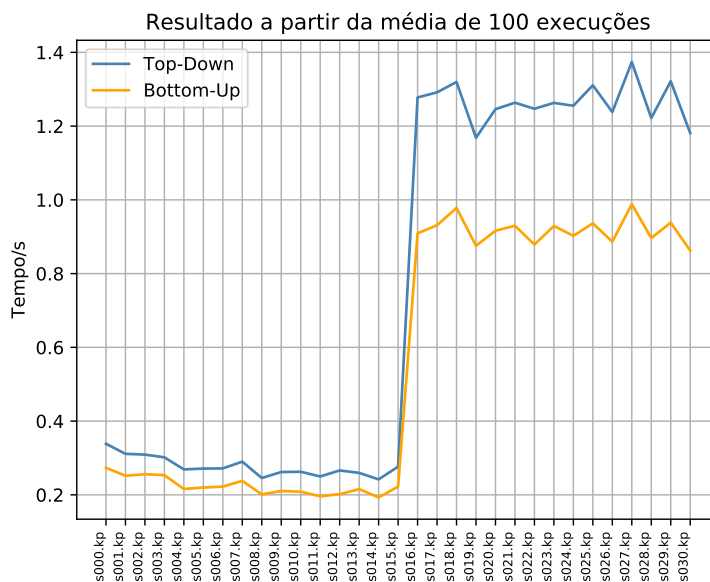


Figura 11: Gráfico do Tempo/Instância

5.1 Resultados Finais

Com isso, podemos concluir que, o objetivo de minizar a variação do tempo de cada execução foi atingida.

No gráfico abaixo, temos ilustrado o tempo que demandou cada resultado, sendo que, as legendas que possuem "AVG" no nome, significa que são os gráficos representados após realizar a média entre as 100 execuções.

Esta figura, tanto quanto a figura 13, representa uma união de todos resultados mostrados até agora.

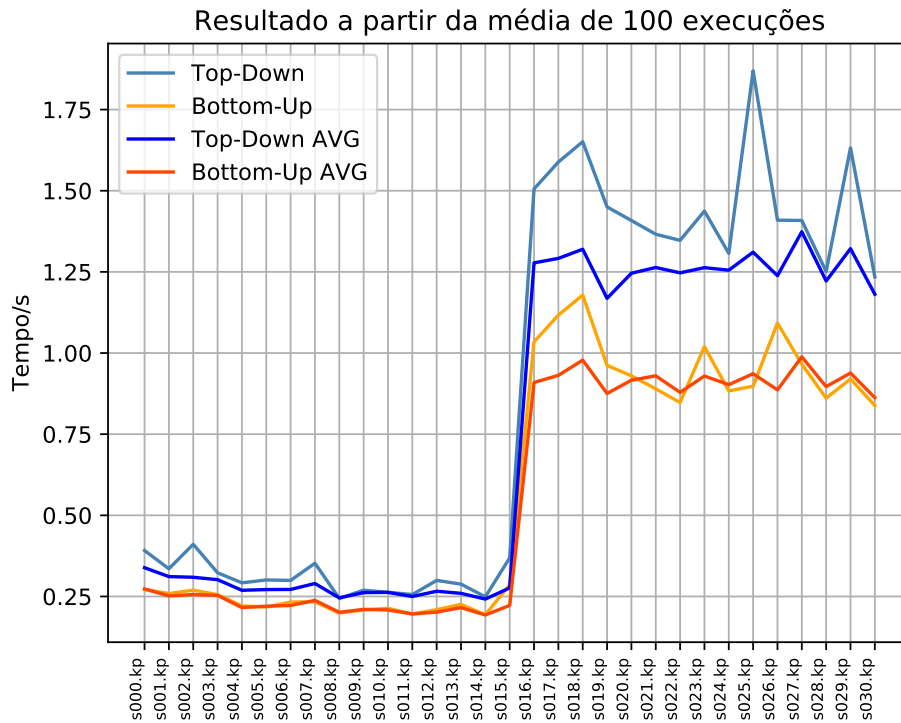


Figura 12: Gráfico do Tempo/Instância - Mesclado

Ao analisarmos o gráfico abaixo, podemos ver que a diferença que [Knapsack Top-Down](#) sofreu após passar pelo processo de refinação foi gritante.

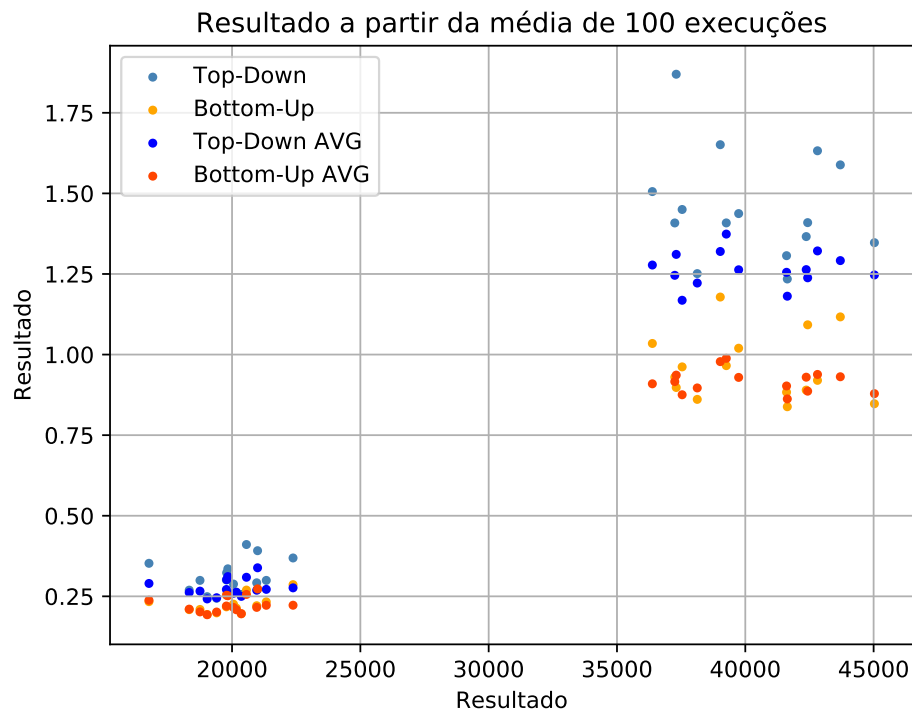


Figura 13: Gráfico do Resultado/Instância