Análise Exploratória de Dados e Regressão Linear

André Martins 2006130025, Cláudia Campos 2018285941, Dário Félix 2018275530

1 Introdução

Este trabalho prático estuda a *performance* de três algoritmos, que resolvem o problema de *maximum-flow* e cuja complexidade é conhecida à *priori*, face às suas possíveis variáveis independentes, tais como: o número de vértices, o número de arcos, a capacidade de cada arco, a topologia do grafo, entre outros.

Para um dado grafo $\boldsymbol{G = (V, E)}$, onde $\boldsymbol{V}$ corresponde ao conjunto de vértices e $\boldsymbol{E}$ ao conjunto de arcos, o algoritmo *Dinic* possui uma complexidade temporal de $\boldsymbol{O(|V|^2|E|)}$, o algoritmo *Edmond-Karp* (EK) de $\boldsymbol{O(|E|^2|V|)}$ e o algoritmo *Malhota, Pramodh-Kumar e Maheshwari* (MPM) de $\boldsymbol{O(|V|^3)}$.

1.1 Definição do Problema

Os problemas de fluxo máximo tentam determinar o fluxo máximo de uma determinada rede, com um ponto único de entrada, um ponto único de saída e múltiplos ramos com capacidade máxima. Dados 3 algoritmos que fornecem uma solução para problemas de fluxo máximo, este trabalho visa responder às seguintes questões:

- Perceber como o número de vértices e o número de arestas (definido pela probabilidade) influenciam o tempo de execução de cada algoritmo? Quão forte é a sua correlação?
- Quão próximo ficou dos valores esperados para a complexidade temporal em cada um dos algoritmos?
- Em que cenários os diferentes algoritmos têm melhor *performance*?

1.2 Variáveis do Problema

Para as nossas experiências consideramos o número de vértices, $\boldsymbol{v}$, e o número de arestas, $\boldsymbol{e}$, como variáveis independentes, fixando as restantes, dado que estas, conforme a complexidade teórica de cada algoritmo, são as que mais influenciam a variável dependente, isto é, o tempo de execução, $\boldsymbol{t}$, em segundos.

2 Métodos

2.1 Equipamento

Todas as experiências e medições foram realizadas na mesma máquina, cuja descrição técnica se encontra na Tabela 1. A máquina escolhida é vantajosa, pois tem muito poucos processos ativos, incluído as do sistema operativo, minimizando a tendência não determinística do tempo de execução caso fosse executada num computador pessoal normal [1].

Sistema Operativo	Raspberry Pi OS (baseado no Debian), Linux version 5.15.61-v8+
Memória	4GB LPDDR4-3200 SDRAM
Processador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Placa Gráfica	VideoCore VI
Disco	SD Card 32GB 100MB/s

Tabela 1: Especificações da máquina [2]

2.2 Compilação dos Algoritmos

Todos os algoritmos foram compilados conforme o *Makefile* disponibilizado.

2.3 Parameterização

2.3.1 Sementes Aleatórias

A cada medição utilizou-se uma *seed* diferente, ordenadamente, no sentido de minimizar a sua influência e de tornar a execução aleatória, comecando por 0.

2.3.2 Número de Medições (Repetições)

Segundo o Teorema do Limite Central, são necessárias pelo menos 30 amostras para poder efetuar uma análise segundo a distribuição normal (*Z-Test*). Por isso, para cada combinação de fatores (exceto a *seed*), realizámos **30 repetições**.

2.3.3 Intervalos dos Fatores

Foram realizados diversos testes para garantir que os intervalos de valores escolhidos são os mais adequados para a análise do problema e para a duração da nossa experiência.

Estes testes mostraram-nos que para grafos com pequeno número de vértices e de arestas o tempo de execução é demasiado pequeno, muita das vezes inferior a um milissegundo. O gerador de *inputs*, programado em *python*, produz uma exceção quando excede o nível máximo de recursão, isto é, quando geramos grafos com número de vértices superior a 1000. Para números de vértices bastante elevados, probabilidades inferiores a 0.1 não produzem grafos válidos. Os algoritmos demoraram bastante mais tempo a executar para grafos com grande número de vértices e de probabilidade.

Por isso, definimos o conjunto presente na Equação 1 para o número de vértices, $\boldsymbol{v}$, e o conjunto presente na Equação 2 para a probabilidade, $\boldsymbol{p}$.

$$V = \{50n + 50 \mid n \in \mathbb{N} \land n \geq 1 \land n \leq 19\} \tag{1}$$

$$P = \{0.1n \mid n \in \mathbb{N} \land n \geq 1 \land n \leq 7\} \tag{2}$$

2.3.4 Tempo Máximo de Execução

Decidiu-se escolher **10 segundos** como tempo máximo de execução por vários motivos: para o intervalo de fatores escolhido, especialmente em relação aos limites superiores, verificou-se em testes preliminares que seria o tempo suficiente para a execução dos algoritmos, e, além disso, para se poder obter os resultados em tempo útil, considerando o número de repetições e de medições definidos a partir da variação dos parâmetros.

3 Resultados e Discussão

3.1 Pré-processamento

3.1.1 *Timeout*

Para as execuções que não terminam no tempo definido, decidiu-se considerar à mesma essas medições, **atribuindo-lhes o tempo máximo de execução**. Decidimos manter estas execuções porque ocorrem apenas com 1 algoritmo (EK) e poderá ajudar a avaliar a performance do algoritmo.

3.1.2 *Outliers*

Dicidiu-se **ignorar eventuais *outliers***, uma vez que a máquina escolhida já minimiza a tendência não determinística do tempo de execução, e assim, as experiências são realizadas nas mesmas condições, portanto, todos os resultados são tendencialmente válidos.

3.1.3 Conversão de Unidades

Durante a análise exploratória de dados, para uma melhor visualização dos resultados, calculámos a média para cada par de fatores: vértices ($\boldsymbol{v}$) e probabilidade ($\boldsymbol{p}$).

No entanto, ao efetuar este cálculo, deixamos de possuir o número de arcos para cada repetição. Para obtermos a média do número de arcos, multiplicamos a probabilidade pelo número máximo de vértices de um grafo não direcionado, conforme ilustrado na Equação 4.

$$\binom{n}{r} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2} \tag{3}$$

$$|E| = p\frac{|V|(|V|-1)}{2} \tag{4}$$

3.2 Análise Exploratória de Dados

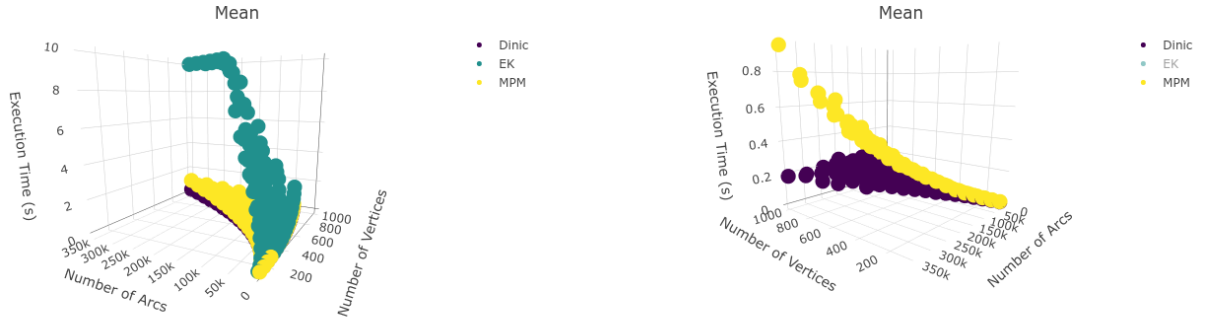


Figura 1: Comportamento dof tempo de execução em função do número de vértices e do número de arestas, à esquerda estão representados os três algoritmos, e à direita apenas o Dinic e o MPM.

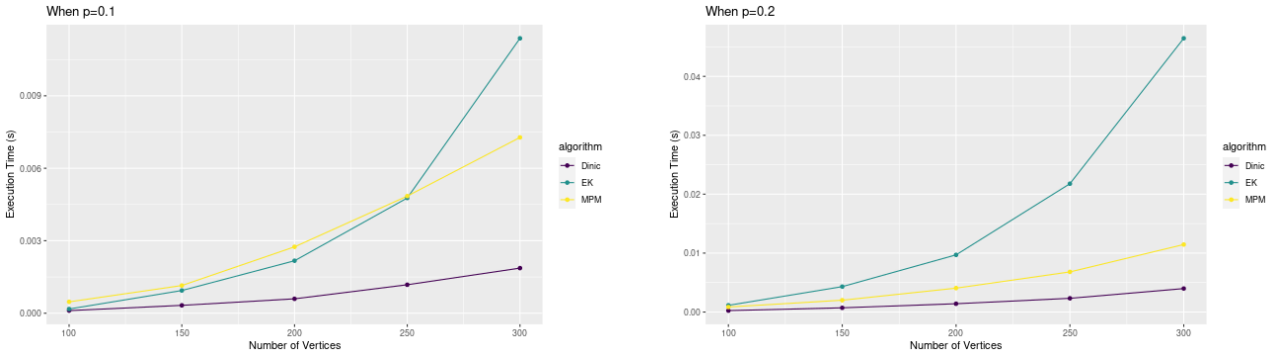


Figura 2: Evolução do tempo de execução em função do número de vértices, até 300 vértices, dos três algoritmos, à esquerda com a probabilidade fixada em 0.1, e à direita fixada em 0.2.

Os resultados obtidos vão ao encontro daquilo que era esperado, com base na complexidade temporal teórica, Equação 5. De facto, e para os intervalos dos parâmetros testados, Subsubseção 2.3.3, o algoritmo **EK** tem uma degradação no tempo de execução muito mais rápida do que os restantes algoritmos, pois no cálculo da sua complexidade temporal é utilizado o quadrado das arestas, que depende do número de vértices conforme a Equação 4, evidenciando ser um fator mais importante do que o número de vértices.

Pela Figura 1, é possível afirmar que o algoritmo **Dinic** é o que teve a melhor *performance* e consistentemente para todos os intervalos dos fatores testados.

O algoritmo **MPM**, genericamente, teve um desempenho ligeiramente pior que o algoritmo **Dinic**, mas significativamente longe do mau desempenho do algoritmo **EK**, sobretudo à medida que o número de vértices sobe e se aproxima do limite superior do intervelo testado. Ainda assim, e ainda sobre o **MPM**, destaca-se na Figura 2, a pior *performance* dos três, para um número de vértices inferiores a 250, com uma probabilidade fixada nos 10%.

3.3 Regressão Linear

Na análise de regressão, verificamos, para cada algoritmo, se o tempo de execução varia conforme a sua complexidade teórica.

Se analisarmos a complexidade dos algoritmos estudados, especificadas na Seção 1, averiguamos que as suas complexidades, para além de dependerem de duas variáveis independentes, $|\boldsymbol{V}|$ e $|\boldsymbol{E}|$, não são lineares, isto é, correspondem a expressões polinomiais.

Por estes motivos, decidimos aplicar a cada algoritmo, uma regressão linear multivariada com transformações aplicadas à sua variável de resposta e às suas variáveis independentes.

3.3.1 Linear Multivariada

$$
\begin{array}{lll}
O(|V|^2|E|) & O(|E|^2|V|) & O(|V|^3) \\
t = v^2 e & t = v e^2 & t = v^3 \\
v^2 = \dfrac{t}{e} & e^2 = \dfrac{t}{v} & \sqrt[3]{t} = v \\
v = \sqrt{\dfrac{t}{e}} & e = \sqrt{\dfrac{t}{v}} & \\
v = \dfrac{\sqrt{t}}{\sqrt{e}} & e = \dfrac{\sqrt{t}}{\sqrt{v}} & \\
\sqrt{t} = v\sqrt{e} & \sqrt{t} = e\sqrt{v} &
\end{array}
\tag{5}
$$

Para sabermos qual a transformação a aplicar a cada regressão linear, calculámos a inversa de cada expressão dada pela respetiva complexidade.

Como demonstrado na Equação 5, para calcular a inversa da complexidade do algoritmo **Dinic**, resolvemos a equação em ordem a $\boldsymbol{v}$ e obtivemos o resultado $\sqrt{t} = v\sqrt{e}$. Por isso, aplicámos a raiz quadrada à variável de resposta $\boldsymbol{t}$ e à variável independente $\boldsymbol{e}$.

De forma análoga, para calcular a inversa da complexidade do algoritmo **EK**, resolvemos a equação em ordem a $\boldsymbol{e}$, obtivemos a expressão $\sqrt{t} = e\sqrt{v}$ e aplicámos a raiz quadrada à variável de resposta $\boldsymbol{t}$ e à variável independente $\boldsymbol{v}$.

Para calcular a inversa da complexidade do algoritmo **MPM**, resolvemos a equação em ordem a $\boldsymbol{v}$. Obtivemos o resultado $v = \sqrt[3]{t}$ e, por isso, aplicámos a raiz cúbica à variável de resposta $\boldsymbol{t}$.

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2 \tag{6}$$

$$\sqrt{t} = b_0 + b_1 v + b_2\sqrt{e} + b_3 v\sqrt{e} \tag{7}$$

$$\sqrt{t} = b_0 + b_3 v\sqrt{e} \tag{8}$$

Se considerarmos uma regressão linear com duas variáveis independentes, $\boldsymbol{x_1}$ e $\boldsymbol{x_2}$, e a interação entre estas obtemos a expressão da Equação 6.

A complexidade dos algoritmos **Dinic** e **EK** apenas considera o termo resultante da interação entre as variáveis independentes e, por isso, no modelo de regressão linear apenas considerámos este termo, Equações 7 e 8.

```
Call:
lm(formula = sqrt(time) ~ vertices:sqrt(arcs), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.31717 -0.04337 -0.01228  0.03910  0.33347

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         6.842e-02  1.765e-03   38.76   <2e-16 ***
vertices:sqrt(arcs) 8.453e-07  8.100e-09  104.36   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07433 on 3988 degrees of freedom
Multiple R-squared:  0.732, Adjusted R-squared:  0.7319
F-statistic: 1.089e+04 on 1 and 3988 DF,  p-value: < 2.2e-16
```

(a) *Dinic*

```
Call:
lm(formula = sqrt(time) ~ arcs:sqrt(vertices), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.61358 -0.29402 -0.01406  0.25405  1.19877

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         4.445e-01  8.127e-03   54.69   <2e-16 ***
arcs:sqrt(vertices) 3.914e-07  2.555e-09  153.20   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3899 on 3988 degrees of freedom
Multiple R-squared:  0.8548,    Adjusted R-squared:  0.8547
F-statistic: 2.347e+04 on 1 and 3988 DF,  p-value: < 2.2e-16
```

(b) *EK*

Figura 3: Resultados da regressão linear multi-variada

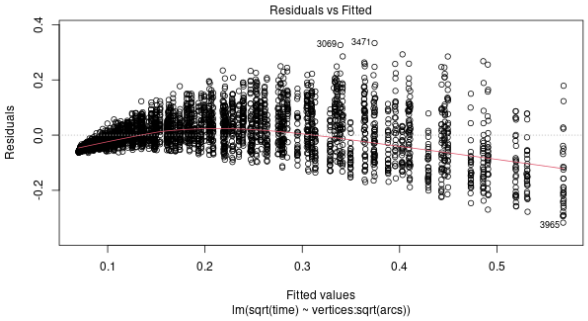
```
Call:
lm(formula = time^(1/3) ~ vertices, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.33105 -0.05015  0.00322  0.05728  0.29535

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.859e-02  3.281e-03   11.76   <2e-16 ***
vertices    7.573e-04  5.340e-06  141.83   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09237 on 3988 degrees of freedom
Multiple R-squared:  0.8345,    Adjusted R-squared:  0.8345
F-statistic: 2.012e+04 on 1 and 3988 DF,  p-value: < 2.2e-16
```

(c) *MPM*

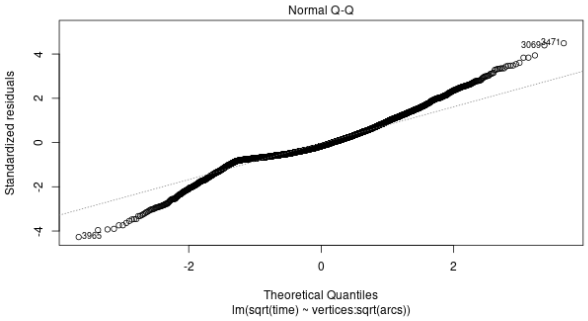
Figura 3: Resultados da regressão linear multi-variada

Segundo os resultados das regressões lineares presentes na Figura 3, obtivemos um $\bar{R}^2$ de 0.7319, 0.8547 e 0.8345 para os algoritmos **Dinic**, **EK** e **MPM** respetivamente. Estes valores são relativamente elevados para os algoritmos **EK** e **MPM**, mostrando que mais de 80% da variação do sistema transformado é explicada pelo respetivo modelo de regressão. No entanto, para o algoritmo **Dinic**, apenas 73.19% da variação é explicada pelo modelo. Ou seja, podemos concluir que existem variáveis independentes relevantes que não foram incluídas no modelo de regressão.

No entanto, o $\bar{R}^2$ não é suficiente para analisar se os termos considerados são os que mais influenciam a variável de resposta e se existem mais variáveis independentes que não foram consideradas. Por isso, analisámos também os pressupostos de linearidade, homocedasticidade e normalidade dos resíduos para cada regressão.

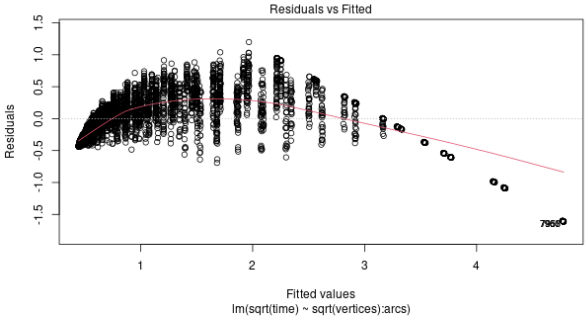


(d) *Residuals vs Fitted*

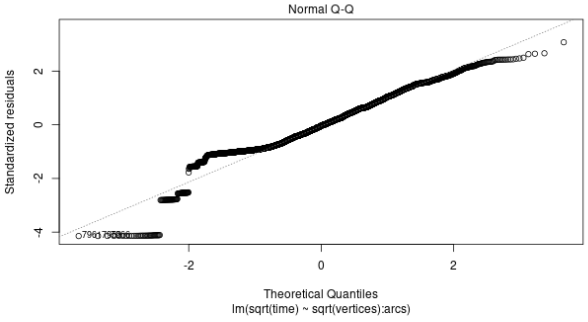


(e) *Normal QQ-Plot*

Figura 4: Pressupostos para o algoritmo Dinic

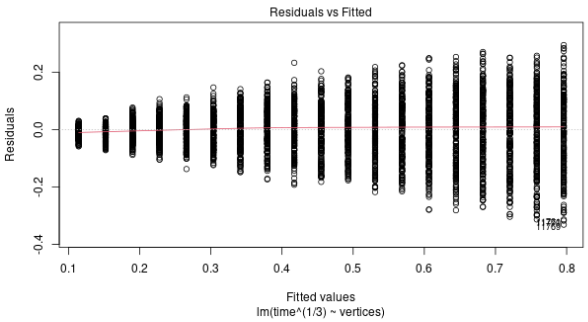


(a) *Residuals vs Fitted*

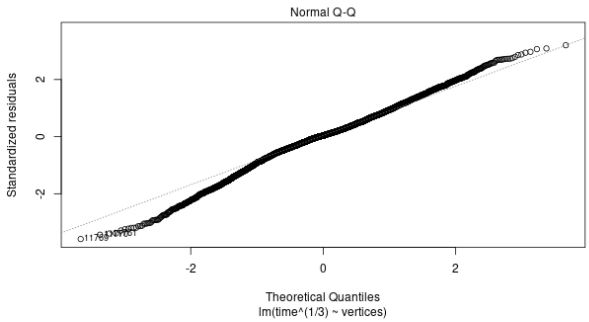


(b) *Normal QQ-Plot*

Figura 5: Pressupostos para o algoritmo EK



(a) *Residuals vs Fitted*



(b) *Normal QQ-Plot*

Figura 6: Pressupostos para o algoritmo MPM

Para os algoritmos **Dinic** e **EK**, apesar da transformação aplicada às variáveis, a relação entre a variável de resposta e a interação entre as variáveis independentes não é linear, uma vez que a média dos valores dos resíduos não é próxima de zero para cada valor estimado. Tal se pode verificar através da linha horizontal das Figuras 4d e 5a. No entanto, para o algoritmo **MPM**, na Figura 6a, a média dos valores dos resíduos é muito próxima de zero, indicando que a relação da variável de resposta transformada com a variável independente é linear.

Para além do pressuposto da linearidade, o pressuposto da homocedasticidade, para os três algoritmos, não se verifica dado que os resíduos não estão uniformemente distribuídos ao longo do eixo dos x, ou seja, a variância dos resíduos não é constante para todos os valores das variáveis independentes.

Por fim, o pressuposto da normalidade também não se verifica para todos os algoritmos. Observando as Figuras 4e, 5b e 6b, notamos que existem resíduos bastante afastados dos quantis extremos da distribuição normal, o que indica que os resíduos não são normalmente distribuídos.

3.3.2 Polinomial

$$y_{dinic} = b_0 + b_1 v + b_2 v^2 + b_3 e + b_4 ve + b_5 v^2 e \tag{9}$$

$$y_{ek} = b_0 + b_1 v + b_2 e + b_3 e^2 + b_4 ve + b_5 v e^2 \tag{10}$$

$$y_{mpm} = b_0 + b_1 v + b_2 v^2 + b_3 v^3 + b_4 e + b_5 ve + b_6 v^2 e \tag{11}$$

Considerar apenas a complexidade omite termos de menor grau da equação. Logo decidimos aplicar uma regressão polinomial, que analisa os vários graus das variáveis independentes, com interações de modo a analisar se há mais termos que influenciam a variável dependente.

```
Call:
lm(formula = time ~ poly(vertices, 2, raw = TRUE) * poly(arcs,
    1, raw = TRUE), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.15247 -0.01695 -0.00115  0.00632  0.38302

Coefficients:
                                                    Estimate Std. Error
(Intercept)                                        1.079e-02  3.612e-03
poly(vertices, 2, raw = TRUE)1                    -1.183e-04  1.962e-05
poly(vertices, 2, raw = TRUE)2                     2.561e-07  1.956e-08
poly(arcs, 1, raw = TRUE)                          6.540e-07  2.160e-07
poly(vertices, 2, raw = TRUE)1:poly(arcs, 1, raw = TRUE) -6.498e-10  5.079e-10
poly(vertices, 2, raw = TRUE)2:poly(arcs, 1, raw = TRUE)  1.872e-13  3.026e-13
                                                   t value Pr(>|t|)
(Intercept)                                          2.987  0.00284 **
poly(vertices, 2, raw = TRUE)1                      -6.033 1.76e-09 ***
poly(vertices, 2, raw = TRUE)2                      13.090  < 2e-16 ***
poly(arcs, 1, raw = TRUE)                            3.028  0.00247 **
poly(vertices, 2, raw = TRUE)1:poly(arcs, 1, raw = TRUE)  -1.279  0.20089
poly(vertices, 2, raw = TRUE)2:poly(arcs, 1, raw = TRUE)   0.619  0.53613
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04704 on 3984 degrees of freedom
Multiple R-squared:  0.6233,    Adjusted R-squared:  0.6228
F-statistic:  1318 on 5 and 3984 DF,  p-value: < 2.2e-16
```

(a) *Dinic*

```
Call:
lm(formula = time ~ poly(arcs, 2, raw = TRUE) * poly(vertices,
    1, raw = TRUE), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-2.8542 -0.1838  0.0049  0.2773  3.5906

Coefficients:
                                                    Estimate Std. Error
(Intercept)                                        3.844e-01  3.436e-02
poly(arcs, 2, raw = TRUE)1                        -2.786e-06  1.667e-06
poly(arcs, 2, raw = TRUE)2                         3.893e-10  1.171e-11
poly(vertices, 1, raw = TRUE)                     -2.063e-03  9.156e-05
poly(arcs, 2, raw = TRUE)1:poly(vertices, 1, raw = TRUE)  6.363e-08  1.844e-09
poly(arcs, 2, raw = TRUE)2:poly(vertices, 1, raw = TRUE) -4.668e-13  1.133e-14
                                                   t value Pr(>|t|)
(Intercept)                                         11.188   <2e-16 ***
poly(arcs, 2, raw = TRUE)1                          -1.671   0.0949 .
poly(arcs, 2, raw = TRUE)2                          33.256   <2e-16 ***
poly(vertices, 1, raw = TRUE)                      -22.530   <2e-16 ***
poly(arcs, 2, raw = TRUE)1:poly(vertices, 1, raw = TRUE)  34.508   <2e-16 ***
poly(arcs, 2, raw = TRUE)2:poly(vertices, 1, raw = TRUE) -41.217   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.688 on 3984 degrees of freedom
Multiple R-squared:  0.9573,    Adjusted R-squared:  0.9573
F-statistic: 1.787e+04 on 5 and 3984 DF,  p-value: < 2.2e-16
```

(b) *EK*

Figura 7: Resultados da regressão polinomial

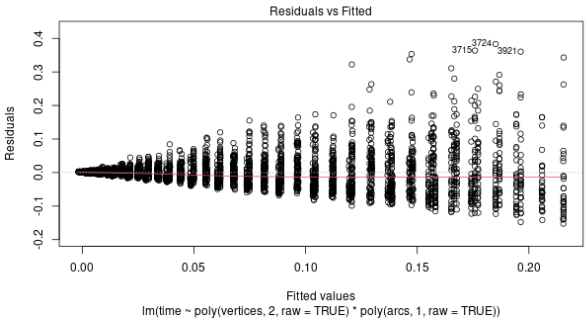
```
Call:
lm(formula = time ~ poly(vertices, 3, raw = TRUE) + poly(vertices,
    2, raw = TRUE):poly(arcs, 1, raw = TRUE) + poly(arcs, 1,
    raw = TRUE), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.29900 -0.01498 -0.00082  0.00936  0.40991

Coefficients:
                                                      Estimate Std. Error
(Intercept)                                          -3.025e-03  7.654e-03
poly(vertices, 3, raw = TRUE)1                        4.153e-05  6.943e-05
poly(vertices, 3, raw = TRUE)2                       -5.444e-08  1.552e-07
poly(vertices, 3, raw = TRUE)3                        7.018e-11  9.803e-11
poly(arcs, 1, raw = TRUE)                            -2.649e-07  3.409e-07
poly(vertices, 2, raw = TRUE)1:poly(arcs, 1, raw = TRUE)  4.022e-09  8.931e-10
poly(vertices, 2, raw = TRUE)2:poly(arcs, 1, raw = TRUE) -1.369e-12  5.763e-13
                                                     t value Pr(>|t|)
(Intercept)                                           -0.395   0.6927
poly(vertices, 3, raw = TRUE)1                         0.598   0.5498
poly(vertices, 3, raw = TRUE)2                        -0.351   0.7257
poly(vertices, 3, raw = TRUE)3                         0.716   0.4741
poly(arcs, 1, raw = TRUE)                             -0.777   0.4371
poly(vertices, 2, raw = TRUE)1:poly(arcs, 1, raw = TRUE)   4.504 6.87e-06 ***
poly(vertices, 2, raw = TRUE)2:poly(arcs, 1, raw = TRUE)  -2.375   0.0176 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.05586 on 3983 degrees of freedom
Multiple R-squared:  0.9248,    Adjusted R-squared:  0.9247
F-statistic:  8163 on 6 and 3983 DF,  p-value: < 2.2e-16
```

(c) *MPM*

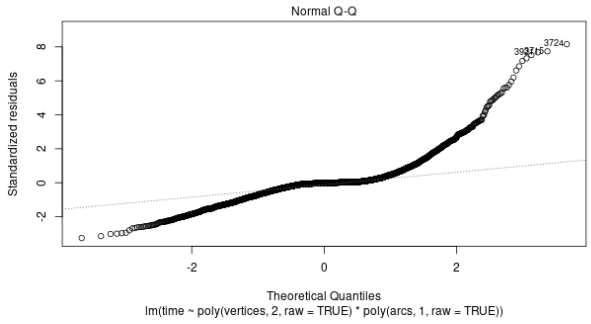
Figura 7: Resultados da regressão polinomial

Segundo os resultados das regressões lineares presentes na Figura 3, obtivemos um $\bar{R}^2$ de 0.6228, 0.9573 e 0.9247 para os algoritmos **Dinic**, **EK** e **MPM** respetivamente.

À semelhança dos resultados da Subsubseção 3.3.1, estes valores são bastante elevados para os algoritmos **EK** e **MPM**, mostrando que mais termos do que os qye foram considerados anteriormente são relevantes para o modelo de regressão polinomial. No entanto, para o algoritmo **Dinic**, apenas 62.28% da variação é explicada por este modelo. Ou seja, apesar de termos considerado mais termos na regressão polinomial, este modelo não explica uma grande percentagem da variação da variável de resposta.

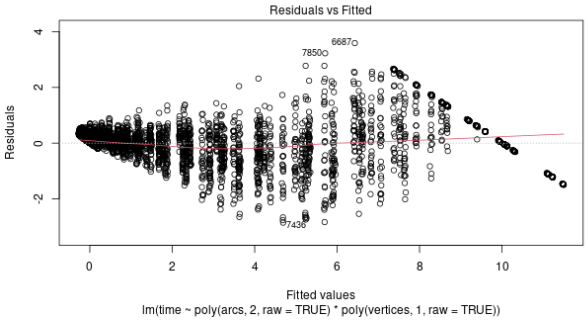


(d) *Residuals vs Fitted*

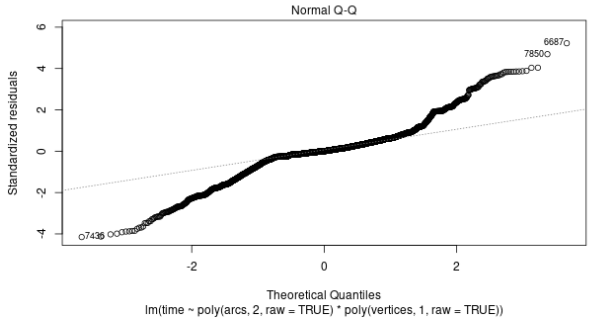


(e) *Normal QQ-Plot*

Figura 8: Pressupostos para o algoritmo Dinic

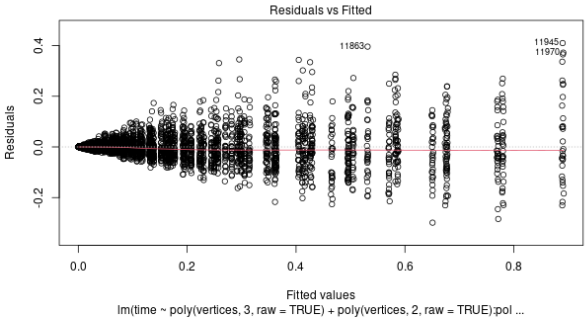


(a) *Residuals vs Fitted*

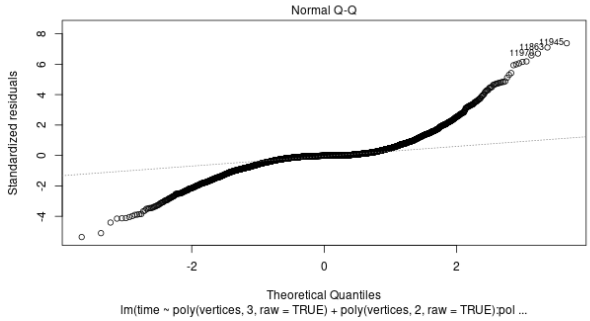


(b) *Normal QQ-Plot*

Figura 9: Pressupostos para o algoritmo EK



(a) *Residuals vs Fitted*



(b) *Normal QQ-Plot*

Figura 10: Pressupostos para o algoritmo MPM

Nas Figuras 8d, 9a e 10a, verificamos que a linha horizontal, que representa a média dos valores dos resíduos para cada valor estimado, está muito próxima de zero, ao contrário dos pressupostos da regressão linear multivariada da Subsubseção 3.3.1.

Os pressupostos da homocedasticidade e da normalidade dos resíduos, à semelhança dos resultados presentes na Subsubseção 3.3.1, não se verificam.

4 Conclusão

Em termos de *performance* o **EK** é o pior algoritmo, e o **Dinic** é o melhor. O **Dinic** é também o mais consistente. Apesar dos resultados ficarem perto da complexidade máxima teórica, esta não explica completamente a variação do tempo de execução. Especulamos que para redes mais densas ($p > 0.7$) o comportamento dos algoritmos se mantenha, continuando o **Dinic** a ser o mais eficiente.

Não obstante, do ponto de vista experimental, seria interessante realizar experiências que determinassem se a capacidade máxima influência o desempenho destes algoritmos.

5 Referências

[1] L. Paquete, "Measurements," 2022.

[2] R. Pi, "Raspberry pi documentation - processors," https://www.raspberrypi.com/documentation/computers/processors.html#bcm2711, 2022.

[3] D. J. Lilja, *Measuring computer performance: a practitioners guide*. Cambridge University Press, 2005.

[4] C. C. McGeoch, *A Guide to experimental algorithmics*. Cambridge University Press, 2012.

[5] L. Paquete, "Exploratory data analysis," 2022.

[6] ——, "Linear regression," 2022.