Trabalho Prático - Meta 1

Exploratory Data Analysis and Linear Regression

José Gonçalves - 2019223292 Luís Jordão - 2018283628 Tiago Carriço - 2018273809

Introdução

O objetivo principal deste trabalho é o desenho e realização de várias experiências de modo a comparar algoritmos que resolvem o problema do fluxo máximo possível de um ponto a outro num dado grafo conexo, verificando se na prática as complexidades teóricas formuladas são válidas.
Os algoritmos testados são *Edmond-Karp* [EK], *Dinic* e *Malhotra, Pramodh-Kumar e Maheshwari* [MPM], e os únicos critérios tomados em conta são se o algoritmo conseguiu descobrir a melhor solução no tempo permitido e a velocidade com que o fez.

Metodologia

De maneira a tentar obter resultados o mais precisos possíveis, irão manter-se todos parâmetros dos casos de teste constantes exceto aquele a ser testado de momento, podendo então variar e ser unicamente testados:

- Número de vértices;
- Probabilidade de um vértice ter um arco para um qualquer outro vértice;
- Capacidade máxima de fluxo de qualquer arco.

Variando apenas uma característica de cada vez, poderemos então analisar os dados resultantes e obter uma regressão linear dos dados e ter uma melhor ideia do peso de cada característica no tempo de execução de cada algoritmo.

Para apoiar a realização das experiências, além do código-fonte C++ para os três algoritmos a testar, foram fornecidos um *makefile* para os compilar e um programa em python capaz de gerar grafos de input. Além disso, antes da fase experimental, começou-se por desenvolver scripts em *bash* para automatizar a produção de ficheiros de input e a produção de tabelas dos tempos de execução dos algoritmos considerados para os inputs anteriormente gerados. Cada ficheiro é gerado com uma *seed* aleatória diferente, a qual fica registada no nome do ficheiro a par dos restantes parâmetros: número de vértices, probabilidade de gerar arco e capacidade máxima dos arcos. Para análise dos tempos medidos recorreu-se ao R para produzir *boxplots*, *scatter plots* e regressões lineares, usados para discussão dos resultados.

Começou-se por fazer uma experiência onde se variou a capacidade máxima dos ficheiros gerados de 660 a 660 desde 100 até 10000. Fixou-se o número de vértices a 500 e a probabilidade de gerar arcos a 100%. O objetivo aqui é observar se a capacidade máxima do arco influencia o tempo de execução dos algoritmos.

A segunda experiência focou-se em variar o número de vértices dos grafos de input para verificar as complexidades temporais assintóticas conhecidas dos algoritmos. O conjunto de ficheiros gerados

engloba grafos com número de nós desde 100 e 2000, com um passo de 200, com probabilidade de gerar arco de 50% e capacidade máxima dos arcos definida a 100.

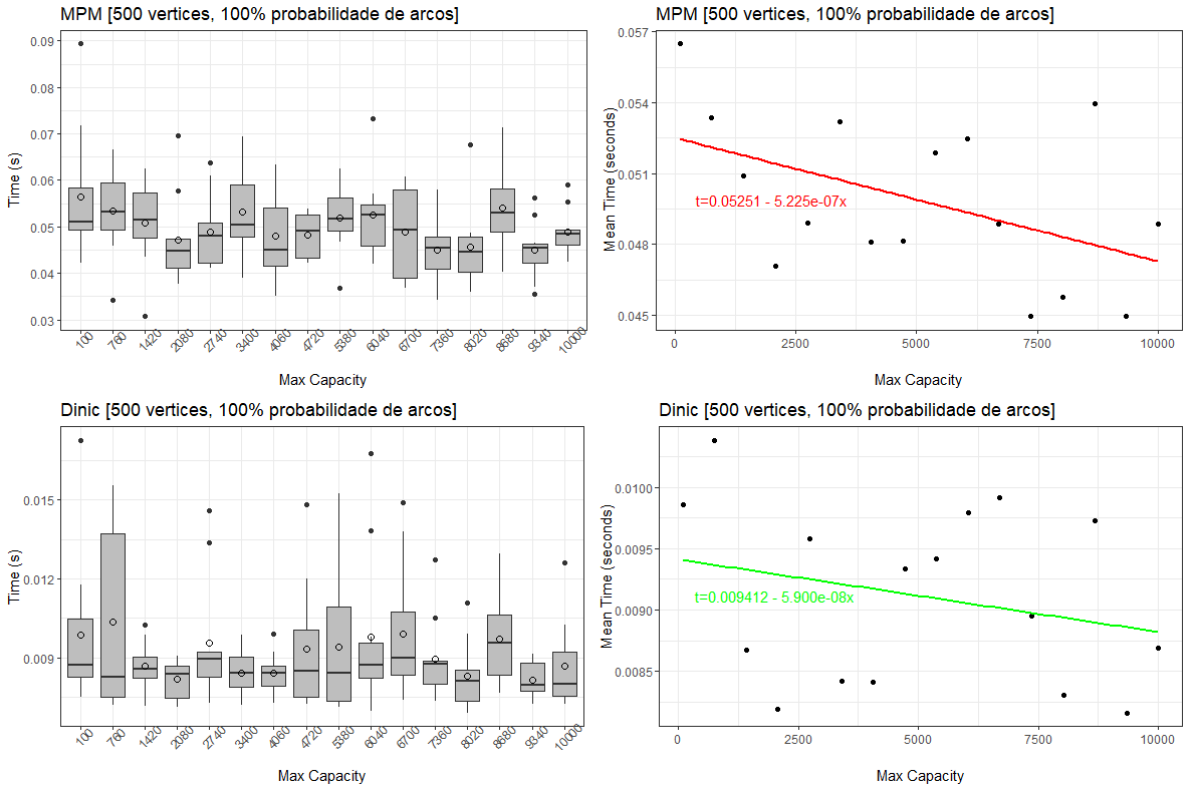
Já cedo neste conjunto de testes, por observação gráfica dos valores obtidos, chegou-se à conclusão de que ao fixar a probabilidade de gerar arco, o número de arcos dos grafos gerados não era igualmente fixo. Este erro no desenho desta experiência levou a uma reconsideração do significado de variar a possibilidade de gerar arco e as suas implicações: a impossibilidade de controlar diretamente o número de arcos dos grafos. Tal dificulta a interpretação de futuras experiências onde o foco seria o número de vértices, pois na verdade estariam a variar duas variáveis.

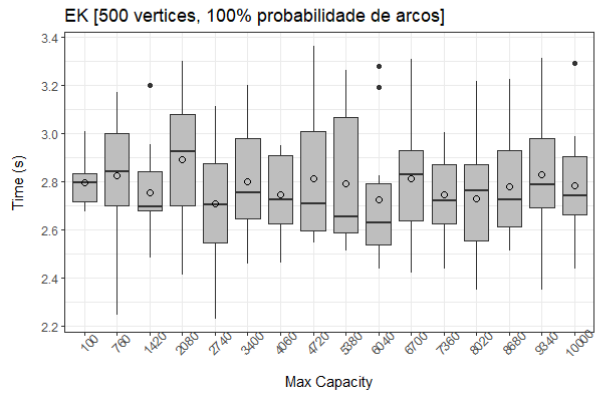
Assim sendo, para as próximas experiências, decidiu-se apenas variar a probabilidade de gerar arcos, continuando o objetivo a ser a avaliação da complexidade temporal dos algoritmos. O primeiro conjunto de grafos tinha 400 vértices e uma capacidade máxima de 100, enquanto que no segundo o número de vértices foi fixado a 1000 de modo a obter maiores tempos de execução para os algoritmos *Dinic* e *Malhotra, Pramodh-Kumar e Maheshwari* e assim diminuir erros de precisão decimal, obtendo-e no entanto conclusões semelhantes.

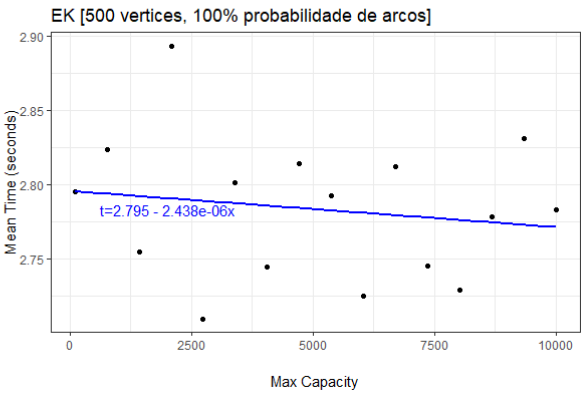
Resultados Experimentais e Discussão

Capacidade Máxima

Como a complexidade temporal de cada um dos algoritmos não depende da capacidade máxima de cada arco é de esperar que, ao se fixar o |V| e o |A|, os resultados são suficientemente semelhantes.



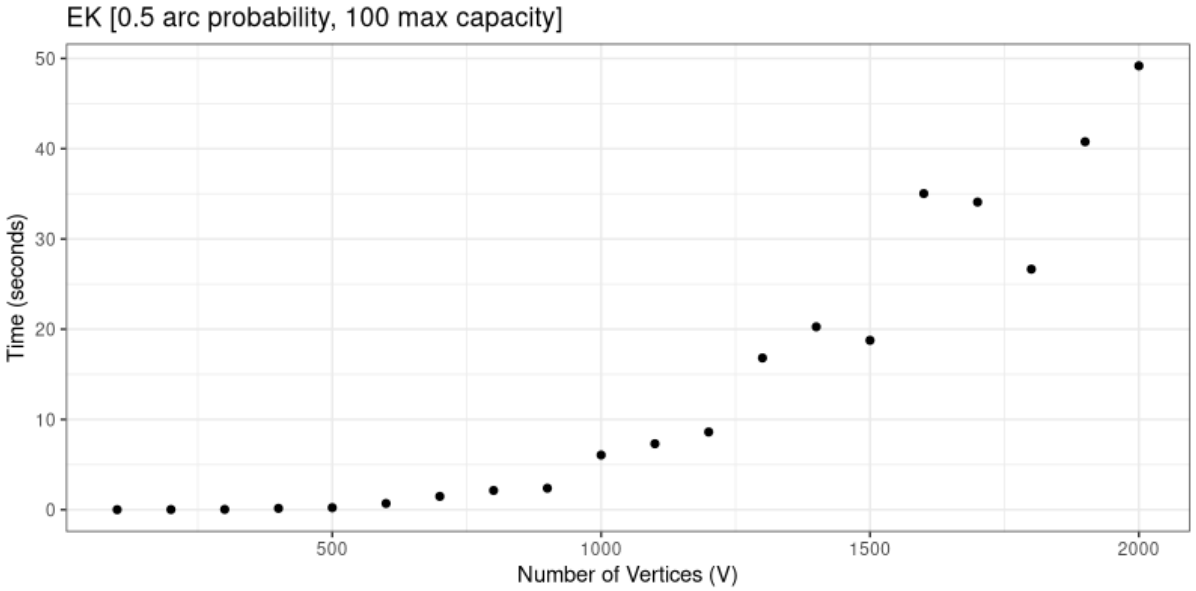




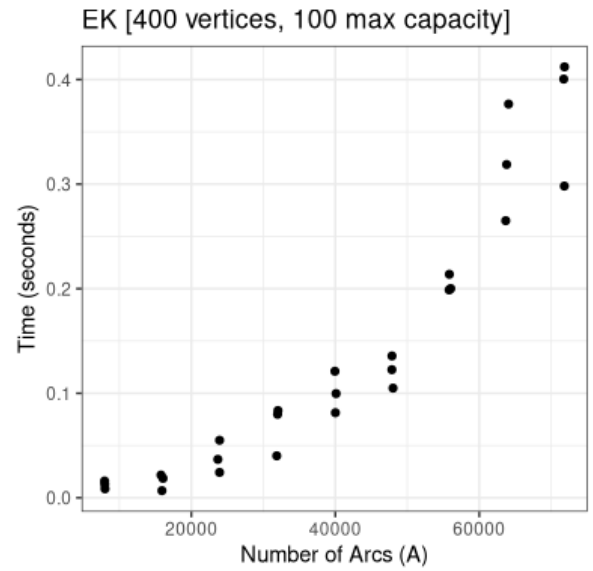
Com a análise dos gráficos é observável que não há aumento do tempo de execução de cada um dos algoritmos, o que pode levar a assumir que esta variável não tem impacto significante no tempo de execução dos programas, sendo as variações observadas atribuíveis ao sistema operativo.

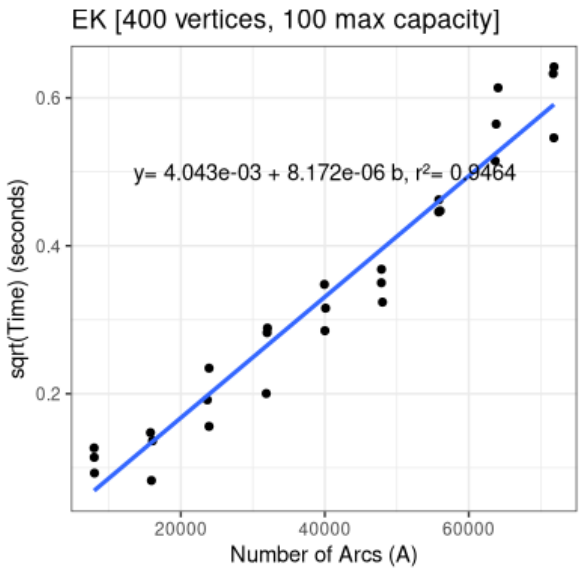
Edmond-Karp

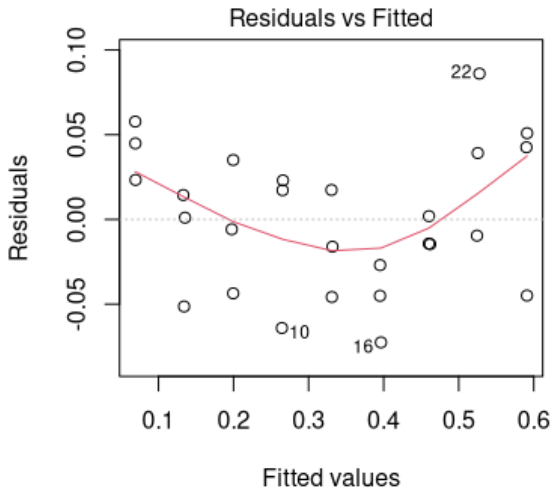
O algoritmo de *Edmon-Karp* tem uma complexidade temporal de $O(|V||A|^2)$. Ao correr a segunda experiência, onde se variava apenas os números de vértices V, era esperado observar um comportamento linear em relação a V. O scatter plot sugere, no entanto, um crescimento polinomial do tempo de execução. Foi neste caso que mais notório se notou o erro de desenho da segunda experiência: assumir que uma probabilidade de gerar arco fixa resultava num número de arcos fixo. Na verdade, com o aumento dos vértices, ocorre um aumento do número de arcos para uma mesma probabilidade, justificando assim o comportamento observado.

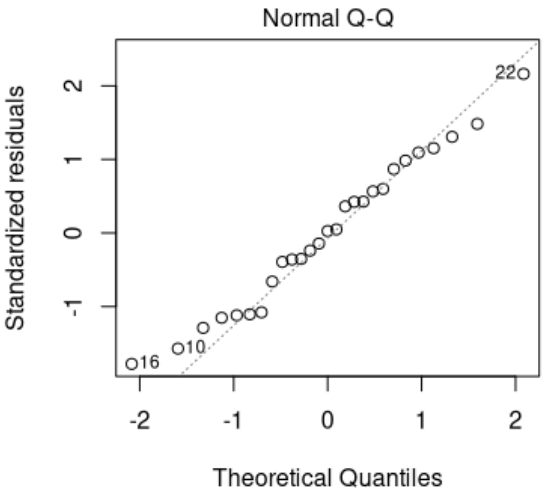


Nas últimas duas experiências, nas quais se variava apenas a probabilidade de gerar arcos e, consequentemente, o número de arcos do grafo A, para um número fixo de vértices, esperava-se observar uma relação quadrática entre A e o tempo de execução. De facto verificou-se que uma regressão linear com o eixo do tempo transformado, através da raiz quadrada, tendo em conta o crescimento quadrático do mesmo, é o que melhor se ajusta, além de se verificar que se pode assumir a sua linearidade através do *Residuals vs Fitted* e do *Normal-QQ plots*.







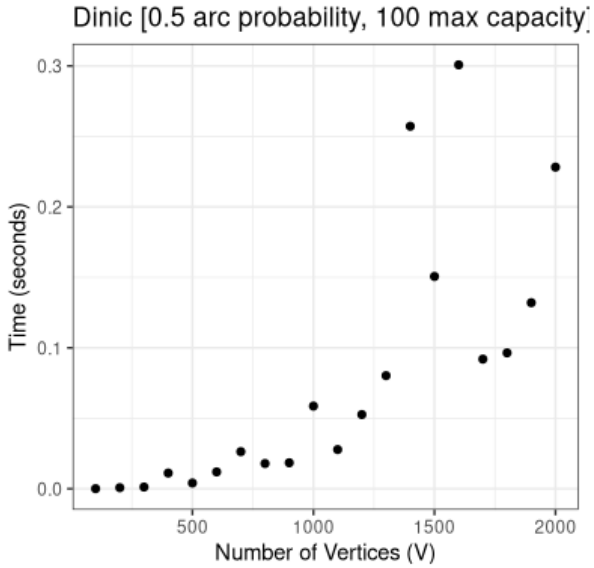


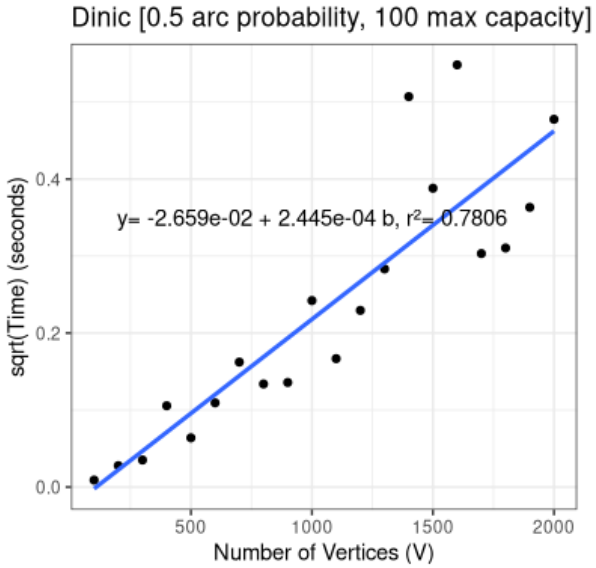
Dinic

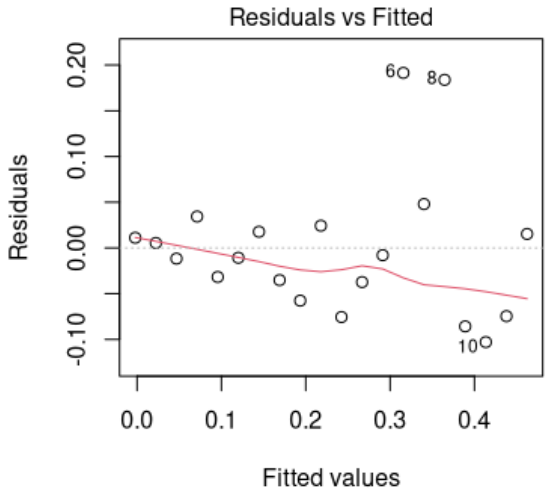
O algoritmo Dinic tem complexidade temporal $O(|A||V|^2)$, pelo que se espera que em testes demonstra uma curva quadrática quando observado face à variação do número de vértices, e um comportamento linear face ao número de arcos.

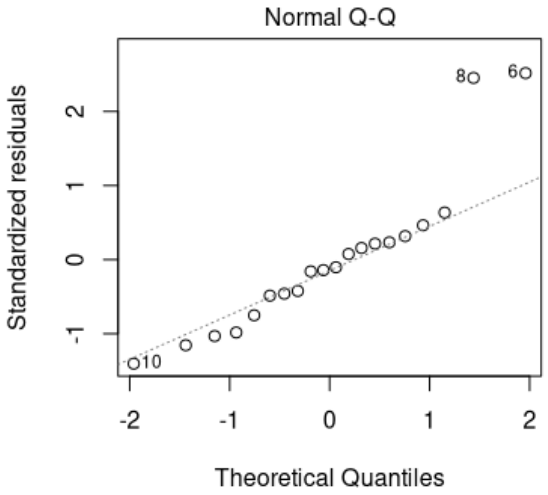
Na segunda experiência, onde se tenta isolar o número de vértices, observa-se que este apresenta o crescimento quadrático esperado. Porém ao aplicar regressão linear, o modelo encontrado tem um bom, mas não excelente ajuste, além que não é possível assumir totalmente linearidade face ao observado no *Residuals vs Fitted plot*, onde os pontos não estão uniformemente distribuídos ao longo da linha vermelha: encontram-se inicialmente próximos mas vão se gradualmente afastando.

Considerando o erro mencionado anteriormente nesta experiência, o qual leva a que o número de arcos também tenha variado, compreende-se estas observações, pois para um número alto de vértices, o número de arcos gerados será ainda maior, pelo que poderá ter conseguido exercer uma influência não negligenciável no tempo apesar do fator quadrático dos vértices.

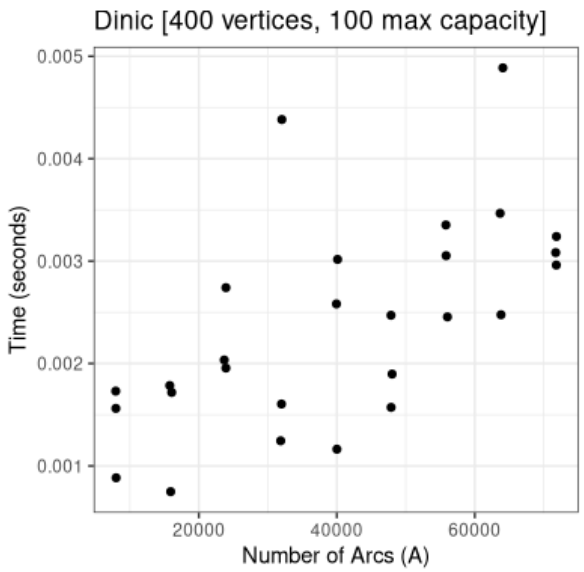


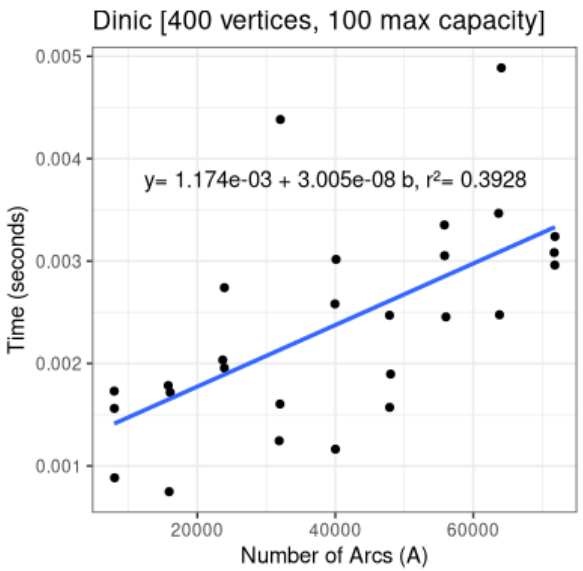


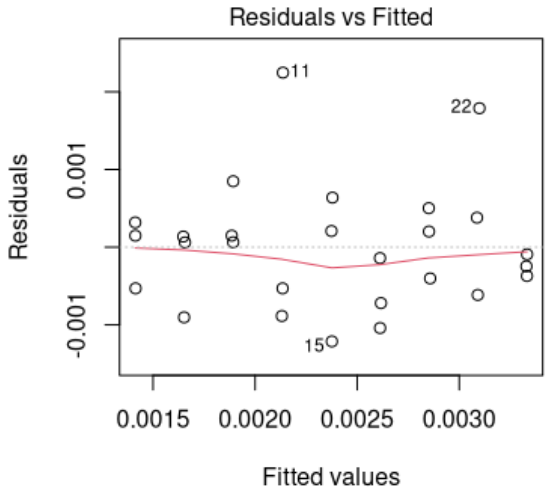


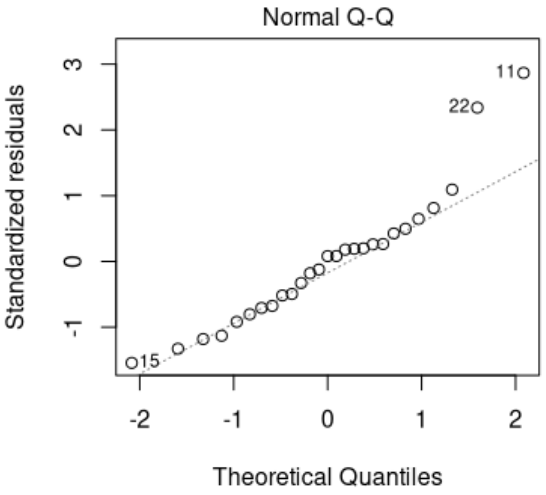


Avançando para a terceira experiência, onde apenas o número de arcos é variável, é possível estabelecer o comportamento linear esperado. Observando o *Residuals vs Fitted* e o *Normal-QQ plots* conclui-se é possível assumir a linearidade dos dados, já que a linha vermelha está praticamente sobreposta ao eixo horizontal, os pontos encontram-se uniformemente distribuídos ao longo desta e no *Normal-QQ plot* os pontos maioritariamente alinham-se com a guia de 45 graus. Porém é relevante destacar o baixo valor do coeficiente de determinação, R2. Uma hipótese para tal pode ser dada tendo em conta os valores de tempo de execução na ordem das milésimas de segundo. A esta escala, qualquer atraso resultante do sistema operativo na gestão de processos pode traduzir-se numa grande variação do tempo.







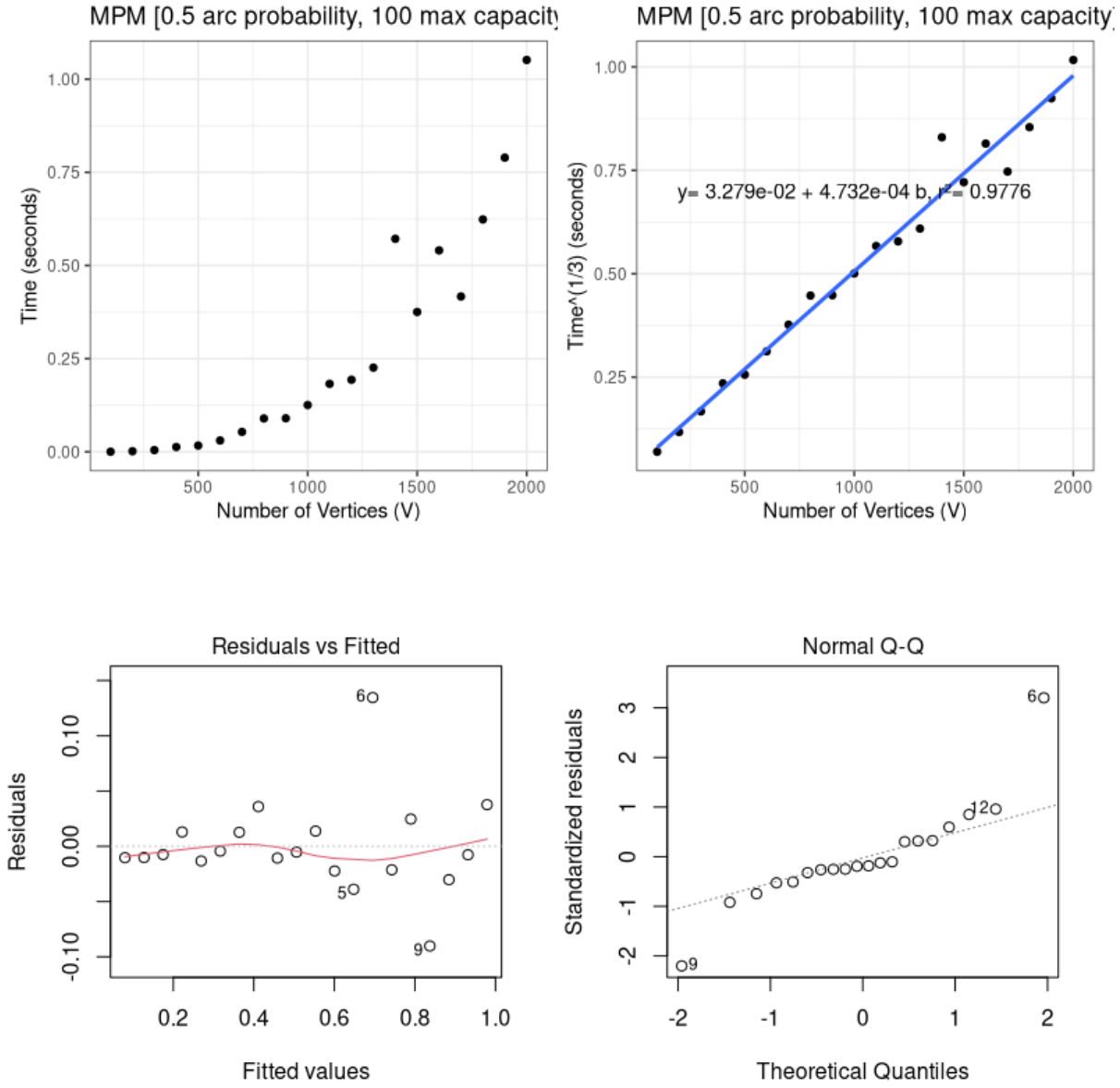


Malhotra, Pramodh-Kumar e Maheshwari

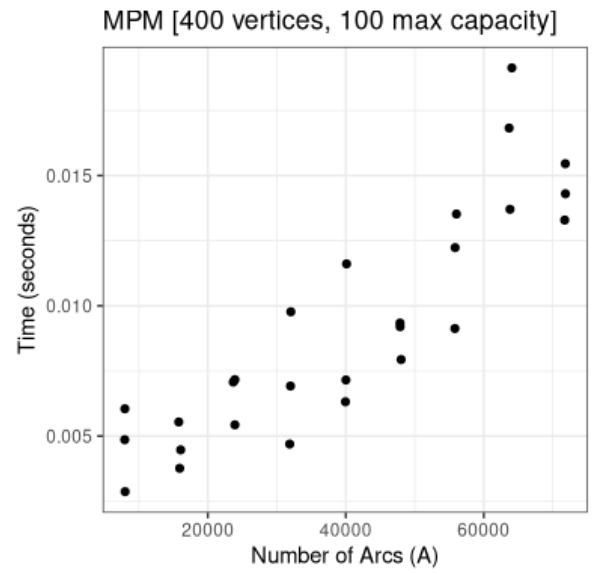
Tendo uma complexidade de $O(|V|^3)$, o *Malhotra, Pramodh-Kumar e Maheshwari* [MPM] será o algoritmo que apresenta uma pior escalabilidade quando testado em função do número de vértices do grafo, sendo que é esperado que se observe uma curva cúbica à medida que os testes avancem.

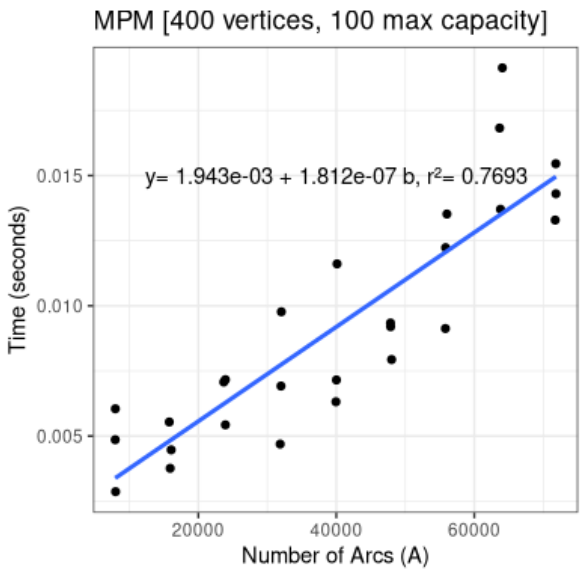
Os resultados obtidos na segunda experiência parecem condizer maioritariamente com as expectativas, o que faz sentido tendo em conta a complexidade unicamente dependente do número de vértices do grafo dado.

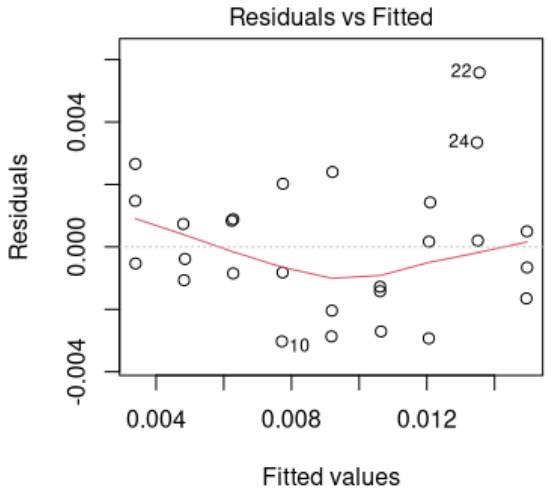
Observando os gráficos referentes aos *Residuals vs Fitted* e *QQ-Plots*, pode-se afirmar que é possível assumir um bom grau de linearidade dos dados, sendo que no primeiro a linha vermelha está muito próxima da linha base em toda a sua extensão, e no segundo os pontos presentes estão relativamente próximos da linha pontilhada.

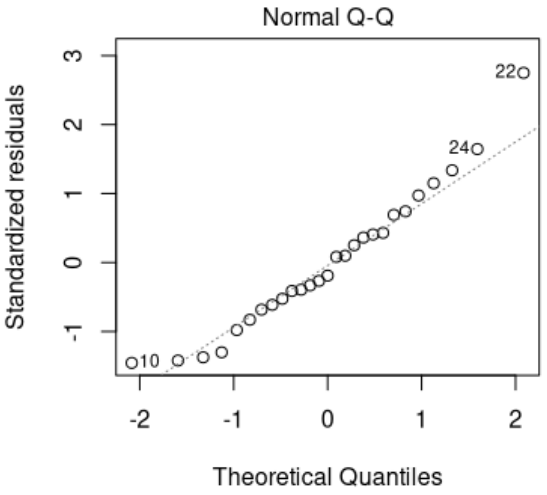


Na terceira experiência, a maioria das condições de linearidade também são constatadas, observando-se uma relação linear positiva entre o número de arcos e o tempo. No entanto, quanto ao gráfico que demonstra a regressão linear, há uma dispersão mais elevada dos pontos observados. Tal pode ser atribuído à escala reduzida dos valores de tempo, os quais se encontram na ordem das centésimas de segundo. Esta dispersão tanto pode ter origem em fatores externos (como outras operações do sistema) como em fatores não controláveis nos casos de teste, ou então é de facto necessário considerar o número de arcos ter influência no tempo do algoritmo. Justifica-se então que este fenómeno seja analisado em futuros estudos e testes.









Conclusão

Partindo de três algoritmos, *Edmond-Karp* [EK], *Dinic* e *Malhotra, Pramodh-Kumar e Maheshwari* [MPM], que oferecem resposta a um mesmo problema do fluxo máximo num grafo conexo, foi possível desenhar experiências que, através do uso de ferramentas como *boxplot*, *scatter plot* e regressão linear, motivaram uma análise exploratória do seu comportamento em relação ao tempo de execução dos algoritmos para diferentes instâncias do problema.

Essa análise contribui para a validação prática das complexidades temporais teóricas dos três algoritmos, abrindo também possibilidade para futuro trabalho. Este pode ser testar se de facto a variável da capacidade máxima dos arcos não influencia os tempos de execução significativamente e se os resultados observados no tempo de execução do MPM face ao número de arcos poderá ser atribuído a erros aleatórios resultantes da concorrência de processos e acessos à memória ou são causados pelo aumento do número de arcos.