Relatório Mini-Projeto 2

Modelação e Desempenho de Redes e Serviços

Ana Ferreira 93301 (aluisaf@ua.pt)
João Ferreira 93305 (joaodiogohf@ua.pt)

fevereiro 2022

1. Tarefa 1

1.1. Alínea a.

1.1.1. Resultados

Flow 1: 32
Flow 2: 32
Flow 3: 38
Flow 4: 24
Flow 5: 36
Flow 6: 37
Flow 7: 25
Flow 8: 41
Flow 9: 28

1.1.2. Discussão

É possível concluir, observando os caminhos criados para cada fluxo, que existem muitas combinações, porque o caminho pode percorrer um conjunto muito grande de ligações, inclusivamente ir até nós mais afastados da origem do que o nó destino, gerando um conjunto relativamente grande de soluções, por exemplo para o fluxo 8, foram obtidos 41 caminhos.

1.2. Alínea b.

1.2.1. Resultados

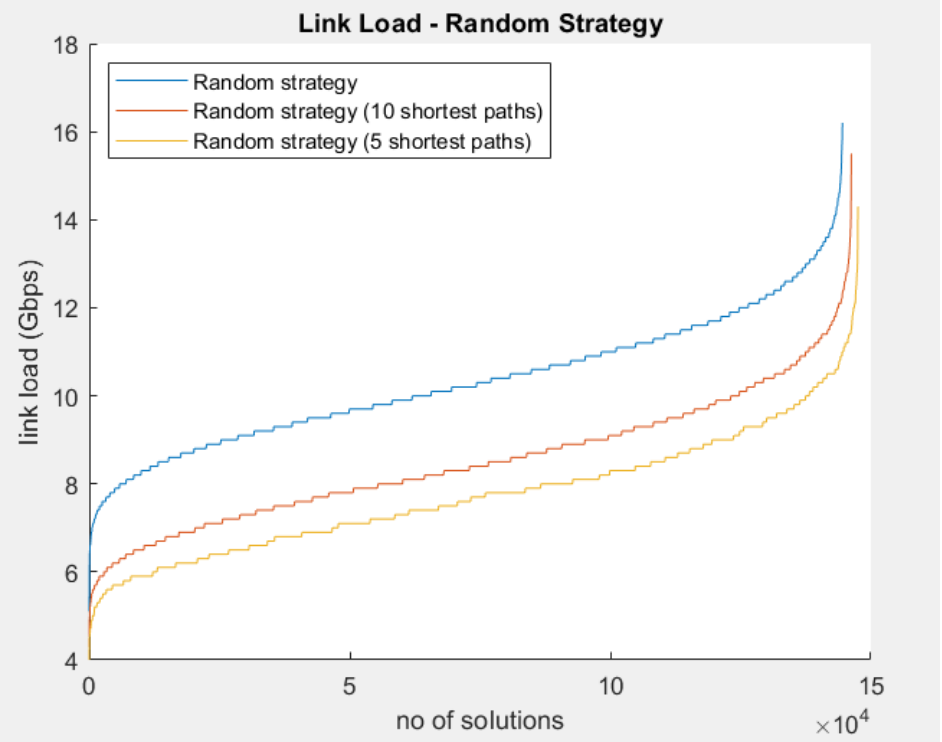

Gráfico 1: Resultados da carga máxima, entres todas as ligações, nas soluções obtidas, por ordem crescente, para a estratégia random.

RANDOM STRATEGY:
Best load = 4.90 Gbps
Worst load = 16.20 Gbps
No. of solutions = 144777
Av. quality of solutions = 10.32 Gbps
Best solution:
1 1 16 14 12 2 5 2 1

RANDOM STRATEGY (10 shortest paths):
Best load = 4.40 Gbps
Worst load = 15.50 Gbps
No. of solutions = 147014
Av. quality of solutions = 8.51 Gbps
Best solution:
3 1 3 8 9 2 3 1 1

RANDOM STRATEGY (5 shortest paths):
Best load = 4.00 Gbps
Worst load = 14.30 Gbps
No. of solutions = 145147
Av. quality of solutions = 7.74 Gbps
Best solution:
1 5 2 2 4 2 5 2 3

Nota:

- O worst load, load máximo obtido entre todas as soluções, foi calculado de forma complementar na comparação dos resultados obtidos.
- O best load refere ao load máximo obtido para a melhor solução.

1.2.2. Discussão

Usando a random strategy, é possível observar que nos 3 resultados, para os diferentes caminhos de menor custo, o número de soluções obtidas foram muito próximas umas das outras, porque para este algoritmo o que influencia é o tempo de processamento do cálculo da carga das ligações de cada solução. Visto que este tempo é independente do número de caminhos usados e como são corridos durante o mesmo tempo, 10 segundos, todas as versões do algoritmo apresentam resultados semelhantes. É possível comprovar, com os resultados obtidos, o que foi apresentado nas aulas teóricas, que quando este algoritmo é usado com caminhos de "melhores características", ou seja, os caminhos mais curtos que, no geral, utilizam menos links para o mesmo fluxo, obtém-se melhores resultados.

1.3. Alínea c.

1.3.1. Resultados

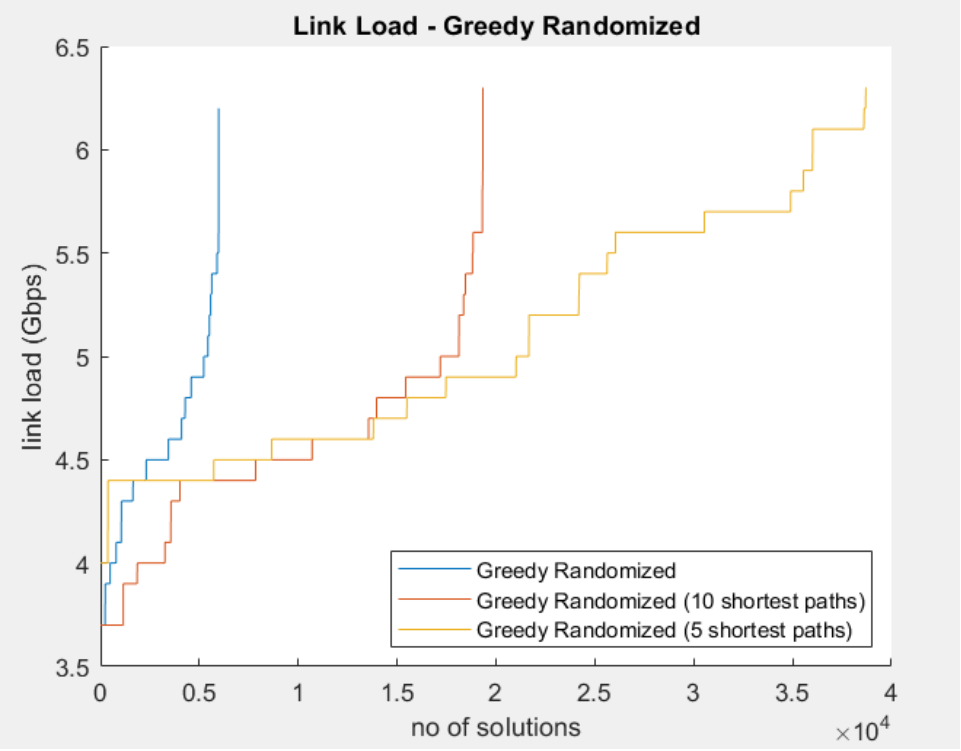

Gráfico 2: Resultados da carga máxima, entres todas as ligações, nas soluções obtidas, por ordem crescente, para a estratégia greedy randomized.

GREEDY RANDOMIZED:
Best load = 3.70 Gbps
Worst load = 6.20 Gbps
No. of solutions = 5981

Av. quality of solutions = 4.54 Gbps
Best solution:
1 1 4 1 1 1 10 7 1
GREEDY RANDOMIZED (10 shortest paths):
Best load = 3.70 Gbps
Worst load = 6.30 Gbps
No. of solutions = 19340
Av. quality of solutions = 4.53 Gbps
Best solution:
1 1 3 1 1 4 10 1 1

GREEDY RANDOMIZED (5 shortest paths):
Best load = 4.00 Gbps
Worst load = 6.30 Gbps
No. of solutions = 38723
Av. quality of solutions = 5.07 Gbps
Best solution:
1 2 2 2 1 3 4 1 1

1.3.2. Discussão

Usando o greedy randomized, é possível observar que ao usar menos caminhos, este gera mais soluções, visto que o algoritmo inicia colocando os fluxos numa ordem aleatória, e de seguida, para cada caminho de cada fluxo calcula os link loads, como existem menos caminhos, o cálculo do link loads é feito menos vezes. Com isto, é possível concluir que o número de caminhos de cada fluxo, influencia o tempo de processamento de cada solução. Ao reduzir o número de caminhos usados para metade é possível observar que o número de soluções obtidas aumentou o dobro.
Tanto para o algoritmo que usa todos os caminhos como para o algoritmo que usa apenas os 10 caminhos mais curtos, o best load obtido foi 3.70 Gbps. Para o algoritmo que usa os 5 caminhos mais curtos esse valor foi de 4.00 Gbps. Uma vez que são excluídas combinações de caminhos para a obtenção das soluções, pode estar a ser excluída a solução que nos dá o melhor load, porque esta pode já não se encontrar nas combinações dadas pelo subconjunto de caminhos que estamos a disponibilizar.
Relativamente à qualidade média dos resultados obtidos, o melhor resultado 4.52 Gbps foi obtido para o algoritmo que usa os 10 caminhos mais curtos, sendo esse valor quase igual ao valor obtido usando todos os caminhos (4.54 Gbps). Para o algoritmo que usa os 5 caminhos mais curtos esse valor foi de 5.07 Gbps. Uma vez que, como explicado anteriormente, se as nossas soluções obtidas pioraram, a média final será maior.
Pode-se concluir que, para este problema de otimização em concreto, usando todos os caminhos ou apenas os 10 caminhos mais curtos os resultados para o best load são melhores. Ainda que, no primeiro caso, a média dos resultados obtidos seja pior.

1.4. Alínea d.

1.4.1. Resultados

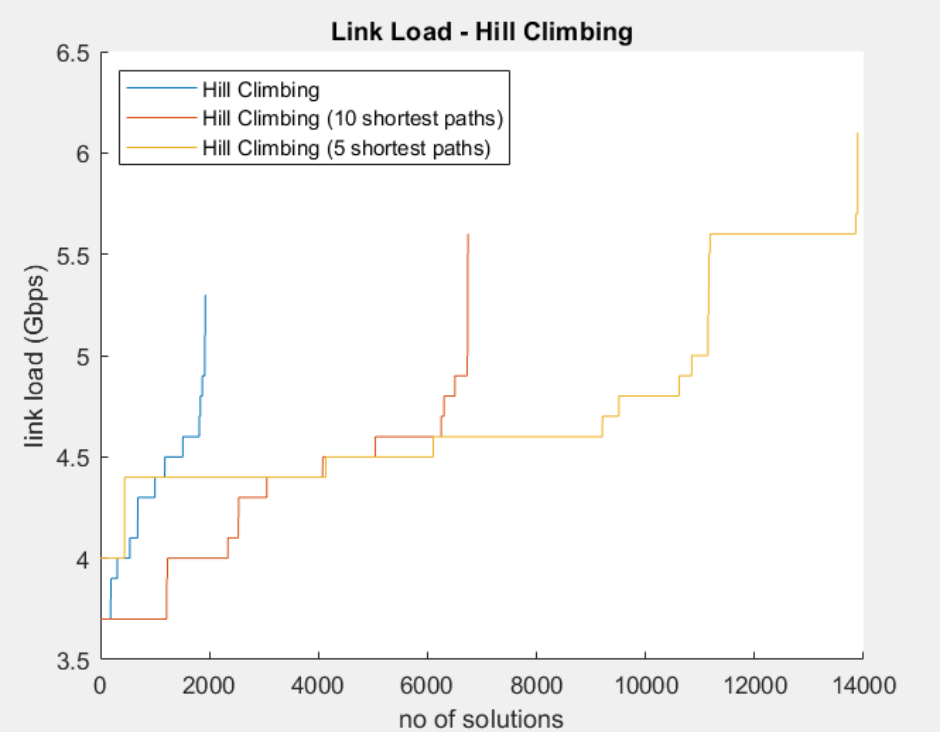

Gráfico 3: Resultados da carga máxima, entres todas as ligações, nas soluções obtidas, por ordem crescente, para a estratégia de hill climbing.

MULTI START HILL CLIMBING:
Best load = 3.70 Gbps
Worst load = 5.30 Gbps
No. of solutions = 1925
Av. quality of solutions = 4.29 Gbps
Best solution:
1 1 3 1 1 4 10 1 1

MULTI START HILL CLIMBING (10 shortest paths):
Best load = 3.70 Gbps
Worst load = 5.60 Gbps
No. of solutions = 6749
Av. quality of solutions = 4.27 Gbps
Best solution:
1 1 4 1 1 1 10 7 1

MULTI START HILL CLIMBING (5 shortest paths):
Best load = 4.00 Gbps

Worst load = 6.10 Gbps
No. of solutions = 13893
Av. quality of solutions = 4.74 Gbps
Best solution:
1 2 2 2 3 2 2 1 1

1.4.2. Discussão

Usando o algoritmo hill climbing é possível observar que mais uma vez à medida que diminuímos o número de percursos utilizados, neste caso, até 5, o número de soluções obtidas aumenta, visto que tal como no greedy randomized, o algoritmo para cada fluxo percorre cada percurso, se tivermos menos percursos para percorrer, vamos gastar menos tempo de processamento, assim conseguimos processar mais soluções.
Porém, com menos caminhos usados, obtivemos um load maior, de 4.00 Gbps, do que nas outras abordagens, onde foi 3.70Gbps para ambas, aqui também a qualidade média foi pior. Isto pode acontecer, porque ao diminuirmos os caminhos, deixam-se de ter certas combinações de caminhos para explorar e provavelmente exclui-se o caminho da solução com melhores resultados. Observa-se que ao usar 10 caminhos, no sétimo fluxo o melhor percurso é o 10, ao usar 5 caminhos, percursos explorados ficam limitados, esta solução já não aparece.

1.5. Alínea e.

1.5.1. Discussão

Comparando todos os algoritmos, conseguimos observar que o best load obtido foi de 3.70Gbps, sendo esta solução obtida em dois algoritmos, no greedy randomized e no hill climbing. Comparando estes 2 algoritmos percebemos que a estratégia de hill climbing em média encontra melhores valores, obtendo um menor número de soluções, uma vez que é mais elaborada e computacionalmente mais exigente. Se ambos os algoritmos chegam ao mesmo valor para o best load, neste caso, pode-se optar pela estratégia greedy, uma vez que é mais simples.
Quanto ao número de percursos usados, os melhores resultados obtêm-se usando pelo menos os 10 caminhos mais curtos, sendo que em média registaram-se melhores resultados quando usamos todos os percursos. Os piores resultados foram obtidos usando apenas os 5 caminhos mais curtos.
Relativamente à estratégia random, ao contrário do que acontecia nas outras estratégias, os resultados melhoram progressivamente quando são usados menos caminhos, sendo o tempo de execução constante para qualquer conjunto de caminhos considerado. Esta estratégia apesar de conseguir encontrar mais soluções, no mesmo intervalo de tempo, relativamente aos restantes algoritmos, obteve piores resultados, uma vez que é computacionalmente mais simples e não usa qualquer critério para explorar as soluções.

2. Tarefa 2

2.1. Alínea a.

2.1.1. Resultados

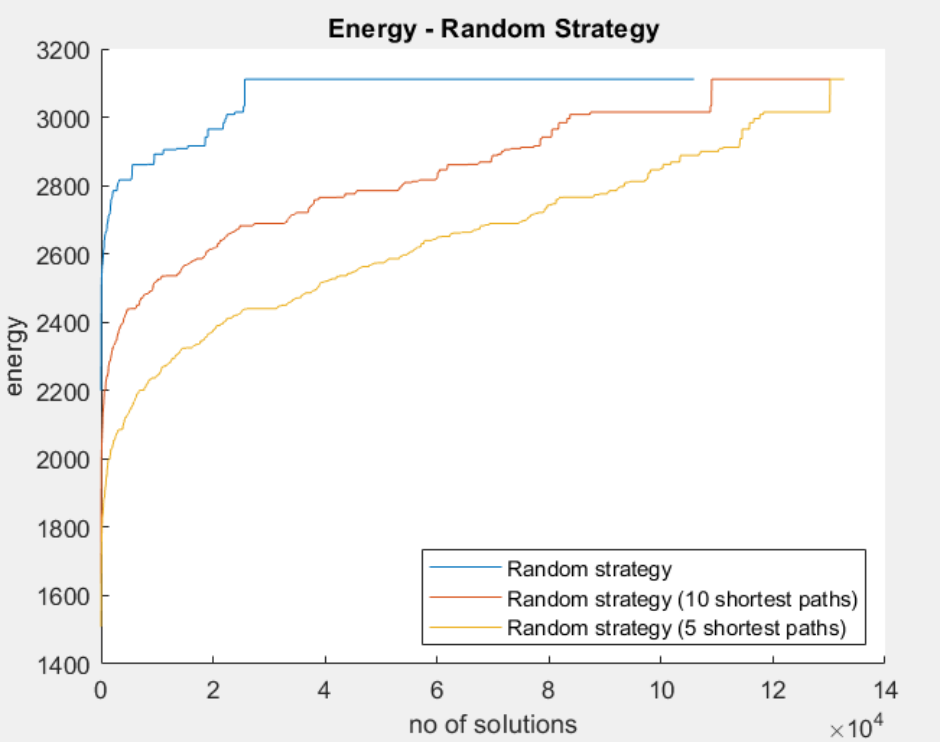

Gráfico 4: Resultados da energia total nas soluções obtidas, por ordem crescente, para a estratégia random.

RANDOM ALGORITHM:
Best energy = 2200.0
No. of solutions = 105865
Av. quality of solutions = 3057.2
Best solution:
16 1 4 9 8 7 9 3 1

RANDOM ALGORITHM (10 shortest paths):
Best energy = 1635.0
No. of solutions = 132230
Av. quality of solutions = 2847.2
Best solution:
2 1 1 1 6 3 7 9 1

RANDOM ALGORITHM (5 shortest paths):
Best energy = 1509.0
No. of solutions = 132692

Av. quality of solutions = 2646.3
Best solution:
2 1 1 4 2 2 5 1 1

2.1.2. Discussão

Usando a random strategy, é possível observar que nos 3 resultados, para os diferentes caminhos de menor custo, o número de soluções obtidas foram muito próximas umas das outras, tal como aconteceu na tarefa 1.
É possível comprovar, com os resultados obtidos, o que foi apresentado nas aulas teóricas, que quando este algoritmo é usado com caminhos de "melhores características", ou seja, os caminhos mais curtos que, no geral, utilizam menos links para o mesmo fluxo, obtém-se melhores resultados. Já que segundo o enunciado o tamanho do link é proporcional à energia consumida, logo se utilizam menos links para o mesmo fluxo pode-se esperar que assim a energia consumida seja menor.

2.1.3. Alínea b.

2.1.4. Resultados

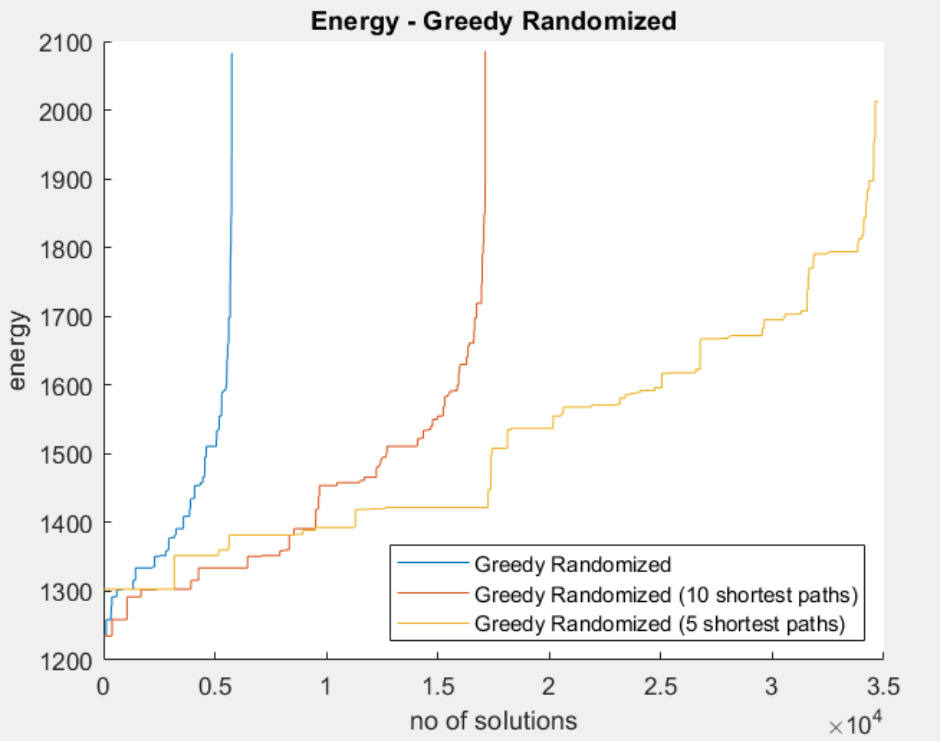

Gráfico 5: Resultados da energia total nas soluções obtidas, por ordem crescente, para a estratégia greedy randomized.

GREEDY RANDOMIZED:
Best energy = 1235.0

No. of solutions = 5754
Av. quality of solutions = 1397.9
Best solution:
10 4 4 4 1 4 3 1 1

GREEDY RANDOMIZED (10 shortest paths):
Best energy = 1235.0
No. of solutions = 17130
Av. quality of solutions = 1415.7
Best solution:
10 4 4 4 1 4 3 1 1

GREEDY RANDOMIZED (5 shortest paths):
Best energy = 1303.0
No. of solutions = 34716
Av. quality of solutions = 1512.1
Best solution:
2 1 1 1 2 3 5 1 1

2.1.5. Discussão

Usando o greedy randomized, é possível observar que ao usar menos caminhos, este gera mais soluções, visto que o algoritmo inicia colocando os fluxos numa ordem aleatória, e de seguida, para cada caminho de cada fluxo calcula a energy consumption, como existem menos caminhos, ele processa as soluções mais rápido.
Quanto à energia obtida, é possível observar que o greedy randomized com todos os percursos e só com os 10 percursos, obtém valores iguais, de 1235.0, isto faz sentido, porque para ambos os algoritmos, os percursos obtidos para a melhor solução, são iguais. Já quanto à média da qualidade das soluções elas são muito próximas, sendo a com 10 percursos, ligeiramente maior.
Correndo o algoritmo usando 5 percursos, foram obtidos os piores valores de energia tanto para a média como para o pior resultados, sendo este de 1303.0, mais uma vez, isto consegue ser explicado, observando que nas outras duas abordagens, por exemplo, o melhor percurso, para o primeiro flow era o 10, mas agora como este percurso já não está no subconjunto de percursos disponibilizados ao algoritmo é impossível obter esta solução, ou seja, podemos estar a tirar a melhor solução ao limitar o subconjunto.

2.1.6. Alínea c.

2.1.7. Resultados

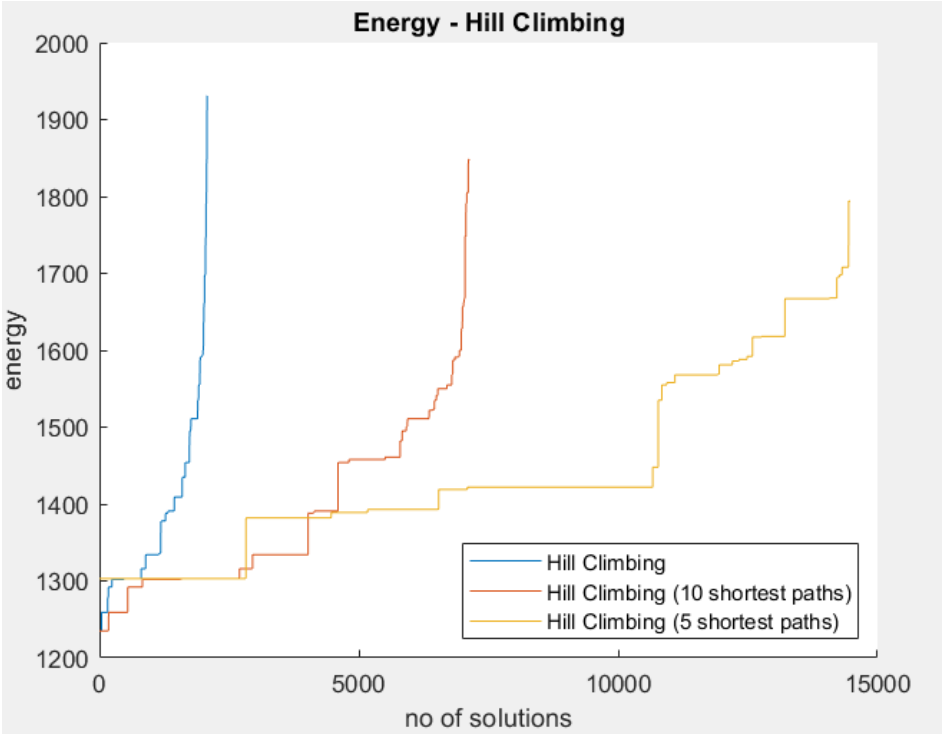

Gráfico 6: Resultados da energia total nas soluções obtidas, por ordem crescente, para a estratégia de hill climbing.

MULTI START HILL CLIMBING:
Best energy = 1235.0
No. of solutions = 2074
Av. quality of solutions = 1375.7
Best solution:
10 4 4 4 1 4 3 1 1

MULTI START HILL CLIMBING (10 shortest paths):
Best energy = 1235.0
No. of solutions = 7138
Av. quality of solutions = 1384.5
Best solution:
10 4 4 4 1 4 3 1 1

MULTI START HILL CLIMBING (5 shortest paths):
Best energy = 1303.0
No. of solutions = 14476
Av. quality of solutions = 1439.4
Best solution:
2 1 1 1 2 3 5 1 1

2.1.8. Discussão

Usando o algoritmo hill climbing é possível observar que mais uma vez à medida que diminuímos o número de percursos utilizados, neste caso, até 5, o número de soluções obtidas aumenta, visto que tal como no greedy randomized, o algoritmo para cada fluxo percorre cada percurso, se tivermos menos percursos para percorrer, vamos gastar menos tempo de processamento, assim conseguimos processar mais soluções.
Porém, com menos caminhos usados, obtivemos uma maior energia consumida, de 1303.0, do que nas outras abordagens, onde foi 1235.0 para ambas. Isto pode acontecer, porque ao diminuirmos os caminhos, vamos deixar de ter certas combinações de caminhos para percorrer e provavelmente estamos a excluir o caminho da solução com melhores resultados. Observando as melhores soluções obtidas para cada fluxo, usando os 10 caminhos mais curtos, no sétimo fluxo, o melhor percurso obtido foi o 10, usando apenas os 5 caminhos mais curtos, como estamos a limitar os percursos possíveis esse percurso já não pode ser escolhido.

2.1.9. Alínea d.

2.1.10. Discussão

Comparando todos os algoritmos, conseguimos observar que o melhor valor para a energia consumida obtida foi de 1235.0, sendo esta solução obtida em dois algoritmos, no greedy randomized e no hill climbing.
Comparando estes 2 algoritmos percebemos que a estratégia de hill climbing em média mais uma vez, encontra melhores valores, obtendo um menor número de soluções, uma vez que é mais elaborada e computacionalmente mais exigente. Se ambos os algoritmos chegam ao mesmo valor para o consumo de energia, neste caso, pode-se optar pela estratégia greedy, uma vez que é mais simples.
Quanto ao número de percursos usados, os melhores resultados obtêm-se usando pelo menos os 10 caminhos mais curtos, sendo que em média obtemos melhores resultados quando usamos todos os percursos. Os piores resultados foram obtidos usando apenas os 5 caminhos mais curtos.
Relativamente à estratégia random, ao contrário do que acontecia nas outras estratégias, os resultados melhoram progressivamente quando são usados menos caminhos, sendo o tempo de execução constante para qualquer conjunto de caminhos considerado. Esta estratégia apesar de conseguir encontrar mais soluções, no mesmo intervalo de tempo, relativamente aos restantes algoritmos, obteve piores resultados, uma vez que é computacionalmente mais simples e não usa qualquer critério para explorar as soluções.

3. Tarefa 3

3.1. Alínea a.

3.1.1. Resultados

Caminhos de maior disponibilidade obtidos:
Flow 1: [1 2 3]
Flow 2: [1 5 4]
Flow 3: [2 4 5 7]
Flow 4: [3 4]
Flow 5: [4 8 9]
Flow 6: [5 7 8 6]
Flow 7: [5 7 8]
Flow 8: [5 7 9]
Flow 9: [6 10]

3.2. Alínea b.

3.2.1. Resultados

Flow 1:
  Must available path: [1 2 3]
  Must available path availability: 99.6508%
  Disjoint path: [1 5 4 3]
  Disjoint path availability: 99.6380%

Flow 2:
  Must available path: [1 5 4]
  Must available path availability: 99.7897%
  Disjoint path: [1 2 4]
  Disjoint path availability: 99.7394%

Flow 3:
  Must available path: [2 4 5 7]
  Must available path availability: 99.7569%
  Disjoint path: [2 3 8 7]
  Disjoint path availability: 99.5472%

Flow 4:

Must available path: [3 4]
Must available path availability: 99.8480%
Disjoint path: [3 2 4]
Disjoint path availability: 99.7861%

Flow 5:
Must available path: [4 8 9]
Must available path availability: 99.7563%
Disjoint path: [4 5 7 9]
Disjoint path availability: 99.7168%

Flow 6:
Must available path: [5 7 8 6]
Must available path availability: 99.6853%
Disjoint path: [5 4 3 6]
Disjoint path availability: 99.6453%

Flow 7:
Must available path: [5 7 8]
Must available path availability: 99.7739%
Disjoint path: [5 4 8]
Disjoint path availability: 99.7417%

Flow 8:
Must available path: [5 7 9]
Must available path availability: 99.8498%
Disjoint path: [5 4 8 9]
Disjoint path availability: 99.6235%

Flow 9:
Must available path: [6 10]
Must available path availability: 99.9416%
Disjoint path: [6 8 9 10]
Disjoint path availability: 99.5896%

Average availability value among all flows of the service: 99.9993%

3.2.2. Discussão

Os pares de caminhos obtidos, para todos os fluxos, são disjuntos nas ligações, ou seja, não partilham nenhuma ligação entre nós. Para além disso, percebe-se que o valor de disponibilidade do caminho alternativo nunca é superior ao valor do caminho de maior disponibilidade. A média obtida para a disponibilidade do serviço foi de 99.9993%.

3.3. Alínea c.

3.3.1. Resultados

		Without protection using must available path for each flow		1+1 protection	
Origin node	**Destination node**	**throughput origin to destination (Gbps)**	**throughput destination to origin (Gbps)**	**throughput origin to destination(Gbps)**	**throughput destination to origin (Gbps)**
1	2	1.0	1.0	1.7	1.5
1	5	0.7	0.5	1.7	1.5
2	3	1.0	1.0	5.5	4.9
2	4	2.4	1.5	5.5	4.1
3	4	2.4	2.1	4.9	4.3
3	6	0	0	1.2	1.5
3	8	0	0	2.4	1.5
4	5	2.9	2.2	10.8	10.3
4	8	1.0	2.2	4.7	6.6
5	7	7.3	7.4	8.3	9.6
6	8	1.5	1.2	2.9	2.8
6	10	1.4	1.6	1.4	1.6
7	8	3.3	4.0	4.8	6.4
7	9	1.6	1.9	2.6	4.1
8	9	1.0	2.2	4.0	5.7
8	10	0	0	1.4	1.6

Tabela 1: Tabela throughput para cada uma das ligações, no caso de uma rede sem proteção que usa os caminhos mais disponíveis e uma rede que usa a proteção 1+1.

Total bandwidth required on all links without protection: 56.30 Gbps

Total bandwidth required on all links with 1+1 protection: 131.80 Gbps

3.3.2. Discussão

Para a discussão foi feita a comparação entre uma rede que não usa proteção usando os percursos de maior disponibilidade para cada fluxo, com uma rede que usa o mecanismo de proteção 1+1, ou seja, com 2 percursos por fluxo, onde o primeiro corresponde ao percurso de maior disponibilidade e o segundo o percurso de maior disponibilidade disjunto do primeiro.
Utilizando o mecanismo de proteção 1+1, onde cada fluxo é duplicado pelos 2 percursos, o throughput de cada nó aumenta significativamente, uma vez que vamos duplicar o throughput para a mesma rede. Neste caso, apenas a ligação entre o nó 6 e 10 apresenta valores de throughput iguais para ambas as redes, uma vez que nenhum dos segundos percursos utiliza esta ligação. Para ligação entre os nós 4 e 5 o throughput, para a rede de proteção 1+1, excedeu o limite da capacidade (10Gbps), verificando os percursos para cada um dos fluxos obtidos na alínea b. percebemos que a ligação 4-5/5-4 é usada por muitos dos fluxos de dados, provocando a sua sobrecarrega.
O throughput total utilizado em todas as ligações mais que duplicou. Estes resultados podem ser explicados pelo facto de que para além de duplicar o throughput na rede, os percursos de maior disponibilidade disjunto ao de maior disponibilidade usados no mecanismo de proteção 1+1, por vezes usam mais ligações quando comparados com os percursos de maior disponibilidade.

3.4. Alínea d.

3.4.1. Resultados

		Without protection using must available path for each flow		**1:1 protection**	
Origin node	**Destination node**	**throughput origin to destination (Gbps)**	**throughput destination to origin (Gbps)**	**throughput origin to destination(Gbps)**	**throughput destination to origin (Gbps)**
1	2	1.0	1.0	1.7	1.5
1	5	0.7	0.5	1.7	1.5
2	3	1.0	1.0	3.4	3.4
2	4	2.4	1.5	4.8	3.6
3	4	2.4	2.1	3.9	3.3
3	6	0	0	1.2	1.5

3	8	0	0	2.4	1.5
4	5	2.9	2.2	6.9	5.6
4	8	1.0	2.2	4.7	6.6
5	7	7.3	7.4	8.3	9.6
6	8	1.5	1.2	2.9	2.8
6	10	1.4	1.6	1.4	1.6
7	8	3.3	4.0	4.8	6.4
7	9	1.6	1.9	2.6	4.1
8	9	1.0	2.2	2.6	4.1
8	10	0	0	1.4	1.6

Tabela 2: Tabela throughput para cada uma das ligações, no caso de uma rede sem proteção que usa os caminhos mais disponíveis e uma rede que usa a proteção 1:1.

Total bandwidth required on all links without protection: 56.30 Gbps

Total bandwidth required on all links with 1:1 protection: 113.40 Gbps

Highest bandwidth value required among all links: 9.60 Gbps

3.4.2. Discussão

Para a discussão foi feita a comparação entre uma rede que não usa proteção usando os percursos de maior disponibilidade para cada fluxo, com uma rede que usa o mecanismo de proteção 1:1, ou seja, com 2 percursos por fluxo, onde o primeiro corresponde ao percurso de maior disponibilidade (designado por percurso de serviço) e o segundo o percurso de maior disponibilidade disjunto ao primeiro (designado por percurso de proteção).
Utilizando o mecanismo de proteção 1:1, onde cada fluxo é enviado por um dos percursos ou pelo percurso de serviço ou pelo percurso de proteção onde o último é só utilizado em caso de falha do primeiro, o throughput aumenta em relação à rede que não usa proteção, uma vez é necessário garantir em caso de falha de uma ligação que todos os fluxos continuam a funcionar.
Tal como aconteceu no mecanismo de proteção 1+1, apenas a ligação entre o nó 6 e 10 apresenta o mesmo throughput. A ligação com maior throughput foi a ligação entre o nó 5 e 7 (8.3 Gbps e 9.6 Gbps, da origem para o destino e do destino para a origem, respetivamente) .
O throughput total utilizado em todas as ligações duplicou. Este aumento advém da redundância necessária para garantir que todos os fluxos continuem a funcionar no caso de uma falha nunca de ligação.

3.5. Alínea e.

3.5.1. Resultados

		1+1 protection		1:1 protection	
Origin node	**Destination node**	**throughput origin to destination(Gbps)**	**throughput destination to origin (Gbps)**	**throughput origin to destination(Gbps)**	**throughput destination to origin (Gbps)**
1	2	1.7	1.5	1.7	1.5
1	5	1.7	1.5	1.7	1.5
2	3	5.5	4.9	3.4	3.4
2	4	5.5	4.1	4.8	3.6
3	4	4.9	4.3	3.9	3.3
3	6	1.2	1.5	1.2	1.5
3	8	2.4	1.5	2.4	1.5
4	5	10.8	10.3	6.9	5.6
4	8	4.7	6.6	4.7	6.6
5	7	8.3	9.6	8.3	9.6
6	8	2.9	2.8	2.9	2.8
6	10	1.4	1.6	1.4	1.6
7	8	4.8	6.4	4.8	6.4
7	9	2.6	4.1	2.6	4.1
8	9	4.0	5.7	2.6	4.1
8	10	1.4	1.6	1.4	1.6

Tabela 3: Tabela throughput para cada uma das ligações, no caso de uma rede que usa a proteção 1:1 e uma rede que usa 1+1.

Total bandwidth required on all links with 1+1 protection: 131.80 Gbps

Total bandwidth required on all links with 1:1 protection: 113.40 Gbps

3.5.2. Discussão

Comparando os resultados obtidos para ambas as proteções 1+1 e 1:1, é possível perceber que todos os valores de throughput das ligações no mecanismo de proteção 1:1 são iguais ou inferiores aos valores obtidos das ligações do mecanismo de proteção 1+1.
O valor de throughput total utilizado em todas as ligações é maior na proteção 1+1, em relação a ligação 1:1. Estes resultados são consequência da maneira de como cada uma das proteção opera, enquanto a proteção 1+1 duplica o tráfego de cada fluxo pelos 2 percursos, a proteção 1:1 apenas usa o segundo percurso no caso do primeiro falhar, permitindo que os recurso do percurso de proteção sejam partilhados entre os diferentes fluxos, por este motivo a proteção 1:1 usa menos recursos que a ligação 1+1. Em contrapartida, enquanto que para a proteção 1+1 o tempo de recuperação a falhar é praticamente nulo, no caso da proteção 1:1 este tempo pode ser significativo, uma vez que o nó de origem tem que ser notificado para passar a transmitir para o percurso de proteção.

4. Tarefa 4

4.1. Alínea a.

4.1.1. Resultados

Fluxo 4:

1º most available path: [3 4]
Disjoint path: [3 2 4]

2º most available path: [3 2 4]
Disjoint path: [3 4]

3º most available path: [3 8 4]
Disjoint path: [3 4]

4º most available path: [3 6 8 4]
Disjoint path: [3 4]

5º most available path: [3 8 7 5 4]
Disjoint path: [3 4]

6º most available path: [3 6 8 7 5 4]
Disjoint path: [3 4]

7º most available path: [3 8 9 7 5 4]

Disjoint path: [3 4]

8º most available path: [3 2 1 5 4]
Disjoint path: [3 4]

9º most available path: [3 6 8 9 7 5 4]
Disjoint path: [3 4]

10º most available path: [3 6 10 9 8 4]
Disjoint path: [3 4]

Nota: Aqui mostramos só o exemplo do fluxo 4, para ver os restantes fluxos basta executar o script da Task 4 que imprime os 10 caminhos mais disponíveis e respectivos caminhos disjuntos.

4.1.2. Discussão

Foram obtidos os 10 pares de caminhos disjuntos, para cada fluxo, calculando primeiro os 10 caminhos mais disponíveis para o respetivo fluxo e de seguida o caminho mais disponível disjunto a cada um dos 10 caminhos anteriores.
No caso do quarto fluxo houve pouca variação dos percursos disjuntos obtidos, na verdade para os 10 caminhos mais disponíveis apenas obtivemos 2 caminhos disjuntos distintos, [3 2 4] e [3 4], que correspondem ao 1º e 2º percurso mais disponíveis, respetivamente.
Esta falta de diversidade nos caminhos disjuntos obtidos acontece também para os restantes fluxos, uma vez que estamos a tentar encontrar o caminho mais disponível disjunto ao k-ésimo mais disponível é normal que muitas das vezes vamos obter os caminhos de melhor disponibilidade, desde que estes não partilhem nenhum link.

4.2. Alínea b.

4.2.1. Resultados

Greedy randomized:

Highest required bandwidth= 9.60 Gbps
No. of solutions = 630
Av. quality of solutions = 9.60 Gbps
Average availability value among all flows of the service = 99.9993%

Best solution

Flow 1 pair:
[1 2 3]
1º path availability: 99.65085%

[1 5 4 3]
2º path availability: 99.63803%

Flow 2 pair:
[1 5 4]
1º path availability: 99.78969%
[1 2 4]
2º path availability: 99.73937%

Flow 3 pair:
[2 4 5 7]
1º path availability: 99.75688%
[2 3 8 7]
2º path availability: 99.54719%

Flow 4 pair:
[3 4]
1º path availability: 99.84802%
[3 2 4]
2º path availability: 99.78606%

Flow 5 pair:
[4 8 9]
1º path availability: 99.75631%
[4 5 7 9]
2º path availability: 99.71684%

Flow 6 pair:
[5 7 8 6]
1º path availability: 99.68533%
[5 4 3 6]
2º path availability: 99.64530%

Flow 7 pair:
[5 7 8]
1º path availability: 99.77394%
[5 4 8]
2º path availability: 99.74175%

Flow 8 pair:
[5 7 9]
1º path availability: 99.84980%
[5 4 8 9]
2º path availability: 99.62348%

Flow 9 pair:
[6 10]
1º path availability: 99.94159%

[6 8 9 10]
2º path availability: 99.58959%

Multi Start Hill Climbing:

Highest required bandwidth= 9.60 Gbps
No. of solutions = 213
Av. quality of solutions = 9.60 Gbps
Average availability value among all flows of the service = 99.9993%

Best solution
Same as Greedy Randomized

Random:
Highest required bandwidth= 9.60 Gbps
No. of solutions = 30118
Av. quality of solutions = 9.82 Gbps
Average availability value among all flows of the service: 99.9990%

Best solution

Flow 1 pair:
[1 5 4 2 3]
1º path availability: 99.57619%
[1 2 3]
2º path availability: 99.65085%

Flow 2 pair:
[1 5 7 9 8 4]
1º path availability: 99.52951%
[1 2 4]
2º path availability: 99.73937%

Flow 3 pair:
[2 4 8 9 7]
1º path availability: 99.59132%
[2 1 5 7]
2º path availability: 99.67747%

Flow 4 pair:
[3 8 9 7 5 4]
1º path availability: 99.47565%
[3 4]
2º path availability: 99.84802%

Flow 5 pair:
[4 2 1 5 7 9]
1º path availability: 99.51261%
[4 8 9]
2º path availability: 99.75631%

Flow 6 pair:
[5 1 2 3 6]
1º path availability: 99.50419%
[5 7 8 6]
2º path availability: 99.68533%

Flow 7 pair:
[5 7 8]
1º path availability: 99.77394%
[5 4 8]
2º path availability: 99.74175%

Flow 8 pair:
[5 1 2 4 8 9]
1º path availability: 99.41944%
[5 7 9]
2º path availability: 99.84980%

Flow 9 pair:
[6 8 9 10]
1º path availability: 99.58959%
[6 10]
2º path availability: 99.94159%

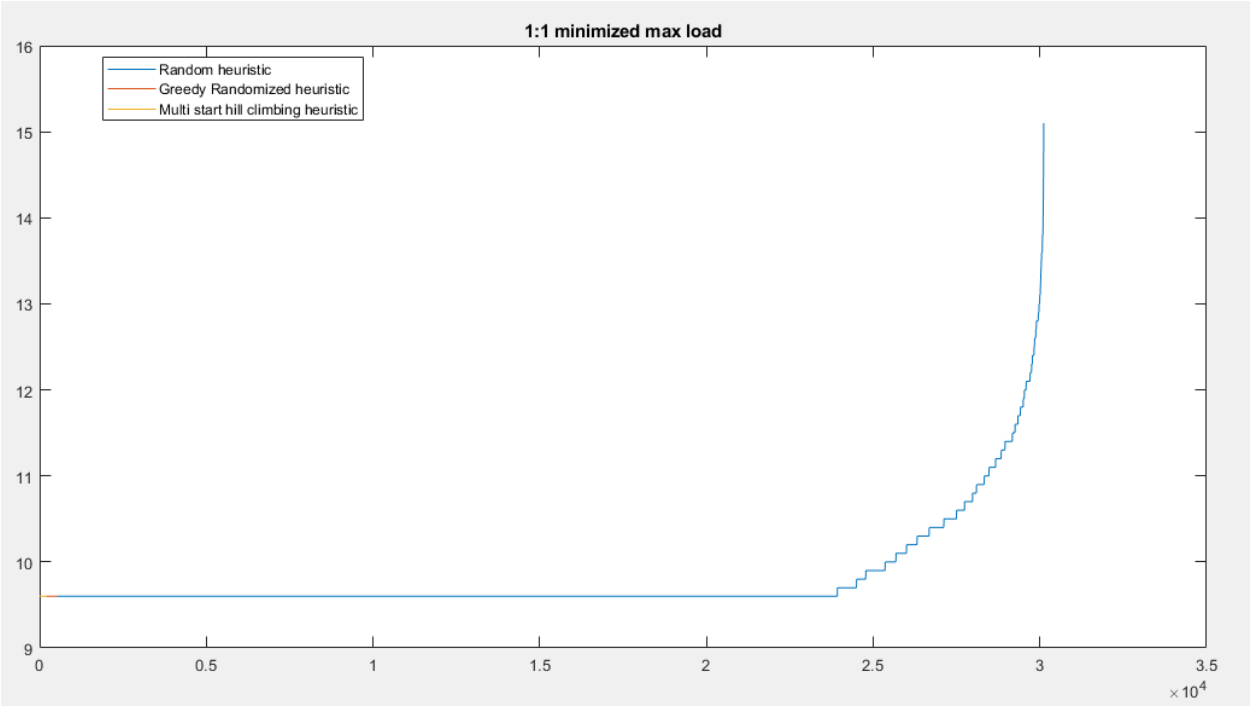

Gráfico 7: Gráfico dos resultados da carga máxima,entre todas as ligações, obtidos para a estratégia random, greedy randomized e multi start hill climbing

4.2.2. Discussão

Analisando os resultados percebemos que, usando um algoritmo de hill climbing que tenta otimizar a maior largura de banda necessária entre todos os links, a solução obtida corresponde exatamente aos pares obtidos na alínea 3.b.. Verifica-se também que todas as 213 soluções encontradas pelo algoritmo são exatamente as mesmas. Analisando os resultados obtidos para o caso do algoritmos greedy randomized, para o mesmo problema de otimização, todas as soluções obtidas correspondem aos mesmos resultados obtidos com a estratégia de hill climbing. O que acontece é que a heurística greedy randomized determina uma solução onde a largura de banda máxima necessária é de 9.60 Gbps, que corresponde aos links de maior disponibilidade juntamente com os de maior disponibilidade disjuntos dos primeiros. Como esta solução não pode ser melhorada a algoritmo de hill climbing não consegue encontrar um vizinho melhor. Tendo em conta que a disponibilidade dos fluxos é determinada a partir da disponibilidade de cada um dos links e que essa disponibilidade, no geral, é tanto melhor quanto menor for o número de links que suportam o fluxo, é expectável que os pares obtidos na alínea 3.b. (de maior disponibilidade) correspondam também à solução onde a largura de banda máxima entre todos os links seja menor, uma vez que tenta minimizar a largura de banda máxima entre todos os links e implicitamente o número de links utilizados para cada um dos caminhos.
Para além disso, tal como discutido na alínea 4.a., a falta de diversidade entre os 10 pares de caminhos disjuntos, para cada fluxo, é outro fator que leva a que os resultados não variem.
Para tentar verificar alguma eventual falha na implementação foi usado o algoritmo random, para o mesmo problema de otimização, e aqui já houve diversidade nos resultados. No

entanto, mais uma vez a melhor solução foi de 9.60 Gbps no link mais sobrecarregado, que confirma, de certa forma, aquilo que foi discutido.
Comparando os resultados com os obtidos na alínea b do da tarefa 3, verifica-se que estes são equivalentes.

5. Explicação da implementação

5.1. Tarefa 1

Foram usadas as versões dadas pelo professor. Foi acrescentada a variável “worst load”, para observar o pior load obtido pelo algoritmos e um print para observação dos percursos obtidos na melhor solução dos fluxos.

5.2. Tarefa 2

Foram usadas as versões dadas pelo professor. Foi acrescentado um print para observação dos percursos obtidos na melhor solução dos fluxos.

5.3. Tarefa 3

5.3.1. Alínea a.

Foi usada a função “calculatePaths” para calcular os caminhos mais disponíveis, para cada link, passando a matriz com os valores de MTBF, calculados com a expressão de MTBF usada nas aulas práticas, após aplicar a transformação logarítmica.

5.3.2. Alínea b.

Foram calculados os caminhos mais disponíveis como na alínea a. Para cada fluxo, foi determinada a disponibilidade dos caminhos mais disponíveis acedendo à matriz, “A”, calculada na alínea a., com a disponibilidade de cada link . A disponibilidade do fluxo corresponde ao produtório das disponibilidade de cada um dos links. De seguida foi criada uma matriz temporária tmpAlog para armazenar a transformação logarítmica do MTBF de todos os links, com exceção dos links pertencentes ao fluxo em questão. Com esta nova variável foi calculada de novo a disponibilidade de cada fluxo como foi feito para os caminhos mais disponíveis.
No final, os valores de disponibilidade de cada um dos fluxos ficaram armazenados nas variáveis pathA1 e pathA2, para os caminhos de maior disponibilidade e para os caminhos disjuntos, respectivamente. Os caminhos mais disponíveis de cada fluxo foram armazenados na variável sP e os caminhos disjuntos na variável disjointPaths.
A disponibilidade média entre todos os fluxos foi determinada através da média da disponibilidade dos caminhos em paralelo (disjuntos) de cada fluxo, usando a expressão das aulas teóricas.

5.3.3. Alínea c.

Em primeiro lugar foi calculada a matriz com a carga de cada um dos links usando os caminhos mais disponíveis, armazenados na variável “sP”, através da função calculateLinkLoads. Em segundo lugar, fez-se o mesmo para os caminhos disjuntos, armazenadas na variável “disjointPaths”. Finalmente a carga total de cada um dos links é dada pela soma das duas matrizes calculadas, o resultado é armazenado na matriz “Loads1Plus1”. No final, a matriz “Loads1Plus1” é percorrida para procurar links que excedem a carga máxima de 10 Gbps. O total de largura de banda é determinado pela soma da carga em todos os links.

5.3.4. Alínea d.

A carga de cada link foi calculada usando a função “calculateLoads1To1” (ver explicação em 5.5.1. calculateLoads1To1) sendo o resultado armazenado na matriz “Loads1To1”. No final, a matriz “Loads1To1” é percorrida para procurar links que excedem a carga máxima de 10 Gbps. O total de largura de banda é determinado pela soma da carga em todos os links

5.4. Tarefa 4

5.4.1. Alínea a.

Primeiramente, foram calculados 10 caminhos de maior disponibilidade para cada fluxo utilizando a função “calculatePaths”. De seguida, foi calculado o caminho mais disponível, disjunto ao k-ésimo caminho mais disponível, para cada um dos 10 caminhos de cada fluxo. Para isso, foi calculado, para cada k-ésimo caminho de cada fluxo, o caminho mais disponível disjunto deste num processo equivalente ao descrito para a Tarefa 3 alínea b. (ver 5.3.1).

5.4.2. Alínea b.

Para encontrar os pares que permitem minimizar a maior carga entre todos os links foi usada uma estratégia de hill climbing que parte de uma solução gerada pelo algoritmo greedy randomized, a sua implementação é equivalente ao código das aulas práticas. A única diferença reside na função de avaliação da solução, em vez de ser chamada a função “calculateLoads” é chamada a função “calculateLoads1To1” (ver 5.5.1). O algoritmo é corrido durante 30 segundos e no final os resultados são imprimidos pela função “printResults”, esta função é também responsável por calcular a disponibilidade de cada um dos caminhos do par disjunto, para todos os fluxos, e disponibilidade média entre todos os fluxos de forma equivalente ao descrito para a Tarefa 3 alínea b. (ver 5.5.2). Também foi feito o plot da maior carga entre todos os links para todas as soluções encontradas.

De forma a complementar a análise foram também implementados dois algoritmos adicionais, um que usa a estratégia random e outro a estratégia greedy randomized, ambos tentam resolver o mesmo problema de otimização e usam a mesma função de avaliação. Tal como no algoritmo de hill climbing, os resultados são imprimidos e os plots desenhados.

5.5. Funções desenvolvidas

5.5.1. calculateLoads1To1

A função calculateLoads1To1 calcula a carga total de cada um dos link da rede, para um configuração de proteção 1:1. Aos pares dos fluxos são passados como argumento, “sP” contém os k caminhos mais disponíveis de cada fluxo, “disjointPaths” contém os k caminhos mais disponíveis disjuntos aos primeiros de cada fluxo. O argumento “nNodes” indica quantos nós existem na rede, “Links” é a matriz que descreve as ligações da rede, “T” é a matriz que descreve o throughput de cada fluxo e “sol” indica qual par estamos a usar para cada fluxo.
Em primeiro lugar é calculada a carga de cada link usando o primeiro caminho para cada fluxo, essa matriz fica armazenada na variável “Loads1To1”. De seguida, é determinada a carga em cada ligação no caso de uma falha única de ligação, para todos os casos possíveis, ou seja, para a falha de cada um das ligações. Para isso, para uma ligação particular verificamos, para todos os fluxos, se a ligação em causa pertence ou não ao primeiro percurso do par, se sim escolhemos o segundo em detrimento do primeiro. Os percursos escolhidos para cada um dos fluxos são armazenados numa variável auxiliar “auxSP”. No final, é calculada a carga de cada ligação para os percursos resultante da falha em causa, essas cargas são comparadas com as cargas da matriz “Loads1To1” e a matriz “Loads1To1” passa a ter, para cada ligação, o valor máximo de carga comparado. Este processo repete-se para todas as ligações e no final é devolvida a matriz “Loads1To1” com as cargas de cada ligação necessária para suportar o mecanismo de proteção 1:1.

5.5.2. Multi start hill climbing first neighbor

Foi desenvolvido também um código que usa a variante first neighbor do algoritmo de hill climbing. O código não foi usado na implementação, uma vez que houve necessidade para tal. A estrutura é semelhante ao hill climbing normal, a única diferença é a verificação de paragem assim que é encontrado um vizinho melhor, através da flag “betterNeighbourFlag”. Quando essa verificação acontece o algoritmo termina e é determinada a solução. Como exemplo é usado o problema de utilização da Tarefa 4, juntamente com o algoritmo greedy randomized.

```
t= tic;
bestLoad= inf;
sol= zeros(1,nFlows);
allValues= [];
while toc(t)<time
```

```
%Optimization algorithm resorting to the greedy randomized strategy:
sol= zeros(1,nFlows);
for i = randperm(nFlows)
   k_best = 0;
   best = inf;
   for k = 1:nSP(i) %pair each pair calculate Loads1To1
      sol(i) = k;
      Loads= calculateLoads1To1(nNodes,Links,T,sP,disjointPaths,sol);
      load= max(max(Loads(:,3:4)));
      if load < best
        k_best = k;
        best = load;
      end
   end
   sol(i) = k_best;
end
load = best;

%HILL CLIMBING:
improved = true;
while improved
   flow_best= 0;      %fluxo
   path_best= 0;      %percurso do fluxo
   best= load;

   betterNeighbourFlag = false;
   flow = 1;
   while ~betterNeighbourFlag && flow < nFlows
      nPaths = nSP(flow);
      path = 1;
      while ~betterNeighbourFlag && path < nPaths
         if path~=sol(flow)
            aux= sol(flow);
            sol(flow)= path;
            Loads= calculateLoads1To1(nNodes,Links,T,sP,disjointPaths, sol);
            load1= max(max(Loads(:,3:4)));
            if load1<best
               flow_best= flow;
               path_best= path;
               best= load1;
               betterNeighbourFlag = true;
            end
            sol(flow)= aux;
         end
         path = path + 1;
      end
      flow = flow + 1;
   end
   if flow_best>0
      sol(flow_best)= path_best;
      load= best;
   else
      improved= false;
   end
```

```
    end
    allValues= [allValues load];
    if load<bestLoad
        bestSol= sol;
        bestLoad= load;
    end
end
```

Código 1: Exemplo da implementação da variante first neighbor do algoritmo multi start hill climbing