Universidade de São Paulo

Problema de Designação

Parte 1

SME0110 - Programação Matemática

Profas. Franklina Toledo e Marina Andretta

COUTINHO, Gabriel dos Reis
gabriel.dos.reis.coutinho@usp.br
9807124

RIBEIRO, Frederico Bulhões de Souza
fredbr@usp.br
11208440

MOREIRA, Maria Eduarda Kawakami
kawakamimadu@usp.br
11218751

CUNHA, Matheus Barcellos de Castro
mathes_cunha@usp.br
11208238

MASTROBUONO, Gustavo Tuani
gumastro@usp.br
10734411

DOS SANTOS, Juliana Perfeito
julianaperfeito@usp.br
10295141

1 Modelo

1.1 Parâmetros

Os parâmetros do modelo consistem em:

- n: número de tarefas a serem realizadas,
- m: número de agentes que realizarão as tarefas,
- c_{ij} : "lucro"(satisfação) do agente i realizar a tarefa j

No nosso modelo, o número de agentes é igual ao de tarefas, portanto $n = m$.

1.2 Variáveis

A variável do modelo é uma variável binária que determina se o agente i realiza a tarefa j ou não.

$$x_{ij} = \begin{cases} 0 & \text{se agente i não realiza a tarefa j} \\ 1 & \text{se o agente i realiza a tarefa j} \end{cases}$$

1.3 Restrições

É necessário garantir a binaridade da variável do modelo, faz-se isso através de uma restrição:

$$x_{ij} \in \{0, 1\}, \text{ i = 1, 2, ..., n; j = 1, 2, ..., n} \tag{1}$$

É necessário também garantir que para uma tarefa j apenas um agente a realize, e que um agente i realize uma única tarefa. Para isso, têm-se as restrições:

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, 2, ..., n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, ..., n \tag{3}$$

1.4 Função Objetivo

O objetivo do problema é maximizar a satisfação global dos agentes ao realizarem as tarefas a eles atribuídas.

Maximizar $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.

2 Toy Problem

2.1 Contextualização

A EDSSC, uma empresa de terceirização de serviços de engenharia de software, tem 5 times distintos os quais todo mês negociam diferentes preços para realizar 5 diferentes serviços para 5 diferentes empresas.

Esse mês os times da EDSSC conseguiram negociar a seguinte tabela de preços.

	Empresa 1	Empresa 2	Empresa 3	Empresa 4	Empresa 5
Time 1	7	6	5	7	10
Time 2	6	7	7	7	9
Time 3	8	8	9	8	9
Time 4	7	6	4	2	4
Time 5	10	2	3	3	4

Qual a combinação que faz com que a EDSSC obtenha o maior lucro possível, fazendo com que cada time aceite exatamente um serviço?

2.2 Resolução

Podemos usar o modelo descrito na seção 1 para resolver o problema proposto, visto que a EDSSC procura maximizar seu lucro.

Tendo isso em mente, temos a seguinte matriz de custo:

$$c = \begin{bmatrix} 7 & 6 & 5 & 7 & 10 \\ 6 & 7 & 7 & 7 & 9 \\ 8 & 8 & 9 & 8 & 9 \\ 7 & 6 & 4 & 2 & 4 \\ 10 & 2 & 3 & 3 & 4 \end{bmatrix}$$

.

Visando o maior lucro, temos a seguinte matriz de frequência que maximiza a função objetivo do problema:

$$x = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

.

Portanto, temos que a solução ótima é dada por:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} = 10 + 7 + 9 + 6 + 10 = 42$$

3 Resolução das instâncias

Todos os algoritmos desenvolvidos foram implementados utilizando a linguagem programação Julia com um conjunto de bibliotecas disponíveis que puderam ser obtidas por meio do gerenciador de pacotes da própria linguagem.

3.1 Modelagem

Com o intuito de resolver um problema genérico de designação de maneira programática, o primeiro passo tomado foi o de usar uma linguagem de modelagem para abstrair o modelo.

Nesse passo foi usada a linguagem de modelagem JuMP que permitiu o uso de macros para definir: variável de decisão, função objetivo e restrições do modelo.

3.2 Solvers

Após abstrair a definição do modelo usando JuMP, foram utilizados os solvers Cbc (Coin-or branch and cut), Gurobi e Hungáro para trazer as melhores soluções possíveis.

Para todas as otimizações geradas foi imposto um limite de tempo de 30 minutos, e nenhuma instância chegou perto de atingir o tempo limite.

Para o modelo Húngaro não foi necessário usar o modelo abstraído pelo JuMP, visto que, dada uma matriz de custo, o problema de designação já está implícito.

Obs: Todos os solvers usados foram obtidos diretamente do gerenciador de pacotes da linguagem Julia (para o Gurobi alguns passos a mais são necessários devido a sua licença de utilização).

4 Análise dos dados

4.1 Especificações técnicas

Todos os dados a serem analisados são resultados da execução dos algoritmos implantados na maquina com as seguintes especificações técnicas:

- Model Name: MacBook Pro
- Chip: Apple M1
- Total Number of Cores: 8 (4 performance and 4 efficiency)
- Memory: 16 GB
- OS: macOS Big Sur
- Julia version: 1.6.3

4.2 Instância A

4.2.1 Tabelas

A primeira tabela mostra se as soluções encontradas nos solvers são iguais entre si.

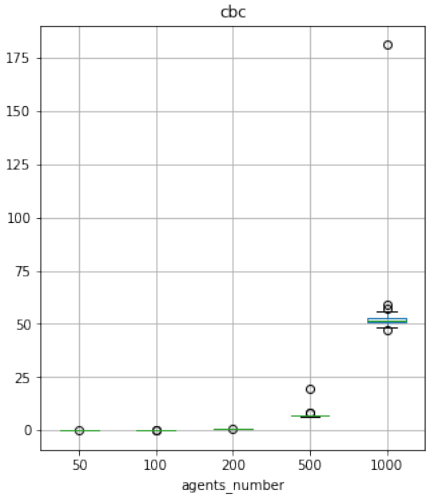
	agents_number	cbc_solution	cbc_is_optimal	gurobi_solution	gurobi_is_optimal	hungarian_solution	are_equal
0	100	100	True	100	True	100	True
1	1000	1000	True	1000	True	1000	True
2	200	200	True	200	True	200	True
3	50	50	True	50	True	50	True
4	500	500	True	500	True	500	True
5	100	4972	True	4972	True	4972	True
6	1000	50000	True	50000	True	50000	True
7	200	9996	True	9996	True	9996	True
8	50	2464	True	2464	True	2464	True
9	500	25000	True	25000	True	25000	True
10	100	4979	True	4979	True	4979	True
11	1000	50000	True	50000	True	50000	True
12	200	9995	True	9995	True	9995	True

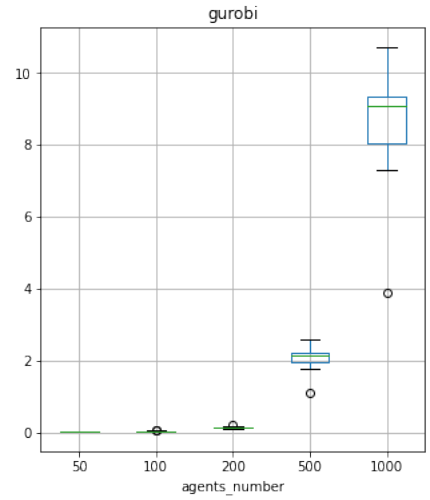
A segunda tabela conterá o tempo que o solver levou para resolver as instâncias.

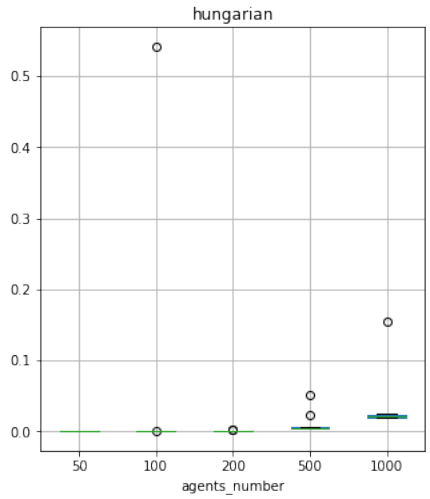
	agents_number	cbc	gurobi	hungarian
0	100	0.339009	0.049691	0.541785
1	1000	181.141	3.87687	0.155365
2	200	0.956972	0.10566	0.00276387
3	50	0.0360661	0.00930285	0.000140041
4	500	19.4373	1.0982	0.023353
5	100	0.282089	0.0397429	0.00143025
6	1000	56.0942	8.9338	0.0190592
7	200	0.599316	0.156279	0.000977209
8	50	0.0332241	0.00988102	0.00012575
9	500	8.51242	1.91467	0.0510599
10	100	0.165752	0.0299292	0.000285792
11	1000	59.2277	8.2021	0.0201864
12	200	0.583674	0.145027	0.0009025

4.2.2 Análise

Aqui podemos ver a distribuição dos tempos de execução em cada instância, agrupados pelo número de agentes e pela engine de execução.


cbc
175
150
125
100
75
50
25
0
50
100
200
500
1000
agents_number


gurobi
10
8
6
4
2
0
50
100
200
500
1000
agents_number


hungarian
0.5
0.4
0.3
0.2
0.1
0.0
50
100
200
500
1000
agents_number

Podemos observar que o cbc teve um tempo de execução muito maior do que os demais, mesmo sem levar outliers em conta. O algoritmo húngaro obteve o melhor desempenho, obtendo tempos de execução inferiores a um segundo em todos os casos.
Houveram alguns outliers, com um dos testes levando mais de 170 segundos no cbc, e um dos testes levando 0.5 segundos no hungarian, onde os outros levaram menos que 0.1 segundo.

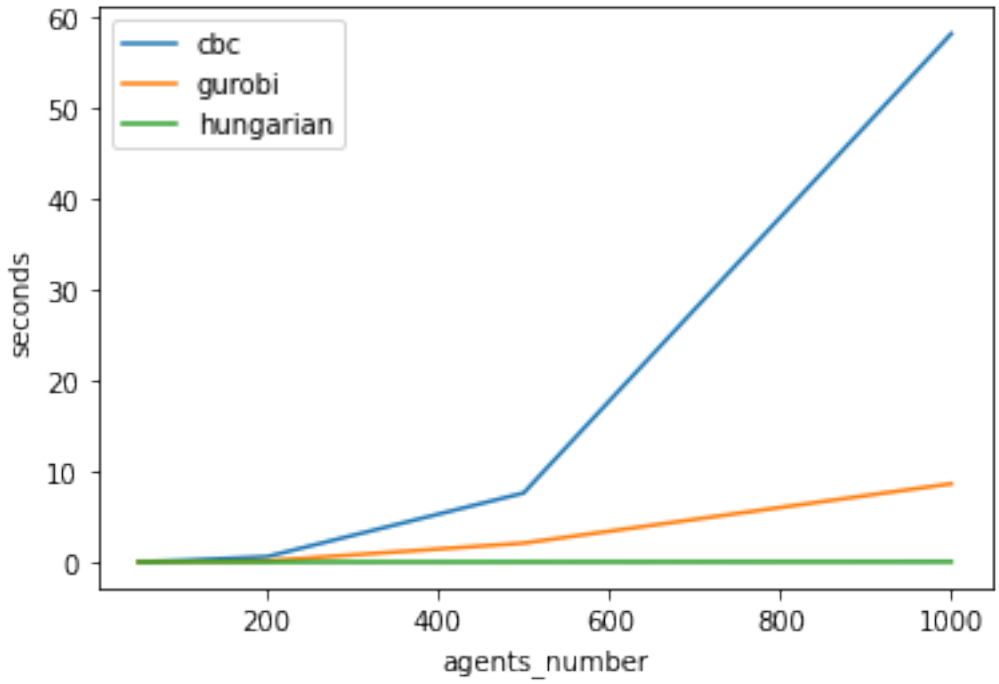


Figura 1: Gráfico de tamanho da amostra por média do tempo de solução

4.3 Instância B

4.3.1 Tabelas

A primeira tabela mostra se as soluções encontradas nos solvers são iguais entre si.

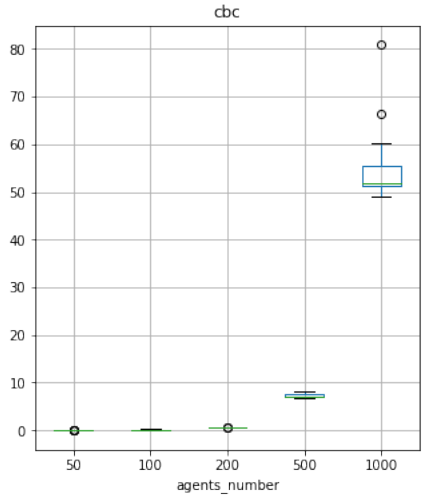
	agents_number	cbc_solution	cbc_is_optimal	gurobi_solution	gurobi_is_optimal	hungarian_solution	are_equal
0	100	4935.02	True	4935.02	True	4935.02	True
1	1000	49926.4	True	49926.4	True	49926.4	True
2	200	9924.66	True	9924.66	True	9924.66	True
3	50	2421.15	True	2421.15	True	2421.15	True
4	500	24921.1	True	24921.1	True	24921.1	True
5	100	4927.71	True	4927.71	True	4927.71	True
6	1000	49926.2	True	49926.2	True	49926.2	True
7	200	9931.69	True	9931.69	True	9931.69	True
8	50	2423.4	True	2423.4	True	2423.4	True
9	500	24925.6	True	24925.6	True	24925.6	True
10	100	4929.36	True	4929.36	True	4929.36	True
11	1000	49925.7	True	49925.7	True	49925.7	True
12	200	9927.93	True	9927.93	True	9927.93	True

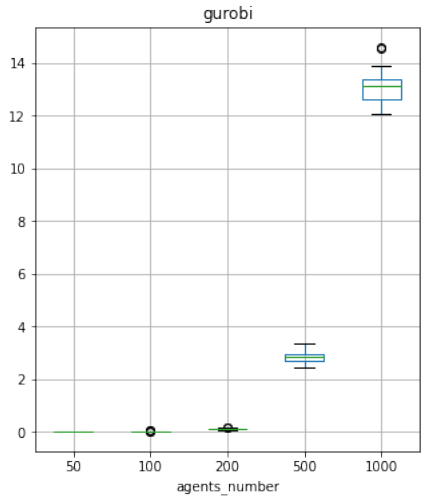
A segunda tabela conterá o tempo que o solver levou para resolver as instâncias.

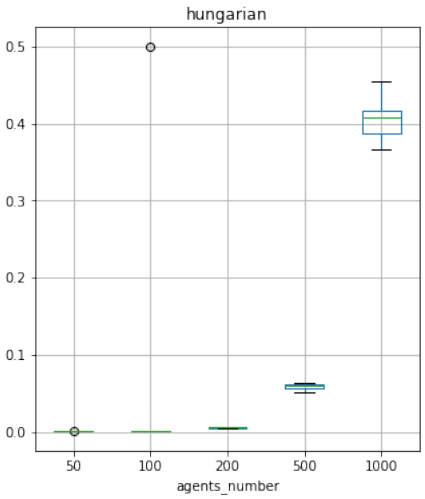
	agents_number	cbc	gurobi	hungarian
0	100	0.224349	0.0429928	0.500588
1	1000	56.589	13.5589	0.431304
2	200	0.611783	0.119022	0.004983
3	50	0.0413752	0.0108931	0.000186667
4	500	7.87535	3.0891	0.0610788
5	100	0.180668	0.029247	0.00106067
6	1000	66.498	13.2337	0.390896
7	200	0.60237	0.129186	0.00501413
8	50	0.0362401	0.011024	0.000213208
9	500	8.21584	2.85412	0.0599286
10	100	0.211595	0.029916	0.00106196
11	1000	80.8254	14.5353	0.411637
12	200	0.60192	0.129106	0.00544817

4.3.2 Análise

Aqui podemos ver a distribuição dos tempos de execução em cada instância, agrupados pelo número de agentes e pela engine de execução.


cbc
80
70
60
50
40
30
20
10
0
50
100
200
500
1000
agents_number


gurobi
14
12
10
8
6
4
2
0
50
100
200
500
1000
agents_number


hungarian
0.5
0.4
0.3
0.2
0.1
0.0
50
100
200
500
1000
agents_number

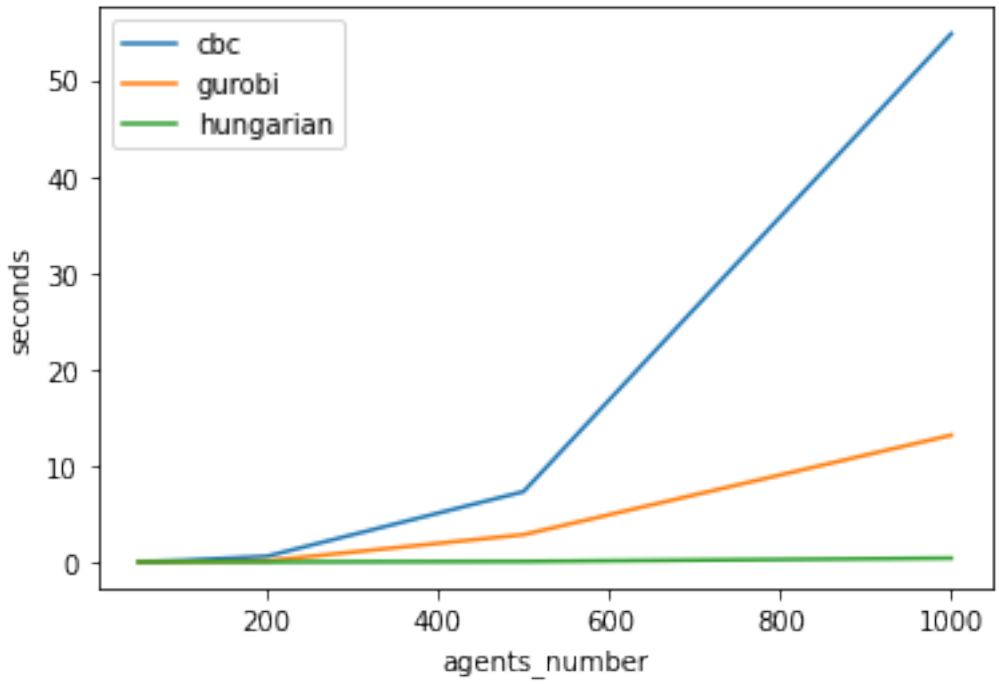


Figura 2: Gráfico de tamanho da amostra por média do tempo de solução

O gráfico acima mostra a diferença entre os tempos de solução dos solvers baseado no tamanho da amostra. Foi feito uma média dos valores agrupando pelo tamanho da amostra. Como pode-se perceber, em amostras pequenas a diferença do tempo entre os solvers não é tão pronunciada, porém ao aumentar o número de agentes ela se destaca.