• U C •

Metodologias Experimentais em Informática
2022/2023

Bernardo Vinagre Viseu de Simões Arzileiro | 2019216874
Filipe Rocha Ribeiro | 2019223576
Miguel António Gabriel de Almeida Faria | 2019216809

Relatório

Maximum Flow Problem - Exploratory Data Analysis and Linear Regression

1. Introdução

Este trabalho foi desenvolvido no âmbito da disciplina de Metodologias Experimentais em Informática, com o objetivo de estudar o desempenho e eficiência de certos algoritmos (Dinic, EK e MPM), que procuram resolver o *Maximum Flow Problem*. Este relatório encontra-se direcionado ao tema abordado na primeira meta, *Exploratory Data Analysis and Linear Regression*.

Para realizar esta análise, foram, inicialmente, identificadas as variáveis existentes no sistema e determinado o tipo das mesmas (dependentes ou independentes). Em seguida, geraram-se ficheiros de *input*, consoante as variáveis determinadas, e feitos testes aos vários algoritmos com esses dados. Com os resultados obtidos após a realização dos testes, foram elaborados gráficos e tabelas de forma a poder analisar melhor o desempenho de cada um dos algoritmos e fazer uma conclusão final, tendo em conta o objetivo deste primeiro trabalho.

2. Declaração do problema

O problema a explorar neste relatório é o *Maximum Flow Problem*, que consiste em encontrar o fluxo máximo que pode ser enviado de um vértice inicial para um vértice final de um grafo direcionado, sem exceder a capacidade máxima de cada arco. Para isto, foram utilizados 3 algoritmos: Dinic, Edmond-Karp (EK) e Malhotra, Pramodh-Kumar and Maheshwari (MPM).

Com esta análise, poderá observar-se a influência que o número de vértices, a probabilidade de gerar um arco e a capacidade máxima de cada arco terá no tempo de execução dos 3 diferentes algoritmos, utilizando *inputs* diversificados de maneira a poder retirar ilações mais exatas.

3. Identificação de Variáveis

O primeiro passo numa análise exploratória de dados é a identificação das diferentes variáveis presentes na experimentação. Posto isto, após analise do enunciado, obtém-se as seguintes variáveis presentes na *Figura 1 e 2*, com distinção entre as variáveis que irão influenciar os resultados dos testes (variáveis independentes) e as variáveis que permitem retirar conclusões acerca dos testes efetuados (variáveis dependentes). Apesar de existir uma variável *seed* na geração de *inputs*, não é considerada como variável independente, visto que apenas é usada para criar aleatoriedade nos casos gerados.

Variáveis Independentes	Descrição
numVertices (**n**)	Número de vértices
arcProbability (**p**)	Probabilidade de gerar um arco
maxCapacity (**r**)	Capacidade máxima de um arco

Figura 1 - Variáveis Independentes

Variáveis Dependentes	Descrição
elapsed	Tempo usado pelo CPU
maxFlow	Valor do *maximum flow* (ou -1)

Figura 2 - Variáveis Dependentes

4. Cenário de Experimentação

Para iniciar o nosso cenário, foi necessário gerar dados para a experimentação, sendo que para esse efeito nos foi fornecido o gerador "gen.py". Para tornar o processo de criação de *inputs* mais eficiente, este gerador foi adaptado, criando um grafo para cada combinação de valores das variáveis independentes. Cada geração de dados teve uma *seed* diferente (foi atribuído o valor do *clock* do CPU).

No que diz respeito aos dados de input, foram realizadas combinações entre as diferentes variáveis independentes. Assim, relativamente ao número de vértices, os seus valores variam entre 250 e 2500, com incrementos de 250 (10); para a probabilidade de gerar arco, estes variam de 20% a 100%, de 20% em 20% (5); quanto à capacidade máxima de cada arco, variam de 50 a 200, com aumentos de 50 (4). Para além disso, para cada uma das combinações geraram-se 3 grafos diferentes, de maneira a evitar que casos anómalos pudessem influenciar os resultados de maneira negativa. Por fim, para cada um dos algoritmos, os testes foram executados 3 vezes.

Resumindo, testou-se 10 valores de número de vértices, cada um com 5 probabilidades distintas e 4 valores de arco diferentes; para cada um desses testes foram gerados 3 grafos diferentes e cada um dos ficheiros foi executado 3 vezes (cada ficheiro possui uma *seed* única, como referido anteriormente). Conclui-se então que, para cada algoritmo obteve-se um total de 10*5*4*3*3 = 1800 resultados de execução. A *Figura 3* representada acima ilustra a forma como cada uma das variáveis se relaciona entre si, bem como uma ideia geral dos tempos obtidos em relação a cada uma delas.

Para realizar a testagem, elaborou-se um outro programa "run_inputs.py" que permite facilitar o processo de testagem dos inputs, percorrendo todos os ficheiros, executando-os em cada algoritmo e guardando todos os valores de *input* e *output* de cada *run* num ficheiro ".csv". Para evitar execuções muito demoradas, limitamos o tempo máximo de CPU a 10 segundos. De notar que nas análises

seguintes o valor de *maximum flow* encontrado vai ser negligenciado, uma vez que para saber se o problema não foi resolvido se pode verificar se o teste alcançou o tempo máximo de CPU definido.

Com o intuito de maximizar a precisão dos resultados, a experimentação foi realizada num único computador com as seguintes especificações:

- OS - Windows 10 (x64)
- CPU - Intel(R) Core(TM) i7-7700HQ @ 2.80GHz 2.81 GHz
- RAM - 32.0 GB 2400MHz
- GPU - NVIDIA GTX 1070

5. Análise Exploratória de Dados

Para melhor compreender qual dos algoritmos tem uma melhor *performance,* foram gerados gráficos *boxplot* dos tempos de execução para cada um dos algoritmos, que permite obter uma compreensão resumida da distribuição dos dados. Além disso, através dos gráficos, representados na *Figura 4*, consegue-se perceber a dispersão, valores atípicos e tendência dos tempos médios.

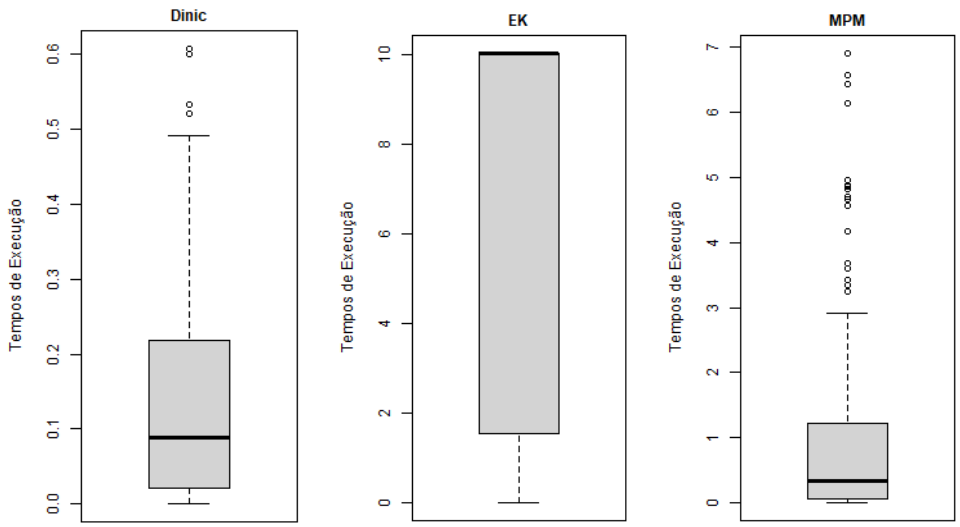


Figura 3 - Gráficos boxplot representativos dos tempos de execução de cada um dos algoritmos

Como se pode constatar, os algoritmos Dinic e MPM são bastante mais rápidos que o EK, que atinge o tempo máximo de execução bastantes vezes, possuindo uma grande dispersão dos valores. Apesar de possuir alguns *outliers*, os tempos obtidos usando MPM estão maioritariamente abaixo dos 3 segundos, no entanto possuindo uma mediana baixa, perto dos 0.5 segundos. Já no que diz respeito ao algoritmo Dinic, os seus tempos de execução não passam dos 0.5 segundos (exceto *outliers*), com a mediana dos valores a rondar os 0.1 segundos.

Sabendo então os algoritmos que aparentam ser mais rápidos, pretende-se verificar que variáveis independentes possuem maior influência nos tempos de execução, criaram-se os seguintes gráficos da *Figura 4*, que possui uma comparação entre os algoritmos, isolando cada uma das variáveis independentes relativamente ao tempo médio das execuções que a possuem como parâmetro.

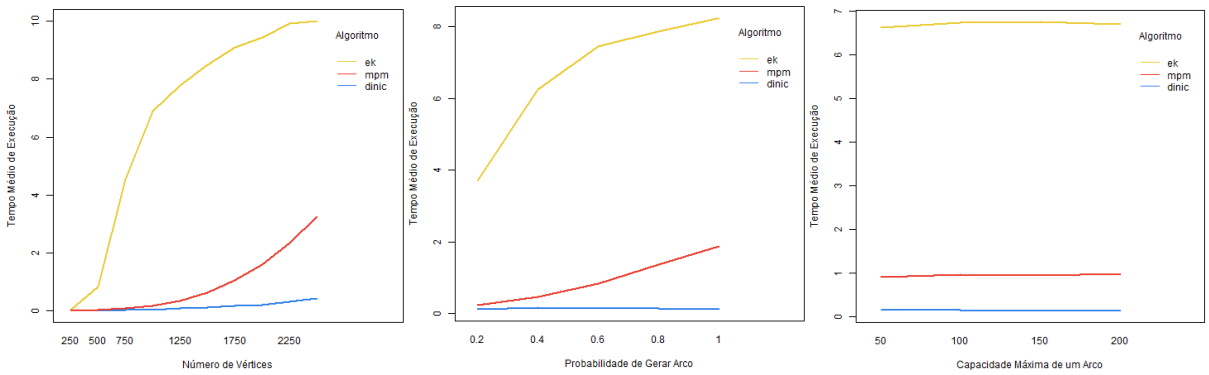


Figura 4 - Comparação dos algoritmos fixando cada uma das variáveis independentes

Observando a *Figura 4*, verifica-se que para os três algoritmos, quanto maior o número de vértices e a probabilidade de gerar arco, maior é o tempo médio de execução. Observando mais atentamente, podemos ver que, no que diz respeito ao número de vértices, o algoritmo Dinic tem um comportamento (aproximadamente) linear, o MPM apresenta um comportamento exponencial e o EK um comportamento logarítmico. Em relação à probabilidade de gerar arco, o algoritmo EK apresenta novamente um comportamento logarítmico, MPM um comportamento linear e Dinic um comportamento constante. Quanto à capacidade máxima, todos os algoritmos apresentam comportamento constante.

Conclui-se que o valor da capacidade máxima de um arco (r) é pouco significante em relação às outras duas variáveis usadas. Desta forma, as tabelas produzidas apresentam os tempos médios de execução, tendo em conta todas as combinações possíveis entre a probabilidade de gerar um arco (p) e o número de vértices (n), sendo que, devido à explicação acima, foi feita uma média dos valores que envolviam a capacidade máxima do arco. Importante referir que valores que atinjam o máximo tempo de CPU definido não serão tidos em conta na análise dos resultados (embora possam estar representados).

Dinic	250	500	750	1000	1250	1500	1750	2000	2250	2500
0,2	0,0009345	0,0039807	0,0105761	0,0243696	0,0556397	0,0841255	0,142971	0,1688543	0,2780567	0,4749887
0,4	0,0015065	0,0072544	0,0214843	0,0563091	0,0822339	0,1560381	0,1935264	0,2804886	0,4183118	0,42389
0,6	0,0017685	0,0061174	0,0293034	0,0482362	0,1132924	0,1182024	0,2029072	0,1990201	0,2796563	0,4130233
0,8	0,0015014	0,0071219	0,025903	0,0485345	0,0705376	0,0955203	0,1527348	0,2300886	0,3124036	0,4346848
1	0,0011445	0,0068502	0,0201684	0,0390903	0,0680536	0,0946672	0,1318858	0,1796723	0,2751724	0,3389405

EK	250	500	750	1000	1250	1500	1750	2000	2250	2500
0,2	0,0080924	0,0522955	0,2673646	0,7646154	1,5758867	2,4339592	5,3400892	7,0775983	9,5845122	10,007958
0,4	0,0180283	0,180328	0,9492084	4,0758886	7,2407814	10,00295	10,008958	10,013786	10,020881	10,022225
0,6	0,0381558	0,4305647	4,1193733	9,6740475	10,006472	10,012278	10,017289	10,020756	10,022939	10,020183
0,8	0,0644001	1,1276864	7,3789	10,007611	10,010306	10,054583	10,073472	10,017608	10,014519	10,018453
1	0,0909687	2,2836319	9,9465033	10,0175	10,014833	10,018028	10,017667	10,011278	10,013378	10,014833

MPM	250	500	750	1000	1250	1500	1750	2000	2250	2500
0,2	0,0026326	0,0116566	0,0317566	0,0560155	0,12137	0,1810755	0,2591899	0,3427483	0,4827111	0,7467266
0,4	0,0037178	0,0178844	0,0533937	0,1245926	0,2035899	0,3621094	0,4865493	0,7417607	1,1083899	1,4108108
0,6	0,0047652	0,0247019	0,0857501	0,1617535	0,3393623	0,5319074	0,9789461	1,3171856	1,9815203	2,8157886
0,8	0,0050304	0,0322369	0,1025134	0,2136004	0,4553937	0,8291901	1,3875942	2,3499083	3,4979522	4,7624964
1	0,0053783	0,0377988	0,1197242	0,3160219	0,5341728	1,2307886	2,1191931	3,2105697	4,6454172	6,5160989

Figura 5 - Tempos médios para os diferentes algoritmos

6. Regressões Lineares

As regressões lineares foram calculadas para cada um dos algoritmos, tendo sido aplicadas não só para os dados originais, mas também para as suas respetivas transformações quadráticas e cúbicas, de maneira a observar qual das possibilidades se adequa de melhor forma a cada um deles. A tabela com os dados relativos a todas as regressões aplicadas pode ser consultada na *Figura 9*.

Uma vez que a aplicação das regressões lineares para cada uma das combinações entre número de vértices e probabilidade de gerar arco iria causar dificuldades na análise dos gráficos devido ao seu elevado número, foram estabelecidas algumas probabilidades específicas para cada um dos algoritmos, tendo em conta os resultados obtidos no gráfico da variação dos tempos médios de execução relativamente à probabilidade de gerar arco da *Figura 4*. Deste modo:

- para o algoritmo Dinic, visto que os valores de tempo permanecem constantes independentemente da probabilidade, escolheu-se o valor médio, 0.6;
- para o algoritmo EK, visto que os valores de tempo não aparentam ter nenhuma relação óbvia relativamente à probabilidade, escolheram-se 3 pontos igualmente dispersos entre si (de modo a conseguir envolver mais casos), 0.2, 0.6 e 0.8;
- para o algoritmo MPM, visto que os valores de tempo crescem de forma linear relativamente à probabilidade, escolheu-se um valor inicial e final (definindo uma reta), 0.2 e 0.8, respetivamente.

As regressões aplicadas nos tempos obtidos usando Dinic encontram-se ilustradas na *Figura 6*, onde se pode notar que a regressão linear com transformação quadrática ou cúbica se adequa a este algoritmo. Recorrendo à tabela das informações, verifica-se que a transformação cúbica conseguiu um valor de R^2 muito ligeiramente superior (0.87712637).

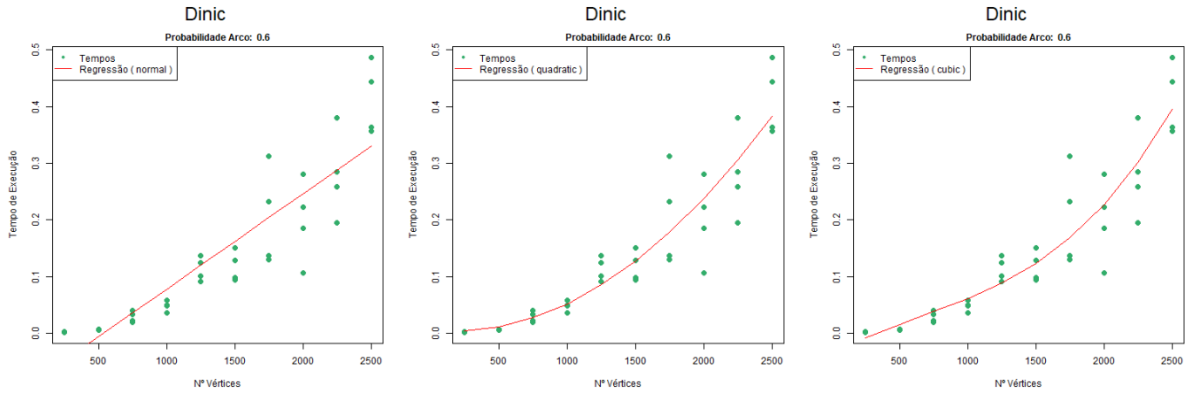


Figura 6 - Regressões lineares (sem e com transformações) aplicadas relativamente aos tempos de execução usando Dinic

Em relação aos gráficos das regressões lineares aplicadas aos tempos usando o algoritmo EK, presentes na *Figura 7*, o uso da transformação quadrática permitiu um melhor ajuste dos dados quando a probabilidade é 0.2, onde o valor de R^2 se encontra próximo de 1 (0.971556). Importante salientar que como para as outras probabilidades os tempos de execução atingem rapidamente o tempo limite de execução, os resultados não são tão bons, mesmo assim a regressão cúbica continuou a ser aquela com melhores valores.

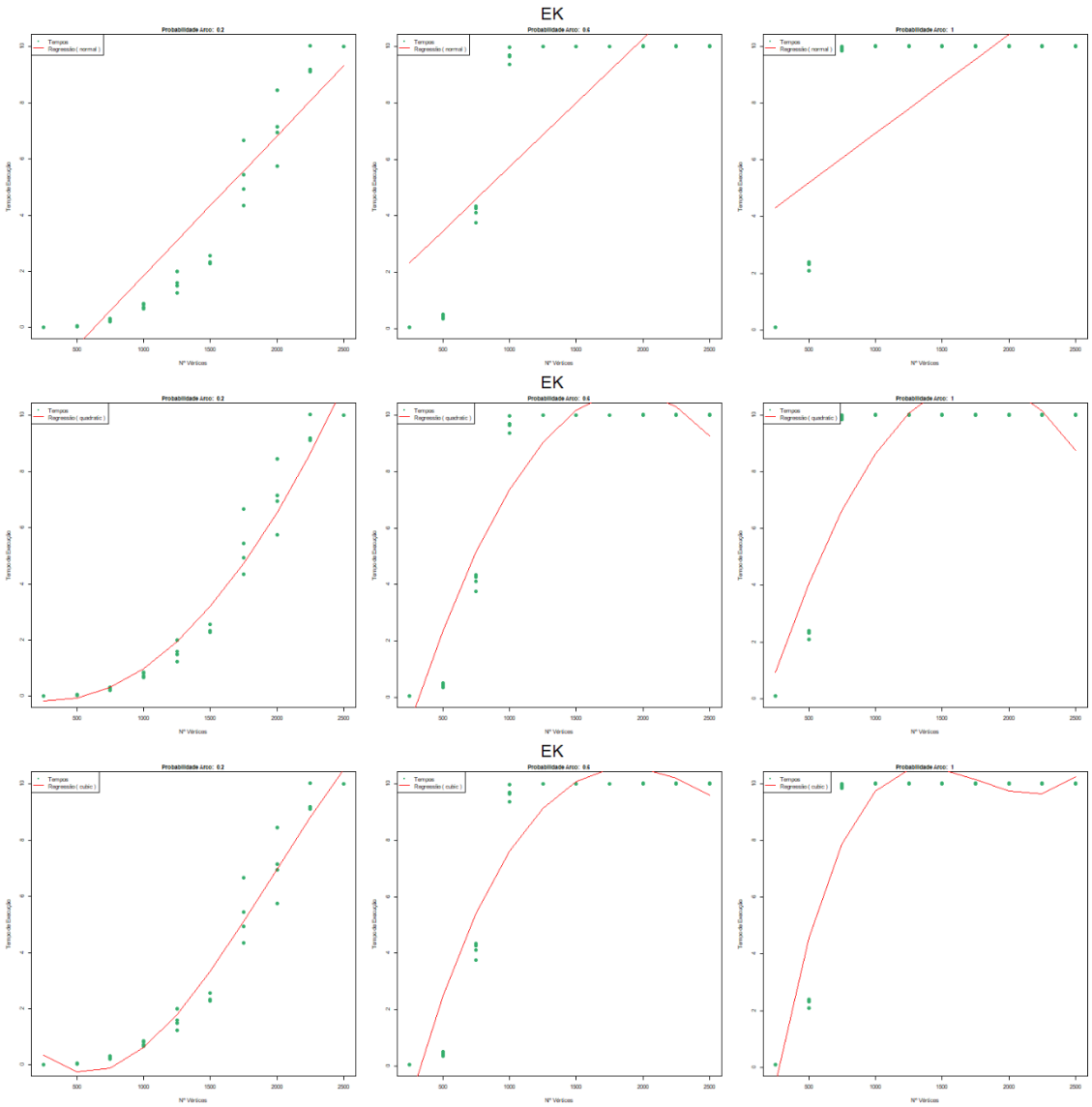
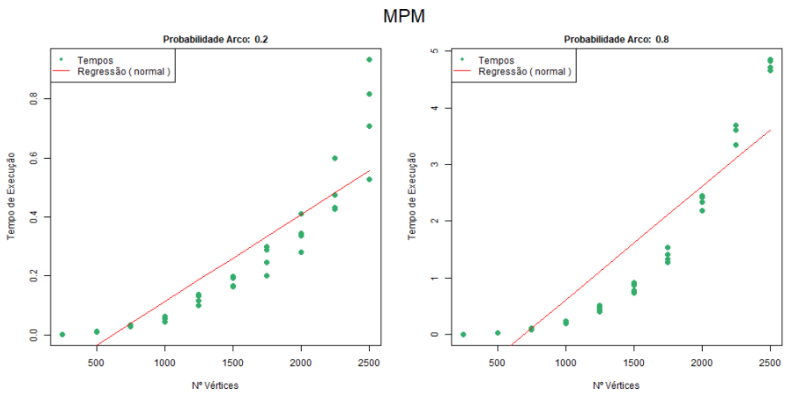


Figura 7 - Regressões lineares (sem e com transformações) aplicadas relativamente aos tempos de execução usando EK

Usando o algoritmo MPM, todas as regressões lineares, representadas de seguida na *Figura 8*, apresentaram bons resultados. No entanto, aquela em que o valor de R^2 mais se aproximou de 1 nas duas probabilidades usadas foi novamente a que recorreu à transformação cúbica.



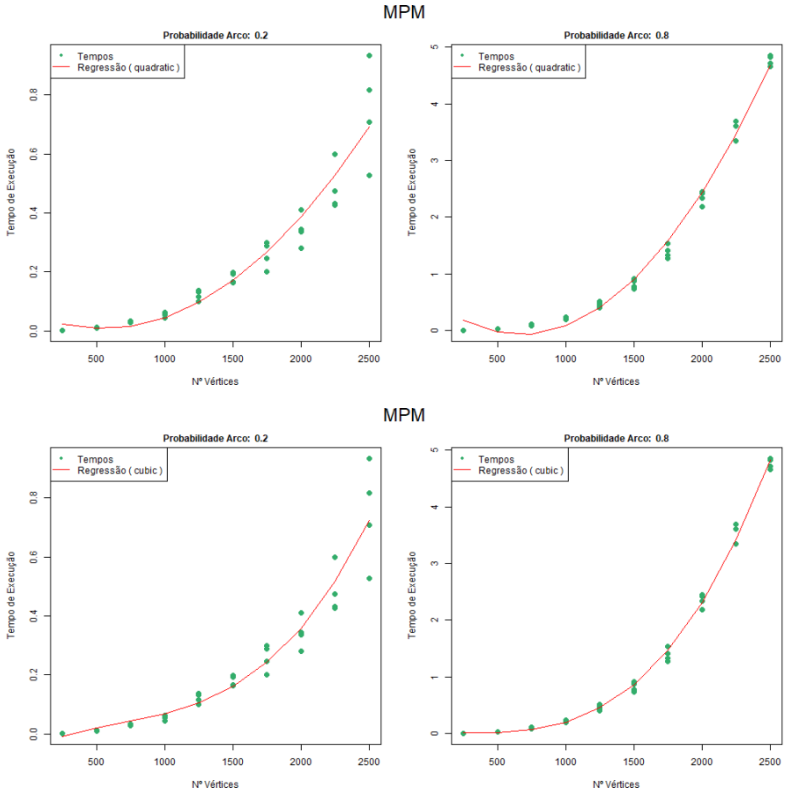


Figura 8 - Regressões lineares (sem e com transformações) aplicadas relativamente aos tempos de execução usando MPM

Algoritmo	Transformação	Probabilidade Arco	Equação	R^2	R
Dinic	Normal	0.6	y = -8.9966e-02 + 1.6809e-04 x	0.81616209	0.9034169
	Quadrática	0.6	y = 6.0808e-03 + -2.4008e-05 x + 6.9852e-08 x^2	0.87254317	0.93410019
	Cúbica	0.6	y = -3.8075e-02 + 1.3265e-04 x + -6.6012e-08 x^2 + 3.2937e-11 x^3	0.87712637	0.93655025
EK	Normal	0.2	y = -3.1341e+00 + 4.9784e-03 x	0.88809174	0.94238619
		0.6	y = 1.1853e+00 + 4.5461e-03 x	0.666553	0.81642697
		1	y = 3.449e+00 + 3.4853e-03 x	0.49386782	0.7027573
	Quadrática	0.2	y = 5.1931e-03 + -1.3002e-03 x + 2.2831e-06 x^2	0.96280387	0.9812257
		0.6	y = -4.8173e+00 + 1.6551e-02 x + -4.3655e-06 x^2	0.91241168	0.95520243
		1	y = -2.7931e+00 + 1.597e-02 x + -4.5397e-06 x^2	0.82902663	0.91050899
	Cúbica	0.2	y = 1.7377e+00 + -7.4468e-03 x + 7.614e-06 x^2 + -1.2923e-09 x^3	0.971556	0.9856754
		0.6	y = -5.9456e+00 + 2.0554e-02 x + -7.8371e-06 x^2 + 8.416e-10 x^3	0.91575256	0.95694961
		1	y = -7.8954e+00 + 3.4072e-02 x + -2.0239e-05 x^2 + 3.8059e-09 x^3	0.91515614	0.95663793
MPM	Normal	0.2	y = -1.8369e-01 + 2.962e-04 x	0.81004906	0.90002726
		0.8	y = -1.3767e+00 + 1.993e-03 x	0.82651939	0.90913112
	Quadrática	0.2	y = 6.2557e-02 + -1.9629e-04 x + 1.7909e-07 x^2	0.92849846	0.96358625
		0.8	y = 5.6253e-01 + -1.8856e-03 x + 1.4104e-06 x^2	0.99208944	0.99603686
	Cúbica	0.2	y = -5.007e-02 + 2.0328e-04 x + -1.6745e-07 x^2 + 8.4011e-11 x^3	0.93802849	0.96851871
		0.8	y = 3.327e-02 + -7.8771e-06 x + -2.181e-07 x^2 + 3.9478e-10 x^3	0.99683257	0.99841503

Figura 9 - Tabela dos valores referentes às regressões lineares

7. Conclusões

Através da análise dos dados, conclui-se que o algoritmo com melhor performance é o Dinic e o com pior é o EK, sendo o MPM mediano. Verifica-se que à medida que o número de vértices aumenta, também o tempo de execução cresce, para todos os algoritmos. Também para probabilidades maiores de geração de arcos, os algoritmos apresentam tempos superiores, apenas com uma particularidade no Dinic, onde apresenta tempos ligeiramente menores para uma probabilidade de 1.

Realizada a análise das regressões lineares, observa-se que os dados obtidos pelos diferentes algoritmos se ajustam melhor à regressão linear com transformação cúbica. Para os algoritmos Dinic e MPM, constata-se que a regressão com transformação quadrática também tem uma boa adaptação aos dados, pelo que não conseguimos ter a certeza se a transformação cúbica apenas obtém melhores valores por se ajustar melhor a eventuais *outliers*.

Em seguida o objetivo será, através dos resultados obtidos, realizar hipóteses com base no que foi exposto nesta análise de dados, a ser testadas na meta 3.

8. Referências

[1] CORRELATION and Simple Linear Regression with R. [*S. l.*], 3 jun. 2016. Disponível em: https://www.youtube.com/watch?v=zfLttx6k_W0&ab_channel=GillesLamothe. Acesso em: 4 nov. 2022.

[2] HOW to Create Interaction Plot in R. [*S. l.*], 26 jan. 2022. Disponível em: https://www.geeksforgeeks.org/how-to-create-interaction-plot-in-r/. Acesso em: 3 nov. 2022.

[3] XTABS: Cross Tabulation. [S. l.], 2016. Disponível em: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/xtabs. Acesso em: 3 nov. 2022.