Trabalho Prático 1

Turma 3DE

1180017 _ Gabriel Pelosi

1191063 _ Sofia Costa

Docente

José Matos

Unidade Curricular

ANADI

Abril, 2022

Índice

1. Contextualização do tema

1.1. Análise do funcionamento dos servidores VPN do DEI

O DEI mantém vários servidores de rede privada virtual (VPN). Estes serviços permitem aos utilizadores, localizados em redes exteriores ao DEI, criar uma ligação de rede virtual no seu posto de trabalho equivalente a uma ligação física às redes internas do DEI. Com esta ligação, os serviços das redes do DEI que não estão acessíveis publicamente passam a estar disponíveis.

Como acontece com todos os serviços de rede do DEI, é mantido um registo das sessões VPN estabelecidas pelos utilizadores. Com base neste registo de atividades pretendem-se vários tipos de análise ao funcionamento destes serviços e a obtenção de conclusões relevantes para a respetiva administração. O registo de atividades a ser objeto de análise é fornecido sob a forma de um ficheiro de texto que por motivos de privacidade foi anonimizado (os nomes dos utilizadores e os endereços de rede foram removidos). Cada sessão é apresentada sob a forma de uma linha de texto, em sequência temporal, contendo os seguintes elementos: servidor, protocolo de VPN, data de início da sessão, hora de início da sessão, hora do fim da sessão, e respetiva duração em minutos.

Os dois parágrafos anteriores, foram retirados do enunciado do Trabalho Prático.

1.2. Análise de Desempenho de Métodos Heurísticos na resolução do problema de Escalonamento

Um problema de escalonamento é geralmente caracterizado por um conjunto T = {T1, T2, . . . , Tn} de tarefas a executar, num conjunto de máquinas M = {M1, M2, . . . , Mm}. Nesta classe de problemas pretende-se encontrar uma solução que corresponda à melhor sequência de processamento das tarefas do conjunto T, numa ou mais máquinas do conjunto M, de modo que todas as tarefas sejam executadas, respeitando as limitações das máquinas e eventuais restrições adicionais impostas ao problema.

O problema de escalonamento [2] é conhecido por ser um problema de complexidade NP-hard ou seja, a partir de uma certa dimensão torna-se impraticável encontrar a solução ótima de forma eficiente (ou seja, resolver o problema de forma exata). Uma das métricas de desempenho (funções objetivo) mais usada para se calcular o valor da solução é o makespan, Cmax. Onde, Cmax = max{Ci}, em que Ci é o instante de tempo da conclusão da tarefa Ti.

Mais uma vez, os dois parágrafos anteriores foram retirados do enunciado do trabalho prático de modo a contextualizar o problema proposto.

2. Fases de Metodologia

Numa primeira fase, ambos os alunos procederam à análise completa do enunciado tirando os devidos apontamentos. De seguida, foi criado um repositório no BitBucket de modo a fazer o controlo de versões do código desenvolvido, por qualquer um dos membros do grupo, no RStudio. Relativamente à resolução de cada problema, não existiu uma divisão concreta de tarefas, ou seja, ambos os alunos reuniram fora de aula para resolver os exercícios propostos em conjunto promovendo, desta forma, a troca de ideias na formulação de uma solução.

No que tem em conta a escrita do relatório, cada aluno ficou responsável por documentar sobre dois exercícios alternadamente, isto é, um aluno escreveu sobre a discussão de resultados do exercício 1 do ponto 4.1 e o exercício 2 do ponto 4.2 enquanto o outro, focou-se na discussão dos resultados relativamente ao exercício 1 do ponto 4.2 e exercício 2 do ponto 4.1.

3. Exploração e Preparação dos Dados

3.1. Análise do funcionamento dos servidores VPN do DEI

O ficheiro com a designação "**vpnsessionsfile.txt**", abrange um período de utilização desde o final de 2016 até ao início de 2018. Neste ficheiro, existem cinco servidores: vsrv8, vsrv10, vsrv11, vsrv16, vsrv17; que suportam vários tipos de protocolo de VPN: PPTP, SSTP, SOFTETHER, OPENVPN L2, OPENVPN L3.

Para que o dado ficheiro seja utilizado em R, este foi importado para um *data frame* com o nome de "vpn_sessions" utilizando a função read.table() que recebe por parâmetro o caminho relativo para o ficheiro no repositório. Como se trata de um ficheiro .txt foi realizado um ligeiro tratamento dos dados como, por exemplo, a modificação do nome das colunas assim como a exclusão das colunas que não serão utilizadas.

	Servidor	Protocolo	Data	Início de Sessão	Fim de Sessão	Tempo de Sessão
1	vsrv11	SOFTETHER	2016-12-25	08:38	10:43	125

Figura 1 - Exemplo da primeira linha do data frame vpn_sessions

3.2. Análise de Desempenho de Métodos Heurísticos na resolução do problema de Escalonamento

O ficheiro com a designação "**mspandata.csv**" contém resultados relativos a 80 instâncias do problema de escalonamento com três colunas identificadas por "Mspan", "Metodo" e "problema". As instâncias encontram-se identificadas com números de 1 a 80, tendo sido apresentadas por ordem crescente de tamanho (por exemplo: o problema 1 é o problema com menor dimensão e o problema 80 é o problema de maior dimensão). Para cada problema conhece-se o valor do makespan ótimo identificado por OPT e o makespan obtido por cada MH identificadas por MH1, MH2 e MH3.

Semelhantemente ao ficheiro do ponto anterior, de modo a utilizarmos os dados do ficheiro "mspandata.csv" em R, foi criado um *data frame* com o nome de "makespan_data" com o auxílio da função *read.csv()* pois o ficheiro em questão possui uma extensão .csv.

Para avaliar a eficiência das MH é usual considerarmos o tempo de processamento na obtenção da solução (CPU time). Quanto menor o tempo de execução de um método mais eficiente será a técnica de

otimização. O ficheiro "**Cpu.run.time.csv**" contém o tempo de processamento (em segundos) das três MH (MH1, MH2 e MH3) na resolução das 50 instâncias com menor dimensão.

Mais uma vez, sendo o ficheiro "Cpu.run.time" um ficheiro .csv, este foi utilizado para criar um *data frame* com a função *read.csv()* ao qual se deu o nome de "cpu_runtime", assim foi possível utilizar a informação necessária proveniente do ficheiro para a resolução dos exercícios.

4. Análise e Discussão dos Resultados

4.1. Análise do funcionamento dos servidores VPN do DEI

Conceitos a entender:

Falha - embora a razão de uma sessão ter uma curta duração possa ter diversas origens, para o efeito deste estudo considera-se que qualquer sessão com duração igual ou inferior a um minuto representa uma falha.

Número de falhas simultâneas - Em cada minuto t0, define-se o número de falhas simultâneas como sendo o número de falhas que inicializaram ou que finalizaram no minuto t0.

Acesso - Um acesso é uma sessão com duração superior a um minuto.

Número de acessos simultâneos - Em cada minuto t0, define-se o número de acessos simultâneos como sendo o número de acessos que inicializaram antes do minuto t0 e terminaram depois do minuto t0.

4.1.1. Exercício 1

a) Quantos acessos teve cada servidor?

Para obter o número de acessos de cada servidor, utilizou-se a função sum() em conjunto com as condições necessárias para obter o número de acessos, ou seja, o nome do servidor corresponder a um presente na lista assim como o tempo de sessão ser superior a um minuto. Este processo foi repetido para cada servidor, uma vez que existem apenas cinco servidores esta proposta de resolução foi, a nosso ver, a mais eficiente.

Os seguintes resultados foram obtidos:

accesses_s10	1935L
accesses_s11	6080L
accesses_s16	16320L
accesses_s17	15075L
accesses_s8	15323L

Figura 2 - Número de acessos de cada servidor

Com base na imagem anterior, podemos verificar que o servidor que teve o maior número de acessos foi o servidor “vsrv16”, com 16320 acessos, enquanto o servidor “vsrv10” obteve o menor número, com 1935 acessos.

b) Quantas falhas teve cada servidor?

Semelhantemente ao exercício anterior, sendo a única diferença na resolução, a definição do tempo de sessão como igual ou inferior a um minuto, foi possível obter o número de falhas de cada servidor.

Obtendo os resultados seguintes:

fails_s10	276L
fails_s11	1493L
fails_s16	3329L
fails_s17	3154L
fails_s8	3398L

Figura 3 - Número de falhas de cada servidor

De acordo com a imagem anterior, podemos observar que o servidor com o menor número de falhas foi o servidor “vsvr10” com apenas 276 falhas, pelo contrário, o servidor “vsvr8” obteve o maior número de falhas com um total de 3398 falhas.

c) Quantas vezes o servidor ”x” usa o protocolo ”y”?

Fazendo uma soma, função sum(), do número de linhas em que o servidor “x” aparece juntamente com o protocolo “y”, no *data frame* vpn_sessions, definido pelo utilizador, podemos observar os seguintes resultados:

	protocol_name	vsrv8_uses	vsrv10_uses	vsrv11_uses	vsrv16_uses	vsrv17_uses
1	PPTP	8166	983	2698	8653	8036
2	SSTP	9349	1136	1088	9815	9210
3	SOFTETHER	158	16	1295	155	168
4	OPENVPN L2	980	76	1389	967	766
5	OPENVPN L3	68	0	1103	59	49

Figura 4 - Número de vezes o servidor "x" usa o protocolo "y"

Pela observação da imagem anterior, podemos verificar que o servidor vsrv16 utiliza, mais vezes do que qualquer outro servidor, os protocolos PPTP e SSTP. Os restantes protocolos, SOFTETHER, OPENVEP L2 e OPENVPN L3 são usados maioritariamente pelo servidor vsrv11. O servidor vsrv10 nunca utiliza o protocolo OPENVPN L3 e podemos concluir também que este é o protocolo menos utilizado enquanto o protocolo SSTP é o mais utilizado.

d) Determine as médias, medianas e desvios padrão mensais de acessos de cada servidor (apenas meses completos)

Para realizar este exercício, foi colocado num vetor, para cada servidor, o número total de acessos de cada mês completo, neste caso apenas os meses de 2017. Deste modo, foi possível utilizar as funções de média, mediana e desvio padrão para cada servidor de modo a obter o número mensal.

Os resultados obtidos foram os seguintes:

```
> # ==== MÉDIA MENSAL DE ACESSOS DE CADA SERVIDOR
> sprintf("O servidor 'vsrv8' tem uma média mensal de acessos aproximadamente igual a %g.", mean(accesses_2017_s8))
[1] "O servidor 'vsrv8' tem uma média mensal de acessos aproximadamente igual a 1152."
> sprintf("O servidor 'vsrv10' tem uma média mensal de acessos aproximadamente igual a %g.", mean(accesses_2017_s10))
[1] "O servidor 'vsrv10' tem uma média mensal de acessos aproximadamente igual a 52.8333."
> sprintf("O servidor 'vsrv11' tem uma média mensal de acessos aproximadamente igual a %g.", mean(accesses_2017_s11))
[1] "O servidor 'vsrv11' tem uma média mensal de acessos aproximadamente igual a 439.167."
> sprintf("O servidor 'vsrv16' tem uma média mensal de acessos aproximadamente igual a %g.", mean(accesses_2017_s16))
[1] "O servidor 'vsrv16' tem uma média mensal de acessos aproximadamente igual a 1223.83."
> sprintf("O servidor 'vsrv17' tem uma média mensal de acessos aproximadamente igual a %g.", mean(accesses_2017_s17))
[1] "O servidor 'vsrv17' tem uma média mensal de acessos aproximadamente igual a 1131.92."
```

Figura 5 - Média mensal de acessos de cada servidor

Com os resultados das médias mensais, para cada servidor, podemos concluir que o servidor com maior número de acessos por mês, é o servidor “vsrv16” com uma média de aproximadamente 1224 acessos por mês. O servidor com o menor número de acessos mensal é o servidor “vsrv10”. Apesar de apenas se tratar de meses completos, o servidor “vsrv10” é o servidor que possui o número mais baixo de acessos no total, o que seria de esperar que tivesses também o número mais baixo de acessos por mês. Tal é também comprovado pelo servidor “vsrv16” que possui o maior número de acesso no total pelo que o resultado obtido era de esperar.

```
> # ==== MEDIANA MENSAL DE ACESSOS DE CADA SERVIDOR
> sprintf("O servidor 'vsrv8' tem uma mediana mensal de acessos aproximadamente igual a %g.", median(accesses_2017_s8))
[1] "O servidor 'vsrv8' tem uma mediana mensal de acessos aproximadamente igual a 619.5."
> sprintf("O servidor 'vsrv10' tem uma mediana mensal de acessos aproximadamente igual a %g.", median(accesses_2017_s10))
[1] "O servidor 'vsrv10' tem uma mediana mensal de acessos aproximadamente igual a 0."
> sprintf("O servidor 'vsrv11' tem uma mediana mensal de acessos aproximadamente igual a %g.", median(accesses_2017_s11))
[1] "O servidor 'vsrv11' tem uma mediana mensal de acessos aproximadamente igual a 373.5."
> sprintf("O servidor 'vsrv16' tem uma mediana mensal de acessos aproximadamente igual a %g.", median(accesses_2017_s16))
[1] "O servidor 'vsrv16' tem uma mediana mensal de acessos aproximadamente igual a 697.5."
> sprintf("O servidor 'vsrv17' tem uma mediana mensal de acessos aproximadamente igual a %g.", median(accesses_2017_s17))
[1] "O servidor 'vsrv17' tem uma mediana mensal de acessos aproximadamente igual a 577."
```

Figura 6 - Mediana mensal de acessos de cada servidor

Com os resultados das medianas mensais, para cada servidor, podemos concluir que o servidor com a maior mediana mensal é o servidor “vsrv16” e o servidor com a menor mediana mensal é o servidor

"vsrv10". Mais uma vez, os resultados obtidos são os esperados tendo em conta os resultados obtidos nas alíneas anteriores.

```
> # ==== DESVIO PADRAO MENSAL DE ACESSOS DE CADA SERVIDOR
> sprintf("O servidor 'vsrv8' tem um desvio padrao mensal de acessos aproximadamente igual a %g.", sd(accesses_2017_s8))
[1] "O servidor 'vsrv8' tem um desvio padrao mensal de acessos aproximadamente igual a 1158.78."
> sprintf("O servidor 'vsrv10' tem um desvio padrao mensal de acessos aproximadamente igual a %g.", sd(accesses_2017_s10))
[1] "O servidor 'vsrv10' tem um desvio padrao mensal de acessos aproximadamente igual a 183.02."
> sprintf("O servidor 'vsrv11' tem um desvio padrao mensal de acessos aproximadamente igual a %g.", sd(accesses_2017_s11))
[1] "O servidor 'vsrv11' tem um desvio padrao mensal de acessos aproximadamente igual a 191.302."
> sprintf("O servidor 'vsrv16' tem um desvio padrao mensal de acessos aproximadamente igual a %g.", sd(accesses_2017_s16))
[1] "O servidor 'vsrv16' tem um desvio padrao mensal de acessos aproximadamente igual a 1216.51."
> sprintf("O servidor 'vsrv17' tem um desvio padrao mensal de acessos aproximadamente igual a %g.", sd(accesses_2017_s17))
[1] "O servidor 'vsrv17' tem um desvio padrao mensal de acessos aproximadamente igual a 1138.83."
```

Figura 7 - Desvio padrão mensal de cada servidor

Com os resultados dos desvios padrão mensais, para cada servidor, podemos concluir que o servidor com o maior desvio padrão mensal, é o servidor "vsrv16" o que este servidor os pontos dos dados relativos aos número de acessos mensais do servidor "vsrv16" estão espalhados por uma ampla gama de valores. Por outro lado, o servidor com um desvio padrão mensal menor, é o servidor "vsrv10", ou seja, os pontos dos dados tendem a estar mais próximos da média.

Assim, para cada servidor e para cada média, mediana e desvio padrão mensal, apenas dois servidores se destacaram no que tem em conta o maior e menor número, o que por um lado seria de esperar tendo em conta o número total de acessos de cada servidor, que foram calculados anteriormente.

4.1.2. Exercício 2

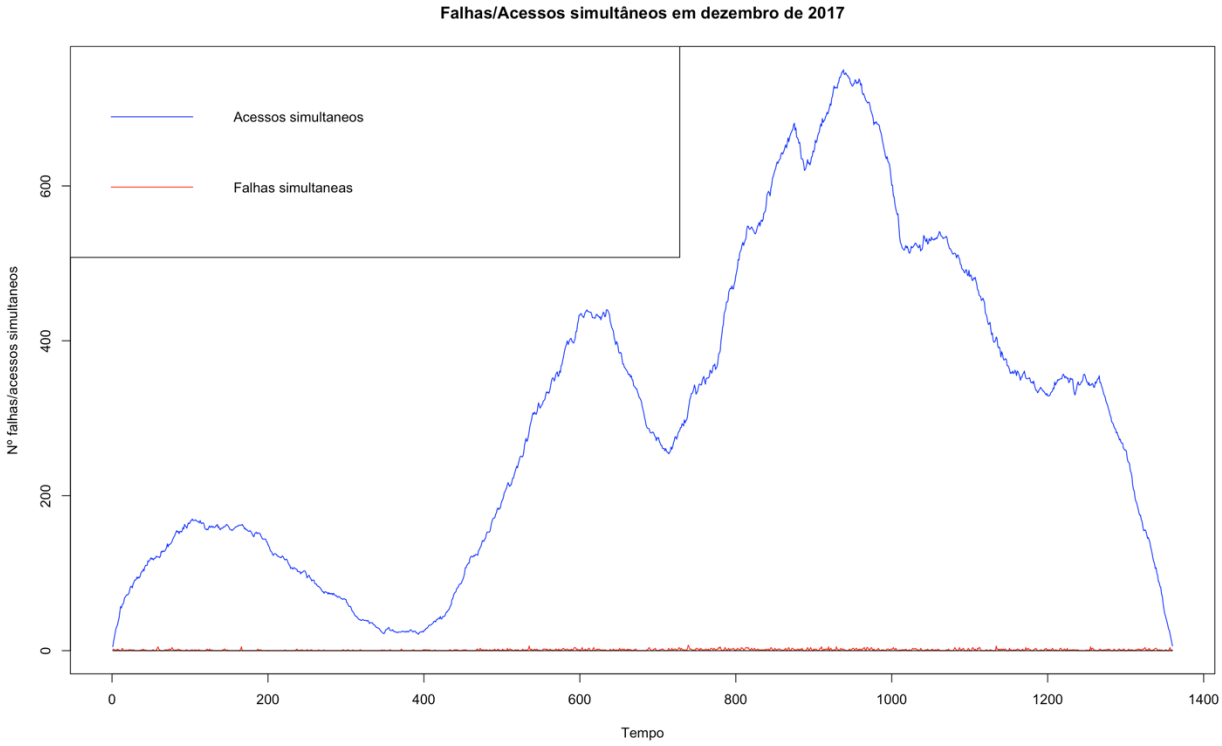
De modo a trabalhar apenas com os dados relativos ao mês de dezembro de 2017, utilizamos a função *filter()* e através do data frame do exercício anterior "vpn_sessions" criamos um novo data frame selecionando apenas as datas compreendidas entre o dia 1 de dezembro e 31 de dezembro. A este novo data frame, demos o nome de "dez2017".

a) Efetue um gráfico que nas abcissas represente o tempo, e nas ordenadas o número de falhas simultâneas e o número de acessos simultâneos.

Para resolver este exercício, foi necessário obter todos os minutos presentes no data frame anteriormente criado e referido (dez2017). Para isto, extraímos para um vetor e ordenamos por ordem crescente todos

os tempos de início de sessão e fim de sessão. Com este vetor, percorremos num ciclo *for* todos os minutos existentes. Dentro deste ciclo *for*, obtemos para cada minuto o número de acessos e falhas simultâneas colocando num novo *data frame* essa informação com o auxílio da função rbind().

De seguida, criamos um Time Series Plot, com a função ts.plot(), que vai criar um gráfico com a informação de todos os acessos e falhas simultâneas do mês de dezembro de 2017.



Com a observação do gráfico, podemos concluir que o número de acessos simultâneos é bastante superior em comparação com o número de falhas simultâneas, o que seria de esperar uma vez que existem mais acessos do que falhas como foi verificado nos exercícios anteriores. O número máximo de falhas simultâneas, não ultrapassaria o número 5, enquanto o número de acessos simultâneos vai além dos 600, o que demonstra uma diferença muito grande.

b) Efetue um diagrama de caixa de bigodes do número diário de falhas simultâneas, para cada servidor (o número diário de falhas simultâneas é o número total de falhas simultâneas que ocorreram nesse dia). Indique quantos outliers existem por servidor.

A solução proposta para este exercício, foi primeiro filtrar para cada servidor, um novo *data frame* que apenas contém as falhas relativas a esse servidor. De seguida, para obter o número diário de falhas simultâneas de cada servidor, agrupamos os dados dos *data frames* anteriores por servidor e data. Deste modo, foi possível obter um outro *data frame* com a as datas ordenadas do mês de dezembro e uma outra coluna que representa o número de falhas em cada dia.

Por fim construímos um boxplot com a informação obtida em cada *data frame* para cada servidor.

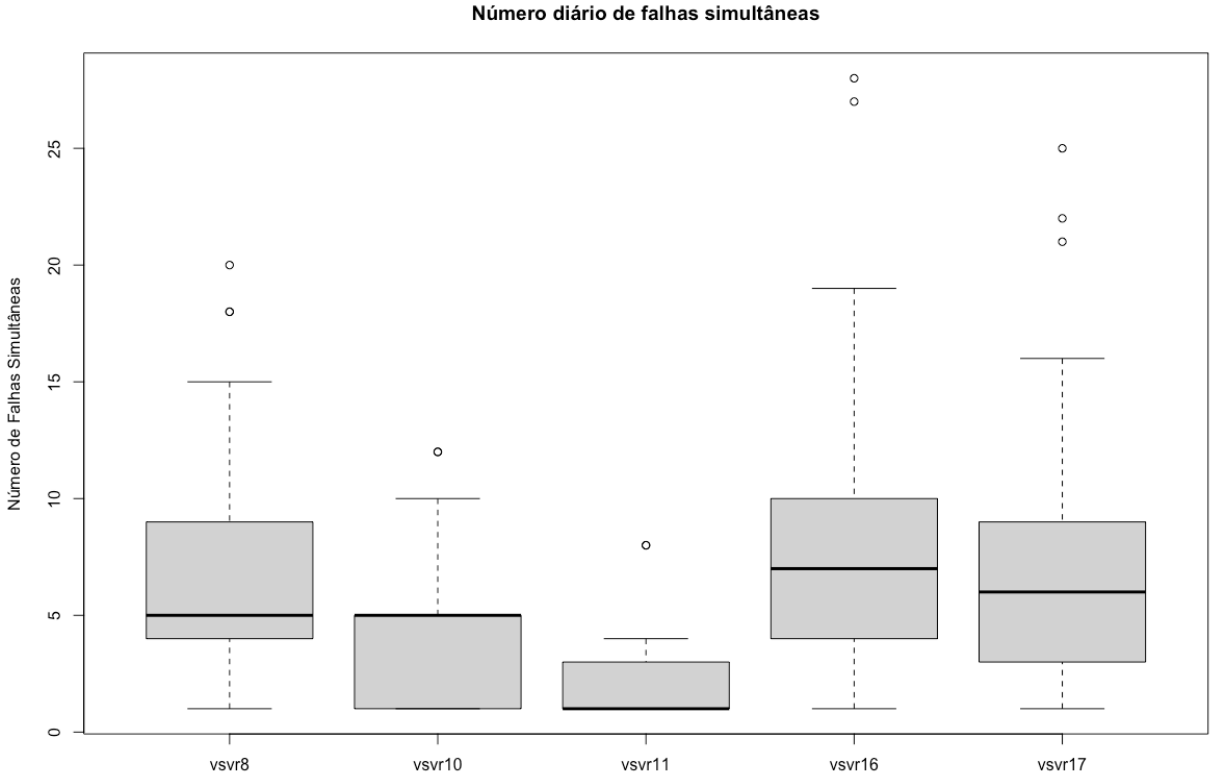


Figura 8 - Número diário de falhas simultâneas de cada servidor

Com a observação do gráfico, podemos concluir que o servidor que possui mais falhas diárias, é servidor vsrv16 e o que possui um menor número de falhas é o servidor vsrv11. Existem ainda alguns outliers significativos, principalmente no servidor vsrv16 pois estes encontram-se mais afastados da caixa de bigodes. Para cada servidor, existem os seguintes outliers, com a observação do gráfico:

- Vsrv8: 2 outliers
- Vsrv10: 1 outlier
- Vsrv11: 1 outlier

- Vsrv16: 2 outliers
- Vsrv17: 3 outliers

c) Verifique se há correlação entre o número de falhas simultâneas e o número de acessos simultâneos, no dia 11 de dezembro de 2017 das 12:00 às 14:00.

De modo a obter um *data frame* com o número de acessos e falhas simultâneas do dia 11 de dezembro das 12:00 as 14:00, o mesmo processo da alínea a) foi realizado, apenas com condições diferentes que vão de acordo com o que é pedido nesta alínea, ou seja, apenas dentro do intervalo de tempo requirido.

Para verificar se há correlação entre os dados obtidos, com a ajuda do data frame criado, df_falhas_acessos_dez11, começamos por utilizar o teste de Pearson. O teste de Pearson possui os seus prossupostos, ou seja, caso um dos prossupostos falhe, não é possível utilizar este teste para verificar a correlação entre os dados. Sendo o primeiro prossuposto descrito por "as variáveis serem contínuas e não devem existir *outliers* significativos", criamos um *boxplot* para testar este facto.

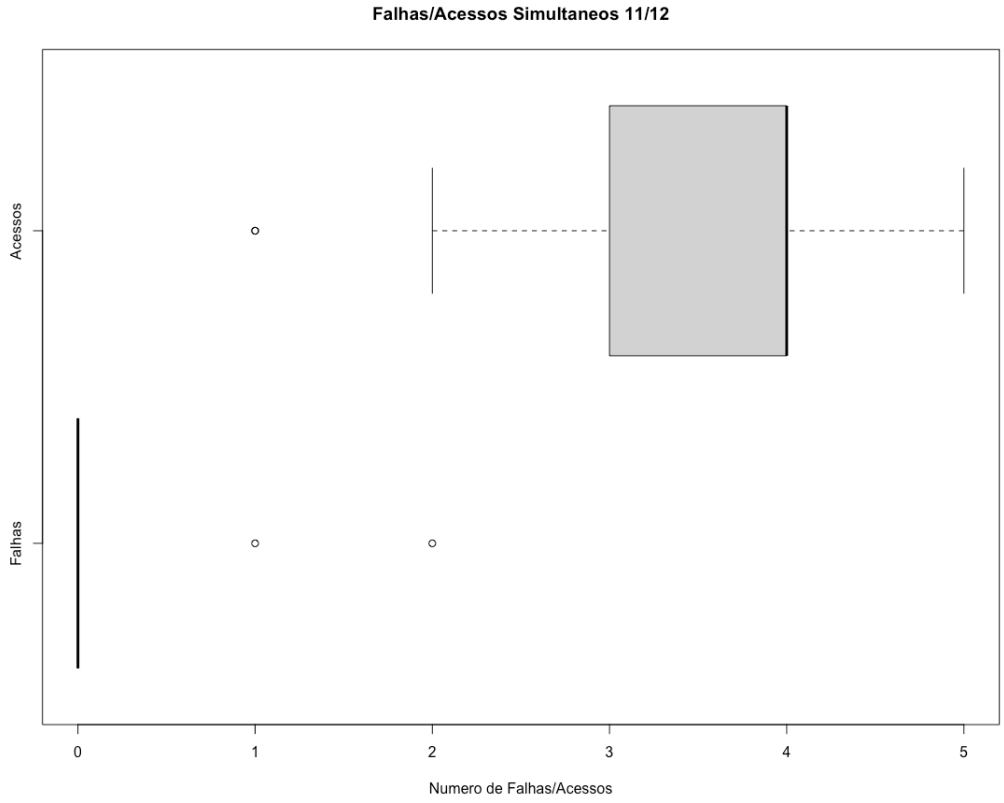


Figura 9 - Falhas/Acessos Simultâneos no dia 11 de dezembro

Como se pode observar na imagem anterior, existem outliers significativos pois estes querem dizer que existem resultados que diferem muito dos restantes. Assim, descartamos o uso do teste de Pearson para testar a correlação entre os dados.

De seguida, iremos usar o teste de Spearman, cor.test(), que é utilizado quando as variáveis são ambas ordinais ou quando uma das variáveis é continua e a outra ordinal. Para o teste de Spearman, obtemos os seguintes resultados:

```
        Spearman's rank correlation rho

data:  df_falhas_acessos_dez11$Num_Acessos_Simultaneos and df_falhas_acessos_dez11$Num_Falhas_Simultaneas
S = 1676.6, p-value = 0.4336
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1716352
```

Figura 10 - Teste de Correlação Ordinal de Spearman (alínea c)

Podemos assim concluir que, como rho (ρ) está mais próximo de 0, as variáveis de acessos simultâneos e falhas simultâneas do dia 11 de dezembro das 12h as 14h, estão fracamente correlacionadas. Existe uma fraca associação positiva entre as classificações devido ao número positivo de rho, contudo, esta associação não é significativa devido ao valor de p-value ser superior a 0.05.

4.2. Análise de Desempenho de Métodos Heurísticos na resolução do problema de Escalonamento

4.2.1. Exercício 1

a) Efetue um gráfico tal que: o eixo das abcissas contém os problemas e o eixo das ordenadas contém o valor do makespan das três MH's para cada instância (use cores para distinguir as MH).

Para resolver este exercício, foi utilizada a função *ggplot()* que no parâmetro "data" recebe o *data frame* com a informação do ficheiro, no eixo das abcissas contém os problemas e no eixo das ordenadas contém o valor do makespan de cada MH. Foi obtido o seguinte gráfico:

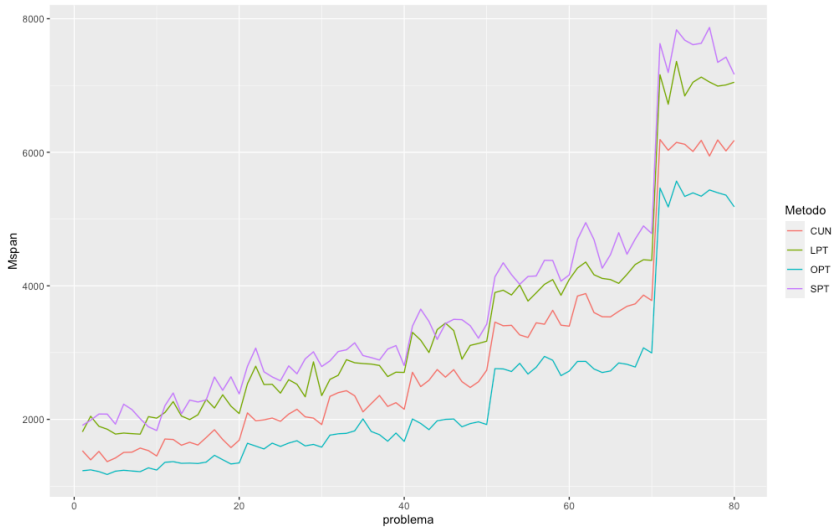


Figura 11 - Gráfico representativo do makespan obtido por cada MH

Pela observação do gráfico, podemos concluir que o método que mais se aproxima do OPT, é o método CUN pelo que este é considerado a técnica mais eficaz uma vez que o seu makespan é o menor. Contrariamente, os métodos SPT e LPT são considerados os menos eficazes pois apresentam valores de makespan muito mais superiores.

b) Considere apenas os dados relativos às 10 instâncias com menor tamanho. Haverá diferenças significativas entre o desempenho das três técnicas?

Com vista a considerar apenas os dados relativos às 10 instâncias com menor tamanho, ou seja, as instâncias numeradas de 1 a 10, foi criado um *data frame* que apenas inclui os dados relativos aos problemas de 1 a 10 para cada método. Vale ressaltar que, o método OPT foi excluído pois não caberia testa-lo nessa situação. De seguida, foi construído um gráfico da mesma forma que o exercício anterior mas utilizando o novo data frame com o nome de “smallest_10_data”:

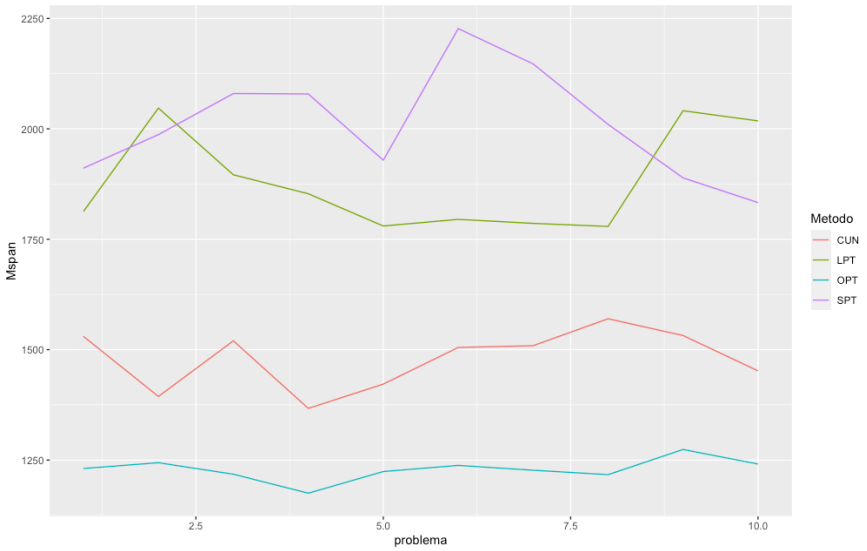


Figura 12 - Gráfico com as 10 instâncias de menor dimensão

De modo a auxiliar a identificação de diferenças significativas, foi realizado um teste de Friedman para analisar mais de duas amostras emparelhadas, tal como são aquelas que desejamos analisar. Formularam-se as seguintes hipóteses:

H0: O desempenho é igual entre as três técnicas.

H1: O desempenho é diferente entre as três técnicas.

Através da função friedman.test() e data frames com os valores de cada técnica relativamente às 10 instâncias de menor tamanho, o valor de p-value obtido foi 0.0003707. Este valor é muito baixo em comparação com 0.05 pelo que se rejeita H0, ou seja, o desempenho das três técnicas apresenta diferença significativa mesmo com apenas as primeiras 10 amostras.

```
> friedman.test(y = smallest_10_data$Mspan, groups = smallest_10_data$Metodo, blocks = smallest_10_data$problema)

        Friedman rank sum test

data:  smallest_10_data$Mspan, smallest_10_data$Metodo and smallest_10_data$problema
Friedman chi-squared = 15.8, df = 2, p-value = 0.0003707
```

Figura 13 - Teste de Friedman para as primeiras 10 amostras

c) No caso de a resposta da alínea anterior ser positiva, identifique qual a técnica mais eficaz.

Após determinar que há diferença significativa no desempenho das 3 técnicas, para identificar a técnica mais eficiente, será necessário realizar um test post-hoc, que irá comparar as técnicas duas em duas.

Como foi selecionado o teste de friedman para determinar se havia diferença significativa no desempenho, para o teste post-hoc, será utilizado o test Namenyi, que irá realizar comparações de dois em dois, para apresentar uma conclusão sobre a diferença do desempenho dos métodos. O resultado obtido foi o seguinte.

```
    CUN     LPT
LPT 0.01021 -
SPT 0.00042 0.64383
```

Figura 14 - Teste de Post-Hoc para as primeiras 10 amostras

De acordo com o resultado do teste Namenyi, executado através da função PMCMRplus::frdAllPairsConoverTest(), pode-se concluir que o melhor desempenho é da técnica CUN, pois a comparação dos pares apresenta p-values altos quando feitos entre o SPT e o LPT, porém, quando feitos entre o CUN, o p-value é baixo e chega até ser menor que o nível de significâncias 0.05.

A técnica mais eficaz, é a técnica CUN pois esta apresenta um makespan menor e encontra-se mais próxima do valor do makespan ótimo (OPT).

d) Responda às duas perguntas anteriores, mas usando os dados das 20 instâncias com maior dimensão.

Semelhantemente à resolução dos exercícios anteriores, com vista a considerar apenas os dados relativos às 20 instâncias com maior tamanho, ou seja, as últimas 20 instâncias, foi criado um *data frame* que apenas inclui os dados relativos aos problemas de 61 a 80 para cada método. De seguida, foi construído um gráfico da mesma forma que o exercício anterior, mas utilizando o novo data frame com o nome de "biggest_20_data":

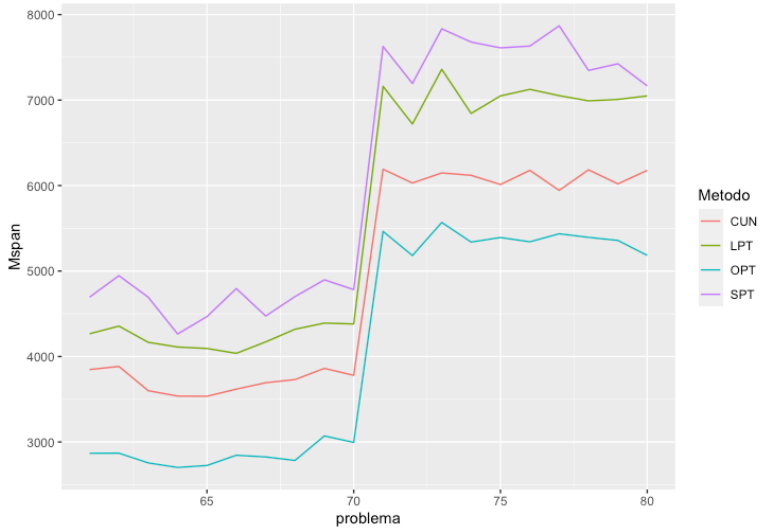


Figura 15 - Gráfico com as 20 instâncias de maior dimensão

Para auxiliar na identificação de diferenças significativas, foi realizado um teste de Friedman para analisar amostras. Formularam-se as seguintes hipóteses:

H0: O desempenho é igual entre as três técnicas.

H1: O desempenho é diferente entre as três técnicas.

Através da função friedman.test() e data frames com os valores de cada técnica relativamente às 20 instâncias de maior tamanho, o valor de p-value obtido foi 2.061e-09. Este valor é muito baixo em comparação com 0.05 pelo que se rejeita H0, ou seja, o desempenho das três técnicas apresenta uma diferença significativa.

```
> friedman.test(y = biggest_20_data$Mspan, groups = biggest_20_data$Metodo, blocks = biggest_20_data$problema)

        Friedman rank sum test

data:  biggest_20_data$Mspan, biggest_20_data$Metodo and biggest_20_data$problema
Friedman chi-squared = 40, df = 2, p-value = 2.061e-09
```

Figura 16 - Teste de Friedman para as 20 maiores instâncias

Para determinar qual a técnica com melhor desempenho, será necessário realizar um test post-hoc, assim como foi feito para as 10 instâncias de menor dimensão.

```
    CUN     LPT
LPT 0.0045  -
SPT 7.6e-10 0.0045
```

Figura 17 - Teste de Post-Hoc para as 20 maiores instâncias

Ao analisar o resultado, pode-se afirmar que assim como para as 10 instâncias de menor dimensão, as 20 de maior dimensão também apresentam o CUN como método com melhor desempenho. Isso dá-se ao facto de o p-value apresentado pela comparação dos pares ser extremamente menor quando é realizada a comparação com o CUN.

e) Por vezes usa-se o makespan normalizado, Cnorm para avaliar o desempenho de uma MH. Com, Cnorm = (CMH – COPT)/COPT onde CMH é o makespan de uma MH e COPT é o makespan ótimo. Repita todas as alíneas anteriores usando o makespan normalizado (em vez do makespan).

Seguindo a fórmula do makespan normalizado, foram obtidos os valores de makespan de cada técnica e colocados nos data frames respetivos. Através dos data frames criados, foi possível construir um gráfico onde no eixo das ordenadas possui o valor do makespan normalizado das três MH's e para cada uma das 80 instâncias.

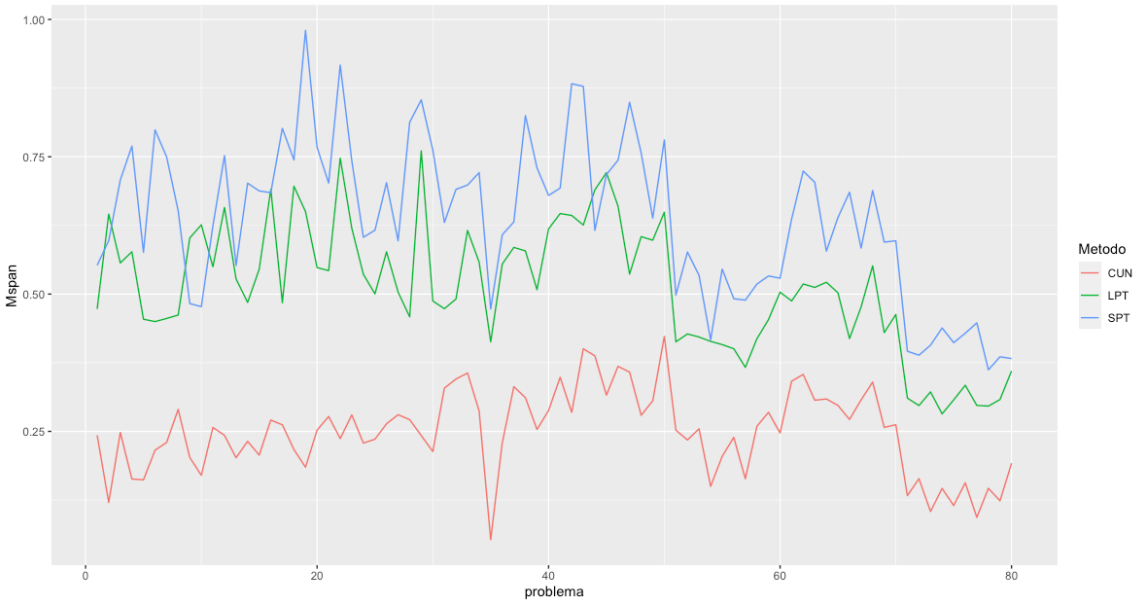


Figura 18 - Gráfico representativo do makespan normalizado obtido por cada MH

O processo de resolução para a obtenção de dados em relação às 10 instâncias de menor tamanho assim como em relação às 20 instâncias de maior tamanho, é de todo igual ao processo referido nas alíneas anteriores. Assim, iremos proceder à demonstração dos resultados obtidos.

Para as 10 instâncias de menor tamanho, obteve-se o seguinte gráfico:

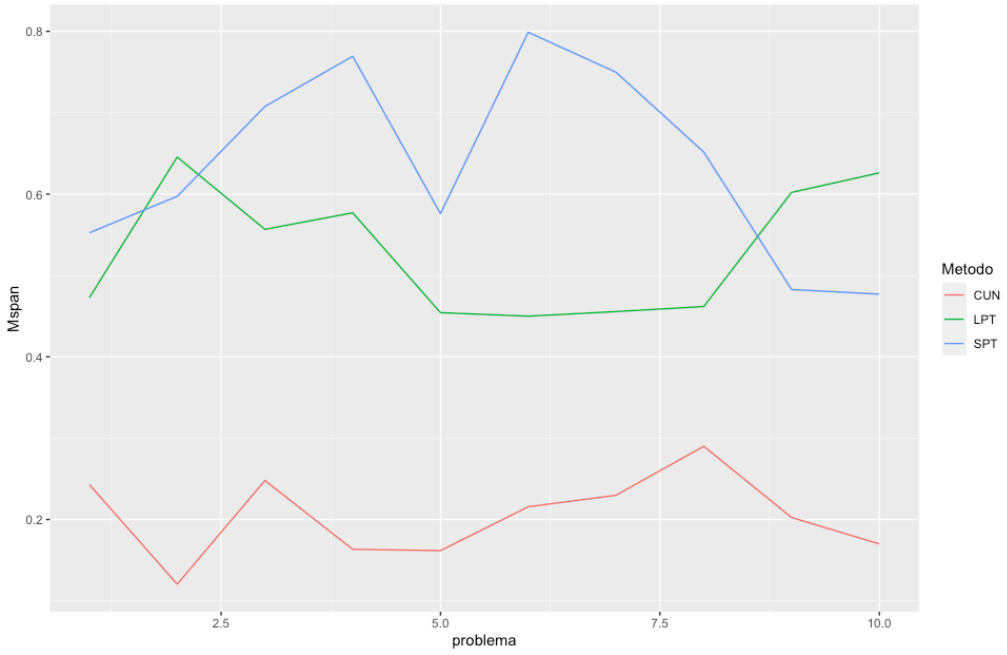


Figura 19 - Gráfico representativo do makespan normalizado das 10 menores instâncias

Para realizar a comparação dos métodos com seus respetivos makespan normalizados, foi recorrido ao test de friedman, pois trava-se de mais de duas amostras emparelhadas. Formularam-se as seguintes hipóteses:

H0: O desempenho é igual entre as três técnicas.

H1: O desempenho é diferente entre as três técnicas.

Através da função friedman.test(), foi passado o data frame das 10 menores instâncias normalizadas e apresentou um p-value de valor 0.0003707. O valor obtido é exatamente igual ao valor retornado no teste realizado com as 10 menores instâncias não normalizadas. Tendo isso em conta, temos de recorrer ao test post-hoc de friedman, também usada nas alíneas anteriores, para determinar qual método é mais eficiente.

```
> friedman.test(Mspan~Metodo|problema, data=smallest_10_data_norm)

        Friedman rank sum test

data:  Mspan and Metodo and problema
Friedman chi-squared = 15.8, df = 2, p-value = 0.0003707
```

Figura 20 - Teste de Friedman para as 10 menores instâncias normalizadas

Para realizar o test post-hoc, será recorrido ao Namenyi test assim como foi feito nas alíneas anteriores.O resultado obtido foi o seguinte

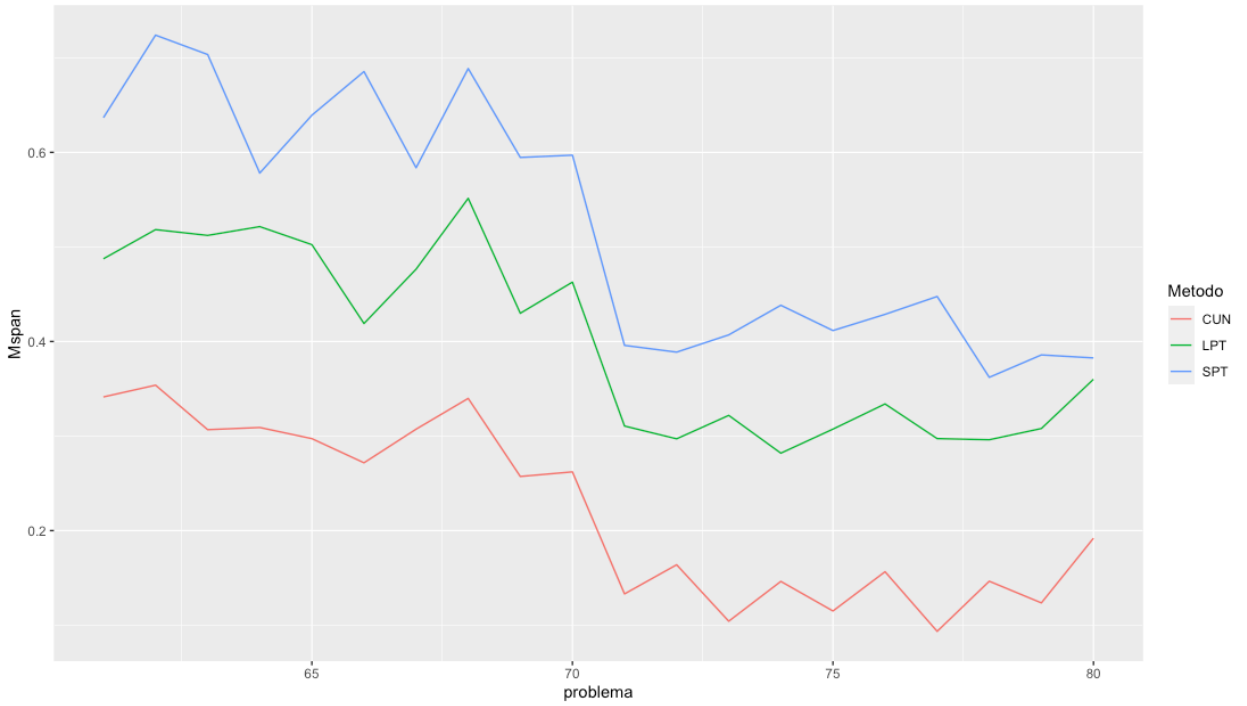
```
data: y, groups and blocks

    CUN     LPT
LPT 0.01021 -
SPT 0.00042 0.64383
```

Figura 21 - Teste de Post-Hoc para as 10 menores instâncias normalizadas

Os valores obtidos através do post-hoc também são exatamente iguais aos dados obtidos com o teste feito as 10 instâncias de menor dimensão. Tendo isso em conta, pode-se concluir que ainda assim, o método CUN tem o melhor desempenho comparado aos outros 2 métodos.

O mesmo processo foi realizado para as 20 instâncias de maior tamanho, obtendo-se o seguinte gráfico:



Para testar as 20 instâncias de maior dimensão com o makespan normalizado, também foi selecionado o teste de friedman, como foi feito para as 10 menores instâncias. Ao executar os testes com as amostras normalizadas, foi retornado o mesmo p-value referente ao teste feito com as 20 maiores instâncias não normalizadas.

```
> friedman.test(y = biggest_20_data_norm$Mspan, groups = biggest_20_data_norm$Metodo, blocks = biggest_20_data_norm$problema)

        Friedman rank sum test

data:  biggest_20_data_norm$Mspan, biggest_20_data_norm$Metodo and biggest_20_data_norm$problema
Friedman chi-squared = 40, df = 2, p-value = 2.061e-09
```

Figura 22 - Teste de Friedman para as 20 maiores instâncias normalizadas

Com isso, para saber qual método teria o melhor desempenho, foi realizado o test post-hoc Namenyi, assim como foi feito nas alíneas anteriores.

```
    CUN     LPT
LPT 0.0045  -
SPT 7.6e-10 0.0045
```

Figura 23 - Teste de Post-Hoc para as 20 maiores instâncias normalizadas

Com os resultados obtidos, pode-se afirmar que o CUN ainda apresenta melhor desempenho comparado com os outros 2 métodos, pois seu p-value retornado na comparação e pares é o menor em relação aos outros pares.

f) Comente os resultados obtidos usando as duas medidas de desempenho usadas (makespan e o makespan normalizado).

Mesmo ao aplicar a fórmula e normalizar os valores do makespan, os valores de teste foram exatamente iguais, indicando que o CUN é a melhor técnica, apresentando uma diferença muito significativa em comparação aos outros métodos.

g) Use os dados relativos às 70 instâncias de menor dimensão. Supondo que o tamanho do problema é a variável preditora e que o makespan normalizado é a variável resposta determine, para cada MH, a reta de regressão linear.

Filtrando, para cada técnica, o data frame de makespan normalizado para que este apenas possua 70 instâncias, obtivemos a regressão linear de cada MH recorrendo à função plot() e lm(). A função plot() vai criar o gráfico com os valores de makespan de cada problema, de seguida, a função lm() cria a reta de regressão linear com auxílio dos dados da variável preditora (problema) e variável resposta (Mspan). Para cada técnica, foram obtidos os seguintes gráficos:

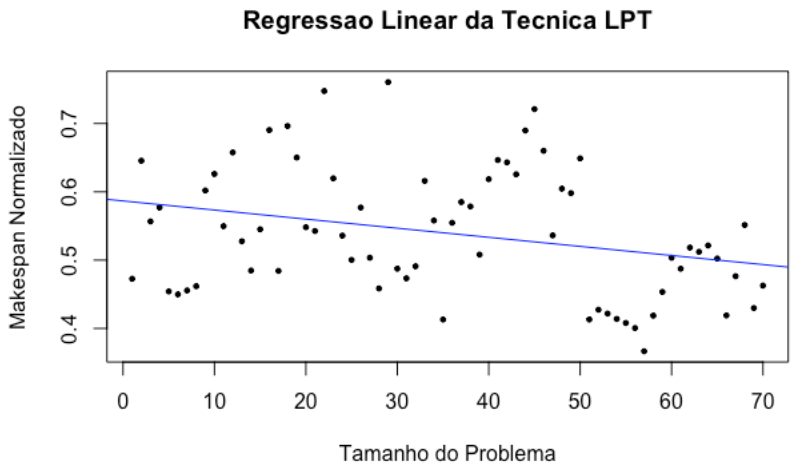


Figura 24 - Regressão Linear LPT

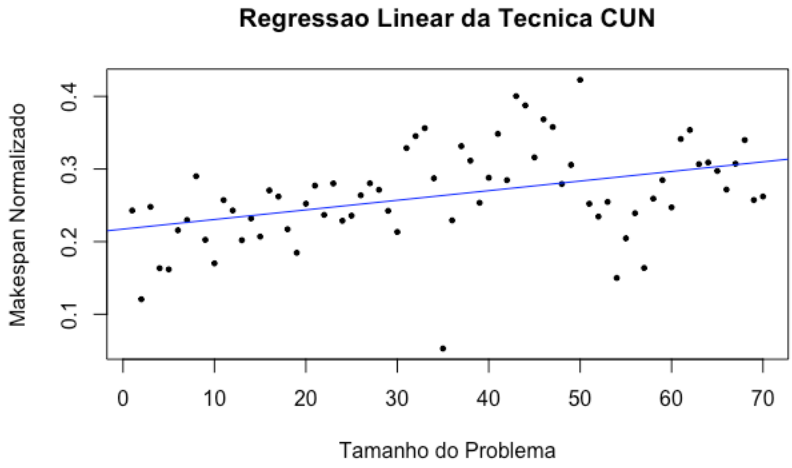


Figura 25 - Regressão Linear CUN

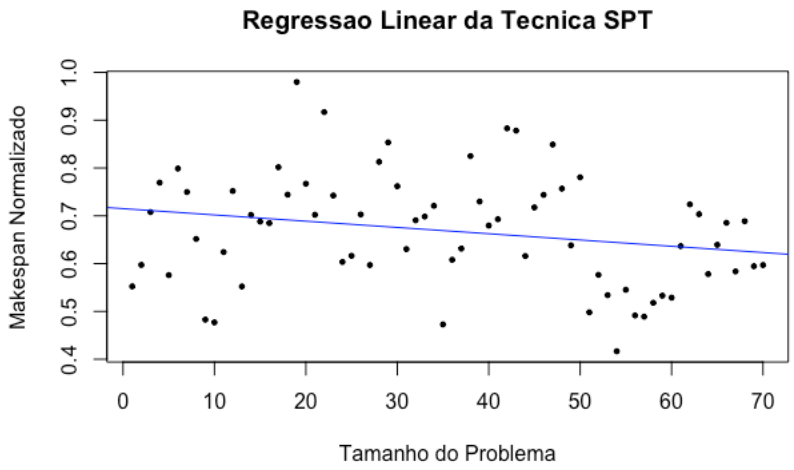


Figura 26 - Regressão Linear SPT

Como a função lm() obtém a reta de regressão linear, foram armazenados em variáveis os valores que definem a reta de cada MH. Para a técnica LPT, a reta de regressão linear é dada por:

Mspan = -0.0013*problema + 0.59

Para a técnica CUN, a seguinte expressão:

Mspan = 0.0013*problema + 0.217

Por fim, para a técnica SPT a reta é definida por:

Mspan = -0.0013*problema + 0.715

As retas de regressão linear possuem todas um coeficiente beta positivo. A única técnica com um declive positivo é a técnica CUN enquanto as restantes apresentam um declive negativo. Ao ser um declive positivo, a correlação entre o tamanho do problema e o makespan é também positiva, ou seja, os fenómenos variam no mesmo sentido. Pelo contrário, se for negativo, os fenómenos variam em sentido inverso.

h) Verifique se os pressupostos sobre os resíduos são verificados (normalidade, homocedasticidade e independência).

- **<u>Normalidade</u>**

Para verificar a normalidade dos resíduos, foram utilizados dois processos recomendados. Através do processo gráfico, utilizamos as funções *qqnorm()* e *qqline()* que vão apresentar um gráfico com base na regressão linear de cada MH. Caso os pontos do gráfico estejam aproximadamente em cima da linha, então verifica-se normalidade dos resíduos. De seguida, recorremos ao teste de Shapiro, shapiro.test(), e com base no valor do p-value, concluímos se os resíduos verificam a condição de normalidade. Caso o valor de p-value for superior a 0.05, verifica-se normalidade, caso contrário, não se verifica a condição de normalidade.

Para cada MH, foram obtidos os seguintes resultados:

- **LPT**

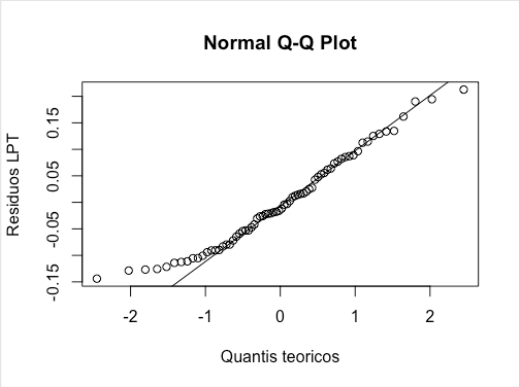


Figura 27 - Gráfico Q-Q para verificar normalidade dos resíduos de LPT

```
        Shapiro-Wilk normality test

data:  residuals(reg_LPT)
W = 0.9646, p-value = 0.04519
```

Figura 28 - Teste de Shapiro para verificar a normalidade dos resíduos de LPT

```
        Lilliefors (Kolmogorov-Smirnov) normality test

data:  residuals(reg_LPT)
D = 0.071221, p-value = 0.5105
```

Figura 29 - Teste de Lilliefors para verificar a normalidade dos resíduos de LPT

Como o valor do p-value obtido no teste de Shapiro é próximo de 0.05, foi realizado ainda um teste de Lilliefors para confirmar suspeitas. Contudo, no teste de Lilliefors, obteve-se um p-value relativamente maior que o p-value obtido no teste de Shapiro. Assim, em último recurso, com a observação do gráfico, decidimos rejeitar a normalidade dos resíduos para a técnica LPT.

- **CUN**

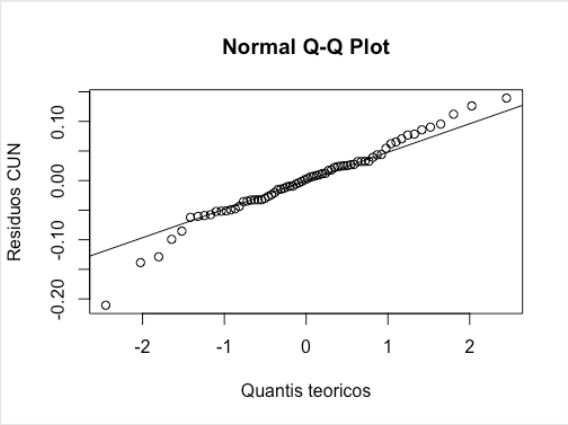


Figura 30 - Gráfico Q-Q para verificar normalidade dos resíduos de CUN

```
        Shapiro-Wilk normality test

data:  residuals(reg_CUN)
W = 0.97242, p-value = 0.125
```

Figura 31 - Teste de Shapiro para verificar a normalidade dos resíduos de CUN

- **SPT**

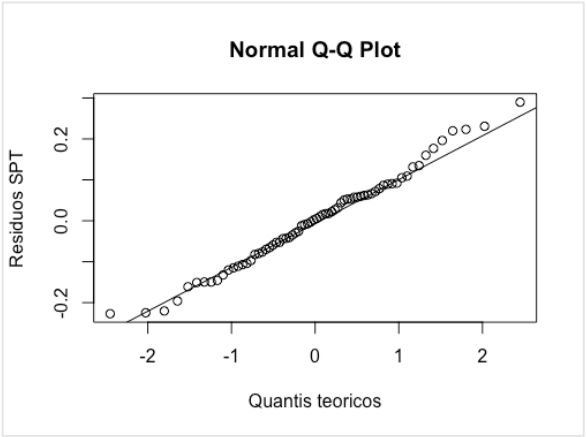


Figura 32 - Gráfico Q-Q para verificar normalidade dos resíduos de SPT

```
        Shapiro-Wilk normality test

data:  residuals(reg_SPT)
W = 0.98819, p-value = 0.756
```

Figura 33 - Teste de Shapiro para verificar a normalidade dos resíduos de SPT

Com a observação do gráfico (qq-normal-plot) de ambas as técnicas CUN e SPT, assim como o resultado obtido no teste de Shapiro, com um valor de p-value superior a 0.05, podemos concluir que se verifica a condição de normalidade para os resíduos CUN e SPT e podemos efetuar inferência estatística. Apesar de não se verificar a condição de normalidade para a técnica LPT, iremos testar ainda a homocedasticidade e independência dos resíduos desta técnica.

- **Homocedasticidade**

De seguida, para verificar a homocedasticidade dos resíduos recorremos primeiro ao método gráfico e criamos um plot dos resíduos vs. os valores ajustados. De seguida, dividimos os dados em dois conjuntos e testamos se a variância seria a mesma, var.test().

Para cada MH, foram obtidos os seguintes resultados:

  - **LPT**

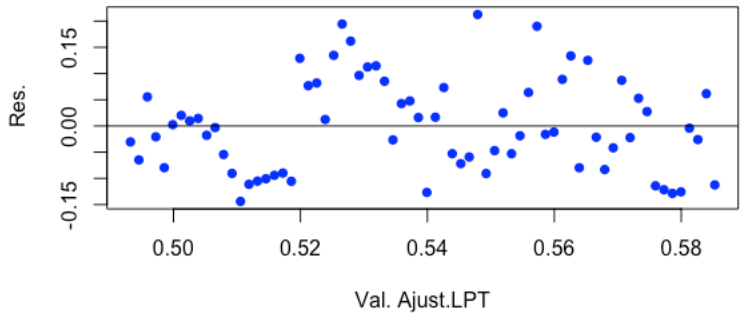


Figura 34 - Plot resíduos LPT

```
        F test to compare two variances

data:  residuals(reg_LPT)[smallest_70_LPT_norm$problema > mx_LPT] and residuals(reg_LPT)[smallest_70_LPT_norm$problema < mx_LPT]
F = 0.96169, num df = 34, denom df = 34, p-value = 0.91
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4854297 1.9052307
sample estimates:
ratio of variances
         0.9616941
```

Figura 35 - var.test() resíduos LPT

- **CUN**

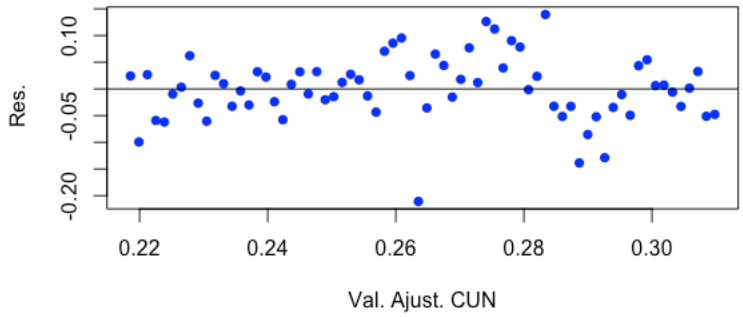


Figura 36 - Plot resíduos CUN

```
        F test to compare two variances

data:  residuals(reg_CUN)[smallest_70_CUN_norm$problema > mx_CUN] and residuals(reg_CUN)[smallest_70_CUN_norm$problema < mx_CUN]
F = 1.3489, num df = 34, denom df = 34, p-value = 0.3872
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6808863 2.6723655
sample estimates:
ratio of variances
          1.348917
```

Figura 37 - var.test() resíduos CUN

- **SPT**

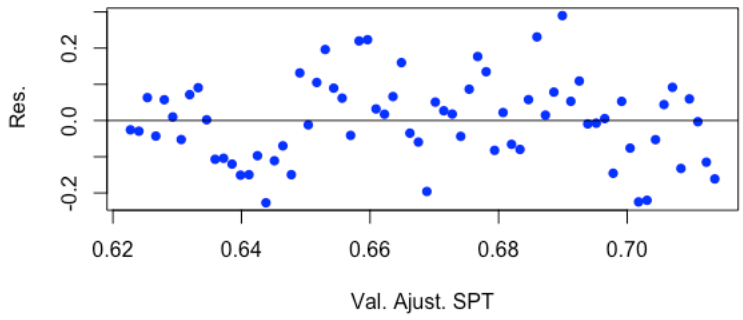


Figura 38 - Plot resíduos SPT

```
        F test to compare two variances

data:  residuals(reg_SPT)[smallest_70_SPT_norm$problema > mx_SPT] and residuals(reg_SPT)[smallest_70_SPT_norm$problema < mx_SPT]
F = 0.87785, num df = 34, denom df = 34, p-value = 0.7063
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4431106 1.7391352
sample estimates:
ratio of variances
         0.8778549
```

Figura 39 - var.test() resíduos SPT

Para cada técnica, o valor do p-value obtido no teste de variância, foi superior a 0.05, pelo que podemos assumir que a condição de homocedasticidade é verificada.

- **Independência**

Para verificar se os resíduos são independentes, formulam-se as seguintes hipóteses:

H0: Os resíduos são independentes.

H1: Os resíduos não são independentes.

Com a efetuação do um teste de Durbin-Watson, reparamos que os valores de p-value, para cada técnica, estavam constantemente a mudar, pelo que decidimos correr o teste várias vezes, colocar os valores de p-value num vetor e de seguida correr um t.test(). Assim, para cada método, obtivemos os seguintes resultados:

- **LPT**

```
data:  vec_LPT
t = -1872.8, df = 9, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 -6.483820e-06  2.406484e-03
sample estimates:
mean of x
   0.0012
```

Figura 40 - t.test() para verificar a independência dos resíduos LPT

- **CUN**

```
    One Sample t-test

data:  vec_CUN
t = -1716.5, df = 9, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 0.001085265 0.003714735
sample estimates:
mean of x
   0.0024
```

Figura 41 - t.test() para verificar a independência dos resíduos CUN

- **SPT**

```
    One Sample t-test

data:  vec_SPT
t = -4999, df = 9, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 -0.0002524314  0.0006524314
sample estimates:
mean of x
    2e-04
```

Figura 42 - t.test() para verificar a independência dos resíduos SPT

Com o teste de Durbin-Watson, os valores de p-value já eram bastante pequenos e inferiores a 0.05, pelo que a hipótese nula seria rejeitada. Contudo, através do t.test() podemos confirmar realmente essa conclusão. Sendo o valor de p-value igual a 2.2e-16, este é um valor muito baixo pelo que a hipótese nula é rejeitada e concluímos que os resíduos não são independentes.

4.2.2. Exercício 2

a) Construa um boxplot que contenha os tempos de processamento de cada MH na resolução de cada instância.

Para resolver esta alínea, recorreu-se à função *boxplot()* que recebe por parâmetro a informação relativa às técnicas CUN, SPT e LPT, ou seja, da coluna 3 à coluna 5, sendo que o eixo das abcissas contém a variável dos “Segundos” e o eixo das ordenadas a variável que identifica os métodos. Obtivemos o seguinte gráfico:

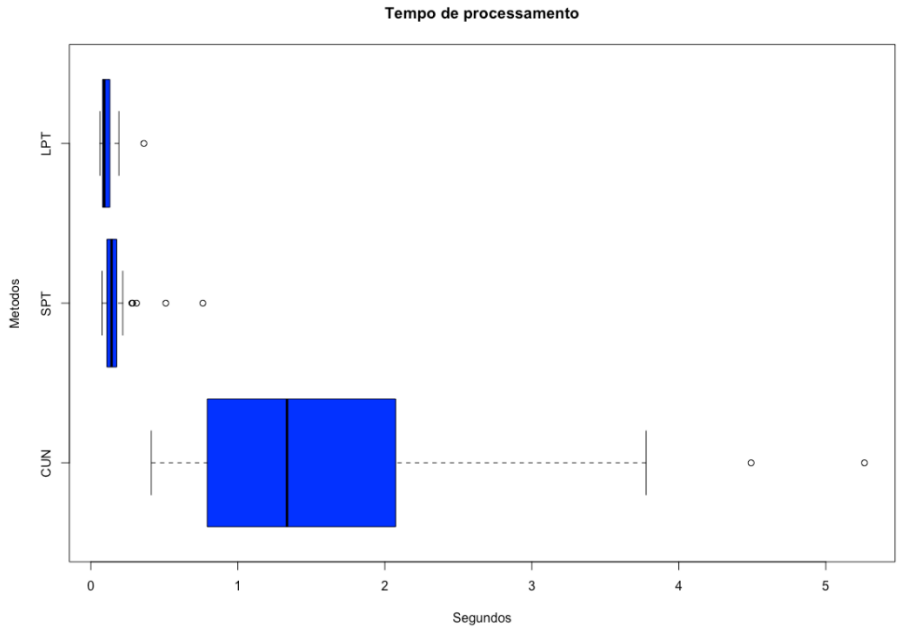


Figura 43 - Boxplot de Tempo de Processamento

Com a observação do gráfico, podemos concluir que CUN é uma técnica que por mais que desempenhe melhor, como foi concluído nos testes com o makespan, apresenta um elevado tempo de processamento e outliers significativos, assim como as outras 2 técnicas também apresentam.

b) Verifique se existem diferenças significativas nos tempos médios de processamento entre as três técnicas.

Para verificar se existe diferença significativa entre os tempos médios de processamento, foi selecionado o teste de Friedman, pois o data frame referente aos tempos de processamento contempla mais de duas amostras emparelhadas, dessa forma, o mais indicado teste a ser usado é o Friedman.

Com isso, formularam-se as seguintes hipóteses:

H0: O desempenho é igual entre as três técnicas.

H1: O desempenho é diferente entre as três técnicas.

Para realizar o teste, foi feito um tratamento para que os dados estivessem agrupados em coluna, sendo a primeira coluna referente ao problema, a segunda linha referente ao método e a terceira linha referente ao tempo de processamento. Com isso, a informação referente a cada problema fica agrupada em linhas.

	Problema	Metodo	Tempo
1	1	SPT	0.5099798
2	2	SPT	0.1858872
3	3	SPT	0.2785206
4	4	SPT	0.0761436
5	5	SPT	0.1411127
6	6	SPT	0.0950125
7	7	SPT	0.1047737
8	8	SPT	0.1485716
9	9	SPT	0.0979392
10	10	SPT	0.2170957
11	11	SPT	0.2017321
12	12	SPT	0.7618296
13	13	SPT	0.1879717
14	14	SPT	0.1682512
15	15	SPT	0.2843661
16	16	SPT	0.1006248
17	17	SPT	0.0965347
18	18	SPT	0.1202390
19	19	SPT	0.1139519

Figura 44 - Data frame com os métodos numa só coluna

Para executar o teste de Friedman, recorreu-se a função friedman.test(), passando por parâmetro as colunas do data frame (Tempo, Metodo e Problema).

Ao executar o teste, foi concebido o seguinte resultado.

```
        Friedman rank sum test

data:  runtime_all_in_one$Tempo, runtime_all_in_one$Metodo and runtime_all_in_one$Problema
Friedman chi-squared = 94.12, df = 2, p-value < 2.2e-16

>
```

Figura 45 - Teste de Friedman para os tempos de processamento

Ao analisar o p-value, pode-se afirmar que seu valor é muito inferior ou nível de significância de 0.05, dessa forma, rejeita-se a hipótese nula. Portanto, há diferença significativa no tempo médio de processamento das 3 técnicas avaliadas.

c) No caso da resposta da alínea anterior ser positiva, identifique qual a MH mais eficiente. Determine a matriz de correlação entre os tempos de processamento de cada MH e interprete os resultados.

Como foi rejeitada a hipótese nula no tópico anterior, afirmou-se que há diferença significativa nos tempos médios de processamento. Para conseguir indicar qual método tem melhor performance, deverá ser realizar um teste post-hoc de friedman, denominado teste de namenyi para comparar as amostras duas a duas.

Para ser possível executar o teste post-hoc, recorreu-se a função MCMRplus::frdAllPairsConoverTest() e foi retornado os seguintes resultados.

```
data: y, groups and blocks

    CUN     LPT
LPT 3.6e-14 -
SPT 1.7e-06 7.8e-06
```

Figura 46 - Teste de Post-Hoc para os tempos de processamento

Com isso, pode-se afirmar que a diferença significante está indicada no método LPT, sendo esse o com melhor desempenho, pois a comparação de dois em dois apresenta um menor p-value quando compara o LPT com as outras varáveis.

d) Determinar a matriz de correlação entre os tempos de processamento de cada MH e interprete os resultados

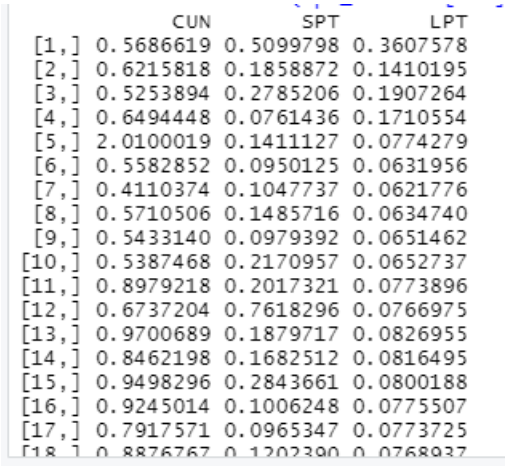
Para determinar a matriz de correlação entre os tempos de processamento, foi feito uma breve análise sobre qual teste seria selecionado para realizar a matriz de correlação. A primeira opção foi o teste de Pearson.

Para saber se de facto esse teste poderia ser utilizador, foi feito a avaliação dos seus pressupostos:

- Não devem existir outliers significativos
    - Como foi representado pelas boxplots no primeiro execício, outliers significativos foram detetados nas amostras. Tendo isso em conta, o teste de Pearson será descartado.

Restam então outras duas alternativas, ou o teste de Spearman ou o teste de Kendall. O teste de Spearman será o escolhido pois é o mais indicado para amostras grandes, diferente do teste de Kendall, que é recomendado para amostras pequenas ou quando há empates no teste de Spearman.

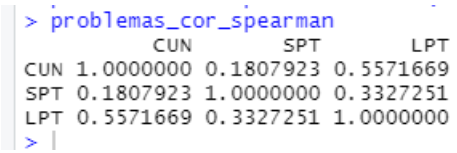
Para realizar o teste, os tempos de processamento foram convertidos em uma matriz, em que cada linha obtinha o tempo e cada coluna o método de processamento, resultando os seguintes dados.



```
            CUN       SPT       LPT
 [1,] 0.5686619 0.5099798 0.3607578
 [2,] 0.6215818 0.1858872 0.1410195
 [3,] 0.5253894 0.2785206 0.1907264
 [4,] 0.6494448 0.0761436 0.1710554
 [5,] 2.0100019 0.1411127 0.0774279
 [6,] 0.5582852 0.0950125 0.0631956
 [7,] 0.4110374 0.1047737 0.0621776
 [8,] 0.5710506 0.1485716 0.0634740
 [9,] 0.5433140 0.0979392 0.0651462
[10,] 0.5387468 0.2170957 0.0652737
[11,] 0.8979218 0.2017321 0.0773896
[12,] 0.6737204 0.7618296 0.0766975
[13,] 0.9700689 0.1879717 0.0826955
[14,] 0.8462198 0.1682512 0.0816495
[15,] 0.9498296 0.2843661 0.0800188
[16,] 0.9245014 0.1006248 0.0775507
[17,] 0.7917571 0.0965347 0.0773725
[18,] 0.8876767 0.1202390 0.0768937
```

Figura 47 - Matriz com os tempos de processamento

Após executar o teste, os seguintes coeficientes de correlação foram apresentados:



```
> problemas_cor_spearman
          CUN       SPT       LPT
CUN 1.0000000 0.1807923 0.5571669
SPT 0.1807923 1.0000000 0.3327251
LPT 0.5571669 0.3327251 1.0000000
>
```

Figura 48 - Coeficientes de correlação para cada MH

Ao analisar os resultados, pode-se chegar as seguintes conclusões:

- SPT e CUN são fracamente correlacionadas devido ao coeficiente de correlação estar próximo de 0.
- SPT e LPT são fracamente correlacionadas devido ao coeficiente de correlação estar próximo de 0.
- LPT e CUN são positivamente correlacionadas devido ao coeficiente de correlação estar próximo de 1.

Para visualizar melhor ainda os dados, foi desenvolvido um corplot resultando a seguinte imagem.

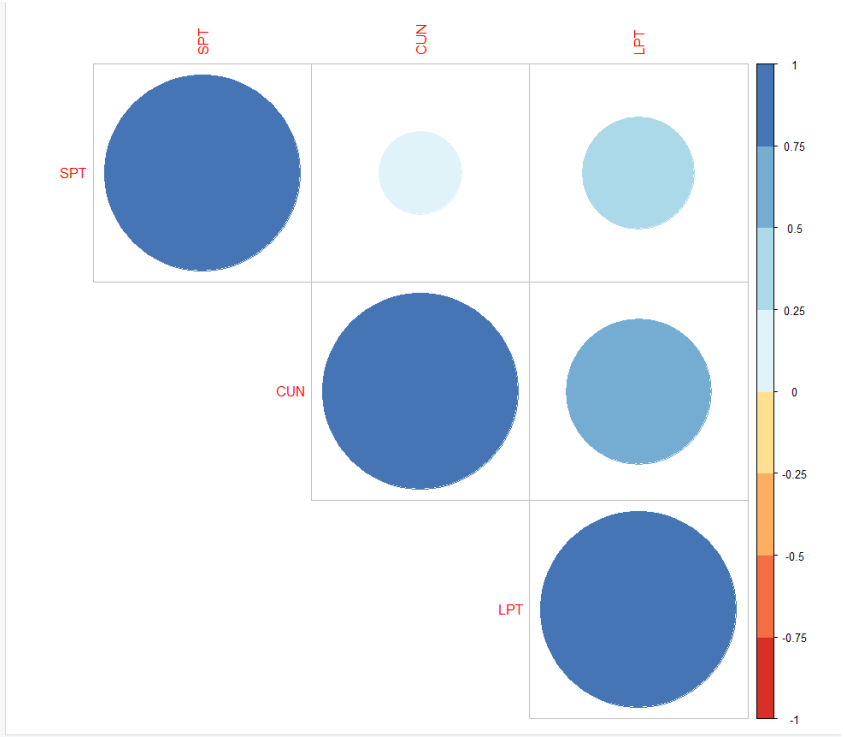


Figura 49 - cor.plot() com base na matriz de correlação

5. Conclusões

Apesar de todas as conclusões feitas ao longo da análise e discussão dos resultados, para cada exercício podemos reforçar as conclusões obtidas.

No problema da “Análise do funcionamento dos servidores VPN do DEI” concluímos que o servidor com maior número de acessos total assim como o servidor com a maior media, mediana e desvio padrão mensal de meses completos, é o servidor “vsrv16”. Contrariamente, o servidor com menor número de acessos, media, mediana e desvio padrão mensal, é o servidor “vsrv10”. De seguida, uma vez que existem mais acessos do que falhas, o número de acessos simultâneos em comparação com o número de falhas simultâneas é muito maior, o que seria de esperado. Em termos de falhas diárias, concluímos que o servidor “vsvr16” é o servidor com maior número de falhas diárias apesar de o servidor “vsrv8” possuir um maior número de falhas total.

No problema da “Análise de Desempenho de Métodos Heurísticos na resolução do problema de Escalonamento”, concluímos que a técnica mais eficaz em termos de valor de makespan, ou seja, a que possui um menor valor de makesapn e que se encontra mais próximo do makespan ótimo (OPT), é a técnica CUN, quer com o makespan normal quer com o makespan normalizado, quer para as 10 menores instâncias quer para as 20 maiores. Quanto aos resíduos da regressão linear de cada técnica, concluímos que nenhum deles verifica a condição de independência. No que tem em conta a eficiência das MH, considerando o tempo de processamento, a técnica mais eficiente é a técnica LPT uma vez que possui o menor tempo de execução como pode ser confirmado pela construção de boxplots e testes. Com a determinação da matriz de correlacao, concluímos que a técnica SPT e CUN assim como SPT e LPT são fracamente correlacionadas devido ao coeficiente de correlação estar próximo de 0, e as técnicas LPT e CUN são positivamente correlacionadas devido ao coeficiente de correlação estar próximo de 1.