Metodologias Experimentais em Informática
2021/22 - 1º Semestre
Meta 1 - Análise Exploratória de Dados

Cesário Silva [2015230724] - `cesariors@student.dei.uc.pt`
João Nunes [2017247442] - `joaonunes@student.dei.uc.pt`
Pedro Carvalho [2017267408] - `pccarvalho@student.dei.uc.pt`

9 de novembro de 2021

1 Introdução

No âmbito da disciplina de MEI, para esta primeira meta, foi realizado um estudo que tenta averiguar qual o desempenho dos algoritmos de backtracking e quais serão os melhores resultados temporais a escalonar os exames para os alunos de uma universidade.

De modo a aferir o desempenho dos diferentes algoritmos foi, em primeiro lugar, analisadas as diferentes variáveis do sistema e criados ambientes de teste que executava os dois algoritmos para diferentes combinações de variáveis, o qual será explicado mais em detalhe na secção 2 sobre a análise do problema.

Com o estudo e consequente previsão foi realizado a experimentação que visa estudar a performance dos dois algoritmos. A descrição da experimentação apresenta-se na secção 3 e os resultados da mesma na secção 4 e, posteriormente, foi feita a análise e discusão dos respetivos resultados sobre o comportamento dos dois algoritmo na secção 5.

Por fim, porcedeu-se à conclusão do trabalho na secção 6.

2 Análise do Problema

De modo a ser possível fazer uma análise mais concisa do trabalho, primeiramente foi identificada uma *research question*, sendo esta a base de toda a exploração dos dados e respetiva análise e discusão dos dados.

Como é que a performance de um algoritmo de escalonamento de exames é afetada pelo número de exames e a probabilidade de dois exames terem pelo menos um aluno em comum?

Com este objetivo em mente, será apresentado de seguida uma síntese dos dois algoritmos de escalonamento de exames utilizados e seguido da identificação das variáveis independentes e dependentes do ambiente.

2.1 Algoritmos de Escalonamento de Exames

Para este trabalho, foram apresentados dois algoritmos de backtracking com uma seed aleatória. Por definição, um algoritmo de backtracking permite construir uma solução recursivamente, removendo os resultados que não vão ao encontro da solução procurada.

O *code1* vai percorrer todos os exames que tem e vai preenchendo o calendário quando encontra um espaço livre para o exame em que se encontra.

O *code2* vai percorrer os espaços no calendário até encontrar um espaço vazio onde pode colocar um exame com o tempo pretendido.

2.2 Variáveis Independentes e Dependentes

Para se proceder à análise exploratória de dados deste trabalho, é importante identificar as variáveis presentes no sistema, assim como verificar se estas são variáveis dependentes ou independentes.

As tabelas seguintes mostram as variáveis do sistema.

Variáveis Independentes	**Descrição**
n	Número de exames
p	Probabilidade de 2 exames terempelo menos um aluno em comum
s	*Random seed* do sistema
c	Número de *time slots* encontrados pelo algoritmo

Tabela 1: Tabela de variáveis dependentes do sistema

Variáveis Dependetes	**Descrição**
cpu_time_used	Tempo total de execução do algoritmo de escalonamento

Tabela 2: Tabela de variáveis independentes do sistema

Durante os próximos cenários de experimentação iremos observar a dependência do tempo final com o resto das variáveis. Apesar de ser uma variável de saída do sistema, o número de *time slots* encontrados pelos algoritmos não será considerada uma variável dependente (ver subsecção 5.4 para mais detalhes).

3 Cenário de Experimentação

No que diz respeito ao cenário de experimentação, em primeiro lugar, é de notar que os dados gerados através do gerador fornecido em Python (*gen.py*) provêm de um gerador sintético, onde todas as diferentes gerações foram feitas com uma *random seed* diferente.

Em relação às variáveis independentes, que são apresentadas como variáveis de entrada no *gen.py*, foi variado o número de exames e a probabilidade de dois exames terem um aluno em comum. No que diz respeito ao número de exames, estes foram variados de 10 a 100 exames, de 10 em 10. Quanto à probabilidade de dois exames terem pelo menos um aluno em comum, esta foi variada com 5 valores de probabilidade diferentes (0.1, 0.3, 0.5, 0.7 e 0.9). Para tornar este processo de geração de casos de teste mais eficiente, foi criado um *shell script* que percorria cada número de exames e probabilidade para gerar um ficheiro com os respetivos dados indicados.

Depois de gerados os ficheiros de dados, cada um destes foi inserido como *input* nos dois algoritmos de escalonamento de exames (*code1* e *code2*) onde, dado um certo tempo de corte - tempo máximo para o algoritmo tentar encontrar os *time slots* necessários - que, para esta experimentação, foi sempre considerado um tempo de 100 segundos, o algoritmo tentaria resolver os dados do ficheiro e encontrar os *time slots* necessários para esse caso. Assim como o processo de geração, também foi criado um *shell script* para agilizar o processo de resolução dos diferentes algoritmos onde, para cada caso de teste, foi compilado 30 vezes, ou seja, para cada número de exames com uma dada probabilidade foi corrido 30 vezes esse caso de teste em cada um dos códigos dos dois algoritmos.

Após todos estes teste em cada algoritmo, foram ainda gerados mais alguns casos de teste para verificar o estado da subida dos tempos necessários para encontrar *time slots*. Por exmeplo, com probabilidade 0.5, os algoritmos já não conseguiam executar com sucesso o sua tarefa com um número de exames igual a 40, contudo ainda apresentava resultados aceitáveis com 30 exames. Logo, nesta situação, foram gerados casos de teste com valores entre 31 e 39, 1 a 1, de modo a verificar como evoluia o tempo de execução.

De seguida, na secção 4, será apresentada toda a análise exploratória de dados com base em gráficos e outros cálculos realizados de modo a ser possível responder à *research question* apresentada na secção 2.

4 Resultados

Com base nas experiências descritas na secção anterior, foram obtidos vários resultados que permittem responder à *research question* apresentada posteriormente.

Em primeiro lugar, foi feita a análise temporal com base no número de exames e na probabilidade de dois exames terem pelo menos um aluno em comum, onde foi feita a média de cada uma das combinações exames/probabilidade em cada um dos algoritmos, como é apresentado nas duas tabelas seguintes (tabelas 3 e 4).

code1	10	20	30	40	50	60	70	80	90	100
0.1	$\simeq$ 0.000000	0.001000	0.001533	0.013767	5.506233	93.606367	100.000000	100.000000	100.000000	100.000000
0.3	$\simeq$ 0.000000	0.001000	0.074633	97.307900	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.5	$\simeq$ 0.000000	0.007033	2.421833	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.7	$\simeq$ 0.000000	0.056533	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.9	0.001000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000

Tabela 3: Média dos tempos no algoritmo code1

code1	10	20	30	40	50	60	70	80	90	100
0.1	$\simeq$ 0.000000	$\simeq$ 0.000000	0.001000	0.009133	5.451533	90.932733	100.000000	100.000000	100.000000	100.000000
0.3	$\simeq$ 0.000000	0.000500	0.070633	95.531800	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.5	$\simeq$ 0.000000	0.007533	2.471833	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.7	$\simeq$ 0.000000	0.060833	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
0.9	0.001000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000

Tabela 4: Média dos tempos no algoritmo code2

Seguidamente, para fazer a análise da dependência do tempo de execução com base no número de exames foi feito um gráfico de pontos e calculada a sua regressão linear para saber a equação da reta (como demonstra a figura 1) mediante o comando *summary(lm.out)*. Foi também encontrado, pelo mesmo comando, o valor de r^2 que é igual a **0.7838** no *code1* e **0.7718** no *code2* e, de seguida, calculado o seu valor teórico através de cálculos para se verificar o quanto se aproximava do valor obtido.

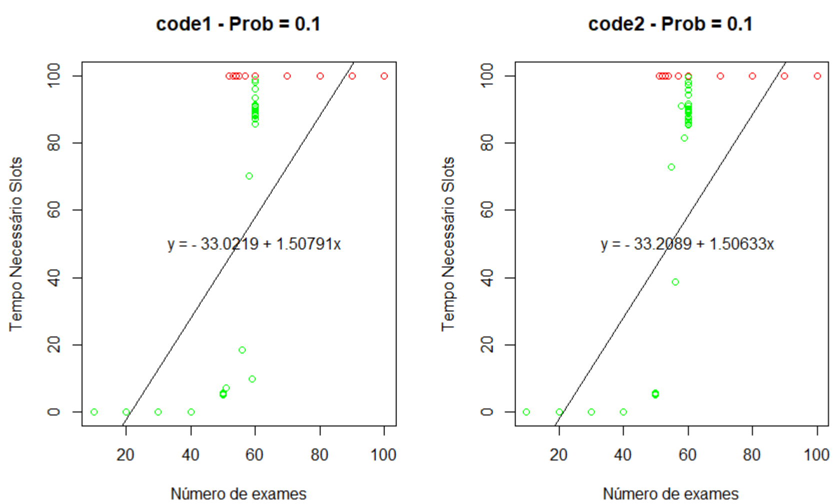

Figura 1: Gráficos de pontos de número de exames por tempo de execução com a respetiva regressão linear para probabilidade 0.1.
A vermelho encontram-se os testes que o algoritmo não conseguiu executar em tempo válido (100 segundos).

Os dados teóricos obtidos através dos cálculos foram os seguintes para ambos os algoritmos:

	code1	**code2**
SST	734608.2366	726135.643
SSE	172546.637	165540.9444
SSR	562061.5996	560594.6986
r^2	0.7651	0.7720
r	0.8747	0.8786

Tabela 5: Tabela de valores teóricos calculados através da equação da reta

Nos seguintes gráficos podemos ver boxplots a alguns dos tempos recolhidos, com a probabilidade e número de exames descritos acima do gráfico. Um boxplot pretende evidenciar a dispersão de valores, fazendo uma distribuição por quartis dos vários tempos que obtivemos. Decidimos não incluir a média pois a mesma já foi referida acima. Para a realização dos gráficos foi utilizada a função boxplot do R, enviando os dados que foram recolhidos para aquele valor de exames.

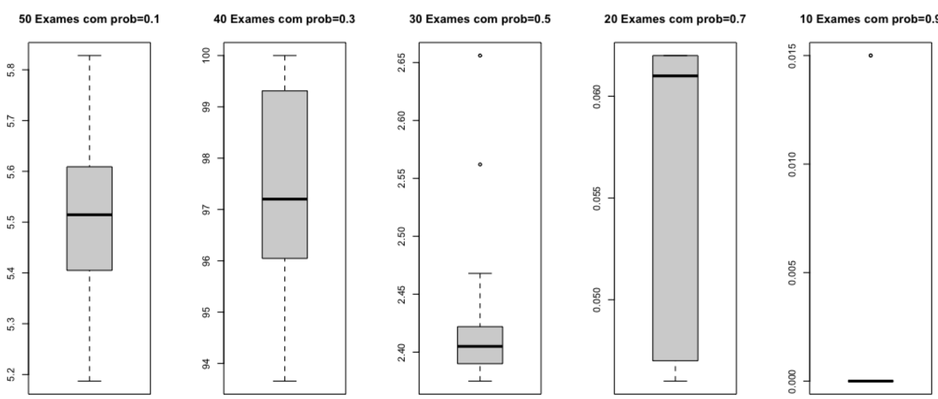

Figura 2: Gráficos boxplot dos valores do tempo para um dado número de de exames numa certa probabilidade

No que diz respeito à avaliação *time slots*/tempo de execução, foi realizado um gráfico para cada um dos algoritmos de modo a verificar como é que o tempo dependia mediante o número de *time slots* encontrados e qual dos algoritmos o melhor em termos temporais, tal como mostra a figura 3.

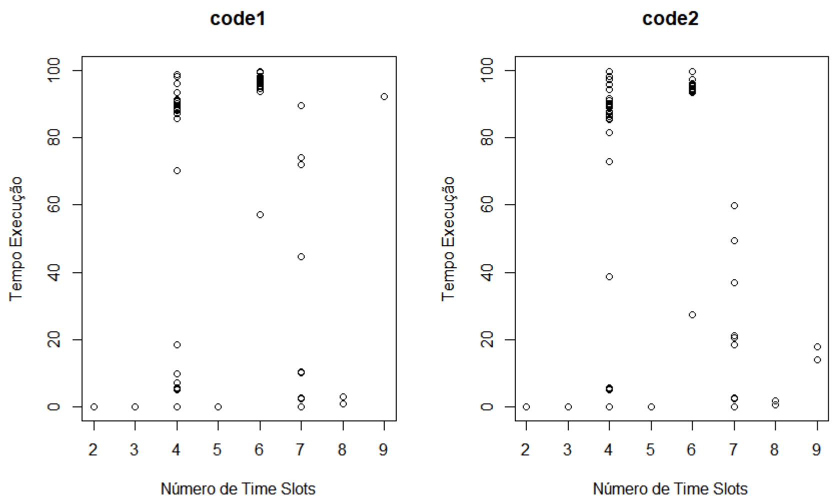

Figura 3: Gráficos de pontos de time slots por tempo de execução

5 Discussão

Esta secção terá como objetivo discutir os resultados obtidos na secção anterior (4) tentando dar resposta à *research question* apresentada no início deste documento com base na análise de como certas variáveis independentes, quando estas são variadas, fazem o algoritmo comportar-se em relação ao tempo de execução.

5.1 Tempos de execução entre algoritmos

Ao comparar os resultados obtidos nas tabelas 3 e 4 é visivel uma diferença, embora pequena, entre a velocidade média de execução dos algoritmos. No entanto não é suficiente para que haja mais casos a correr com uma média inferior ao limite máximo temporal usado nos testes. Isto faz com que esta diferença de tempo nos resultados seja mais trivial.

5.2 Tempo de execução baseado no número de exames

Mediante a análise dos gráficos da figura 1 podemos observar que os tempos de execução dos dois algoritmos com a probabilidade igual a 0.1 não apresentam grande diferença, apresentando um comportamento exponencial em ambos os algoritmo.

Resta agora analisar cada algoritmo individualmente baseado na regressão linear efetuada em cada um deles.

No algoritmo *code1* os valores de r^2 que foram obtidos foram melhores que os calculados manualmente presentes na tabela 5 (quando mais perto de 1 o valor de r^2 estiver, mais eficiente é considerado o algoritmo). Contudo, este valor ainda está longe do que seria esperado para uma experiência destas pois o valor de **0.7838** é considerado ainda muito baixo para estar próximo do ideal. Isto acontece pelo facto de existirem vários *outliers*, nomeadamente entre os valores de 40 e 60.

No caso do algoritmo *code2* os valores obtidos e teóricos de r^2 foram muito próximos um do outro, sendo o valor obtido apenas 0.0002 menor que o valor téorico. Contudo, assim como foi realçado em cima no *code1*, os valores obtidos estão longe do ideal precisamente pelo mesmo motivo de cima - apresentação de vários *outliers*. Estes *outliers* podem ter acontecido devido a uma mudança repentina de ambiente não propositada ou apenas casual.

5.3 Tempo de execução baseado na probabilidade

A partir dos gráficos presentes na figura 2 vemos que não há muita disparidade entre os tempos obtidos, o que aponta para uma estabilidade na dos algoritmos. Foram escolhidos estes valores pois são os valores mais centrais e dos quais se conseguem retirar melhores conclusões, pois outros valores para o número de exame, principalmente com o valor da prob mais alto, já se encontram demasiado perto dos tempos máximos de execução (100 segundos) e mínimos do ambiente de testagem. Também é possível observar que com uma valor de prob mais baixo os resultados têm uma distribuição aproximadamente Normal, a qual vai-se distorcendo com o aumento desse valor.

5.4 Tempo de execução e *time slots*

No que diz respeito à análise da figura 3, podemos observar que, independetemente do algoritmo de escalonamento de exames usado, o número de *time slots* será igual e apenas varia os valores do tempo final de execução, sendo esta a razão para o número de *time slots* encontrados ser considerada uma variável independente.

Em termos de comparação dos tempos para os diferentes algoritmos, podemos observar através dos gráficos da figura 3 que, quando o número de *time slots* é igual nos dois algoritmos, o *code2* tende em ter uma melhor performance que o *code1*. A dado exemplo, quando o número de *time slots* é igual a 9, o tempo de execução do *code1* ronda os 90 segundos, enquanto no *code2* os valores de tempo aproximam-se dos 20 segundos.

6 Conclusão

Após a análise dos dados recolhidos nesta experimentação na secção 5, podemos finalmente responder como é que é afetada a performance dos diferentes algoritmos de escalonamento de exames.

Com isto verificamos que, quando o número de exames aumenta, a performance dos algoritmos irá diminuir sendo em alguns casos difícil de executar as tarefas em tempo útil.

Mediante a probabilidade de dois exames terem pelo menos um aluno em comum, o efeito da performance é semelhante ao do número de exames - diminui a performance quanto maior for a probabilidade. Neste caso ainda é mais notório que os algoritmos apresentam mais dificuldades pois é mais díficil de percorrer a árvore de procura para encontrar uma solução válida.

Finalmente, no que diz respeito à diferença entre os dois algoritmos fornecidos, não existe grande desigualdade nos tempos de execução. Contudo, em certos casos, o *code2* apresenta melhores resultados que o *code1*, mas não com valores muito abaixo dos do outro algoritmo.

Referências

[1] Backtracking Algorithms, `https://bit.ly/3mCj6O9`

[2] Paquete, L., 2021, 'Slide 4 - Exploratory Data Analysis', slides 1-67

[3] Paquete, L., 2021, 'Slide 5 - Linear Regression', slides 4-16

[4] Boxplots, `https://www.statmethods.net/graphs/boxplot.html`

[5] Sum of Squares Total, Sum of Squares Regression and Sum of Squares Error, `https://bit.ly/2YwjLYq`

[6] R-Squared, `https://www.investopedia.com/terms/r/r-squared.asp`