Universidade do Minho
Escola de Engenharia

Inteligência Artificial

Trabalho Prático

Grupo 06

Ana Oliveira a104536
André Campos a104618
Beatriz Peixoto a104170
Sara Azevedo Lopes a104179
Tomás Pinto a104448

Janeiro, 2025

1. Avaliação por pares

A104536 Sofia = 0
A104618 André = 0
A104170 Beatriz = 0
A104179 Sara = 0
A104448 Tomás = 0

Índice

Índice de figuras

2. Apresentação do Caso de Estudo

Este projeto foi desenvolvido no âmbito da unidade curricular de Inteligência Artificial, do curso de Licenciatura em Engenharia Informática.

Neste projeto, o objetivo é otimizar a distribuição de recursos por zonas afetadas por uma catástrofe natural, representadas sob a forma de um grafo, tendo em conta fatores limitantes como falta de recursos, janelas de tempo de acessibilidade às zonas, condições meteorológicas e limitações dos próprios veículos disponibilizados para percorrer as rotas.

Deste modo, este projeto requer a aplicação do conhecimento transmitido ao longo deste semestre sobre os diferentes algoritmos de procura abordados.

3. Formulação do problema

Neste projeto, desenvolvemos 3 modos:

- “Ambiente Estático”, em que não há qualquer alteração no grafo que não seja feita pelos algoritmos.
- “Ambiente Dinâmico Controlado”, em que o utilizador decide quando vão ocorrer mudanças, assim como a sua quantidade, sendo estas aleatórias.
- “Ambiente Dinâmico”, no qual, a cada 20 segundos, é executada uma mudança aleatória num nó ou rota aleatórios, podendo tratar-se de uma mudança de meteorologia, bloqueio ou desbloqueio de uma rota ou produção de medicamentos num nó aleatório.

Tipo:

Versão “Ambiente Estático” e “Ambiente Dinâmico Controlado”:

- Problema de procura.
- Estático: o mundo não se altera durante a deliberação (atuar).
- Discreto: existe um número fixo e finito de ações e perceções possíveis.
- Determinístico: qualquer ação tem um único efeito garantido.
- Acessível: é possível obter informações completas e atualizadas sobre o ambiente.

Versão “Ambiente Dinâmico”:

- Problema de procura.
- Dinâmico: o ambiente pode alterar-se durante a deliberação do agente.
- Discreto: existe um número fixo e finito de ações e perceções possíveis.
- Determinístico: qualquer ação tem um único efeito garantido.
- Acessível: é possível obter informações completas e atualizadas sobre o ambiente.

Representação: Lista de Adjacência que representa um grafo armazenando, para cada nó, uma lista de pares que indicam os nós vizinhos, juntamente com a informação relevante sobre as arestas e nós. Detalhes sobre a informação do nó são aprofundados no capítulo “2.1 Características de nó, aresta e veículo” e a representação destes é aprofundada no capítulo “Interface”

Nó Inicial: O nó inicial é escolhido pelo utilizador na interface em todos os algoritmos, exceto nos algoritmos de Hill-Climbing e Arrefecimento Simulado, nos quais o nó inicial é escolhido aleatoriamente. Mais informações sobre este nó estão presentes no capítulo 3.2.

Teste objetivo: Nó que é mais urgente socorrer. Mais informações sobre este nó estão presentes no capítulo 3.2.

Operadores: Transição entre nós. Capacidade de recuar em alguns algoritmos, como A*.

Custo da solução: Este dá-se pela seguinte fórmula:

Custo = custo_total_arestas * (custo_veiculo / pessoas_socorridas), sendo que custo_total_arestas é a soma dos Peso das arestas do caminho e pessoas_socorridas é o número de pessoas que se socorrer no caminho. Tanto o Peso como custo_veiculo são aprofundados no capítulo 3.1. e pessoas_socorridas é o mínimo entre número de medicamentos na origem, o limite da carga do veículo e a população no nó de origem.

4. Descrição de componentes e as decisões tomadas

4.1. Características de nó, aresta e veículo

Um **nó** representa uma zona/área. Este possui os seguintes atributos:

- Nome: funciona como identificador.
- População: número de pessoas que necessitam de ser socorridas.
- Janela de tempo: período de tempo, em horas, no qual é admissível socorrer um nó. Caso o valor deste seja 0, não é possível socorrer a população. O máximo que este pode ter é 24.
- X: coordenada cartesiana horizontal que localiza o nó no grafo.
- Y: coordenada cartesiana vertical que localiza o nó no grafo.
- Medicamentos: número de medicamentos disponíveis na zona.
- Meteorologia: divide-se em:
    - chuva.
    - tempestade.
    - vento.
    - nevoeiro.

Estes 4 campos representam as condições meteorológicas adversas que uma área enfrenta. Podem ser preenchidos de 0-20, sendo 20 a situação mais grave e 0 indica que a zona não é afetada por essa condição.

- Veículos: lista de veículos que uma zona pode enviar a outra (exemplo: carro).

Uma **aresta** representa um caminho entre dois nós. Esta possui os seguintes atributos:

- Nó de origem: um dos nós da ligação
- Nó de destino: outro nó da ligação
- Peso: distância real entre duas áreas em quilómetros.
- Bloqueada: indica se um caminho está livre ou bloqueado.
- Permitidos: lista de veículos que podem percorrer o caminho.

Um **veículo** representa o que irá carregar os medicamentos entre zonas. Estes possuem:

- Tipo: nome do veículo.
- Custo: valor do custo de utilizar determinado veículo. Por exemplo, o custo de usar um carro é inferior ao do uso de um avião.
- Limite de carga: número máximo de medicamentos que podem carregar entre zonas.
- Combustível disponível: número de quilómetros que podem percorrer.
- Velocidade: velocidade do veículo em quilómetros por hora.

Os nós, arestas e características de um veículo são carregados para memória de ficheiros json. Exemplos de cada um:

Nó - {"nome": "A", "populacao": 0, "tempo": 15, "x": 3, "y": 3, "medicamento": 1200, "meteorologia": {"chuva": 1, "tempestade": 0, "vento": 3, "nevoeiro": 2}, "veiculos": ["barco", "carro"]}

Aresta - {"origem": "A", "destino": "B", "peso": 6, "bloqueada": false, "permitidos": ["barco", "carro"]},

Veículo - "aviao": {"custo": 5, "limite_carga": 1500, "combustivel_disponivel": 500, "velocidade": 200}

4.2. Nó Origem/Destino

O nó de origem é escolhido pelo utilizador através da interface. Um nó sem medicamentos não pode ser nó de origem, pois não pode socorrer outros. No caso do utilizador escolher um nó sem medicamentos como nó origem, a interface avisará que tal não é possível.

O nó destino é aquele que é considerado mais prioritário de socorrer. Para calcular a prioridade de cada nó, temos em consideração a janela de tempo em que ele necessita de ser socorrido, a população que precisa de ajuda e as condições meteorológicas que afetam o lugar. A fórmula para determinar a prioridade de um nó é a seguinte:

prioridade = janela_tempo / (populacao + impacto_meteorologico), com impacto_meteorologico = chuva + tempestade + vento + nevoeiro

O nó com menor valor de prioridade é o que deverá ser considerado como nó destino. Nós com valor 0 para a população por assistir e para janela de tempo, não são assistidos, tendo o valor de prioridade máximo (infinito).

4.3 Heurística e Custo Acumulado

A **heurística** de um nó obtém-se multiplicando a distância euclidiana entre este e o nó destino pela sua prioridade.

Se um nó não possuir pessoas para socorrer, a sua heurística será 5. Em geral, este valor é superior ao dos nós que têm pessoas para socorrer. Isto, pois prioriza-se travessias com nós com pessoas para socorrer, além do destino.

O **custo acumulado** consiste na soma dos campos Peso de todas as arestas, ou seja, a distância real entre dois nós.

4.4. Regras relacionadas com o funcionamento dos algoritmos

1. Se um caminho ficar bloqueado, este não pode ser utilizado.
2. Sempre que um nó é completamente socorrido ficando a população 0, a janela de tempo fica com valor 24, que é o valor máximo da mesma.
3. Deve ser socorrido o nó com maior prioridade, mas se o veículo tiver limite de carga superior à população que precisa de ser socorrida, deve-se socorrer os demais nós do caminho de melhor custo, por ordem de prioridade.
4. Nós com janela de tempo 0 e população superior a 0, não devem ser socorridos, pois esgotou-se o tempo para o qual se podia socorrer o nó.
5. Se o nó de origem, escolhido pelo utilizador, não tiver medicamentos, será notificado o utilizador de que o nó desejado não pode ser o nó de origem.

6. Sempre que um nó tiver medicamentos e população que necessita de ser socorrida em simultâneo, deve usar os seus próprios medicamentos para socorrer a sua própria população.

4.5. Interface

Interface genérica:

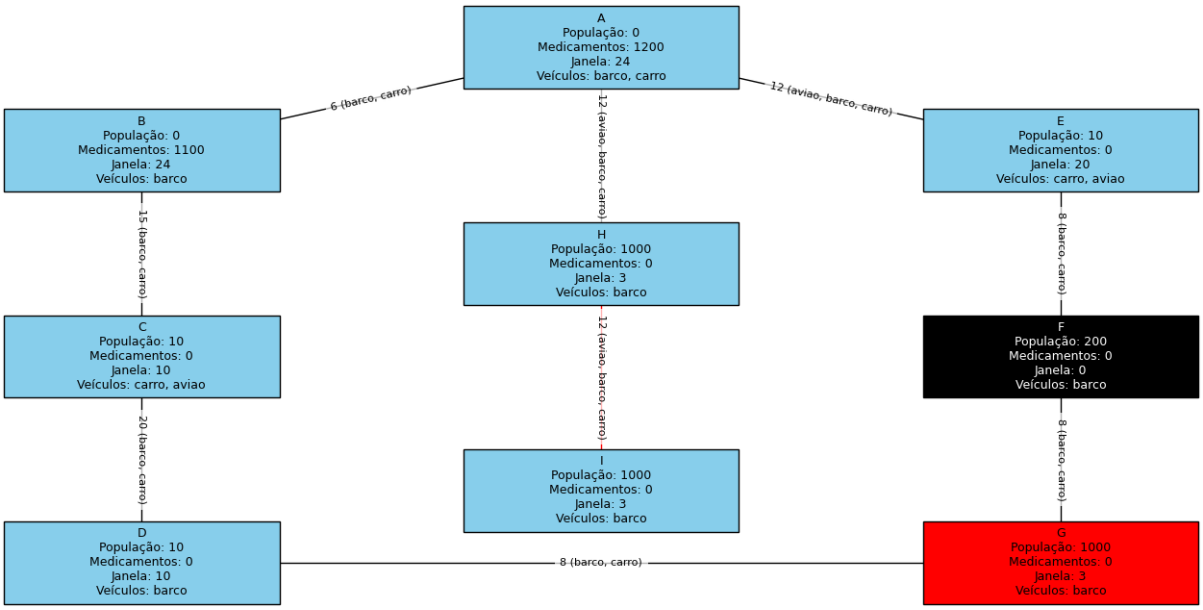

Figura 1 – Grafo da Interface Genérica

Cada **retângulo** representa um **nó**, apresentando as características dos nós mais relevantes, descritas anteriormente, como Nome, Medicamentos, entre outros.

Um **nó** de cor **preta** significa que o nó não deve ser socorrido (esgotou-se o tempo).

Um **nó** de cor **vermelha** significa o nó que é mais urgente socorrer, sendo este o destino.

Um **nó** de cor **azul** é um nó normal. Caso este tenha medicamentos, o utilizador pode selecioná-lo como nó de origem.

Cada **linha** representa uma **aresta**, que liga 2 nós, apresentando algumas das suas características como o Peso e os Veículos que a podem percorrer.

Uma **aresta** de cor **preta** significa um caminho livre que pode ser usado.

Uma **aresta** de cor **vermelha** significa um caminho bloqueado, que não deve ser usado pelos algoritmos.

Caso não seja necessário socorrer nenhum nó, ou seja, se todos tiverem o campo “Populacao” a 0, todos terão a cor **amarela**.

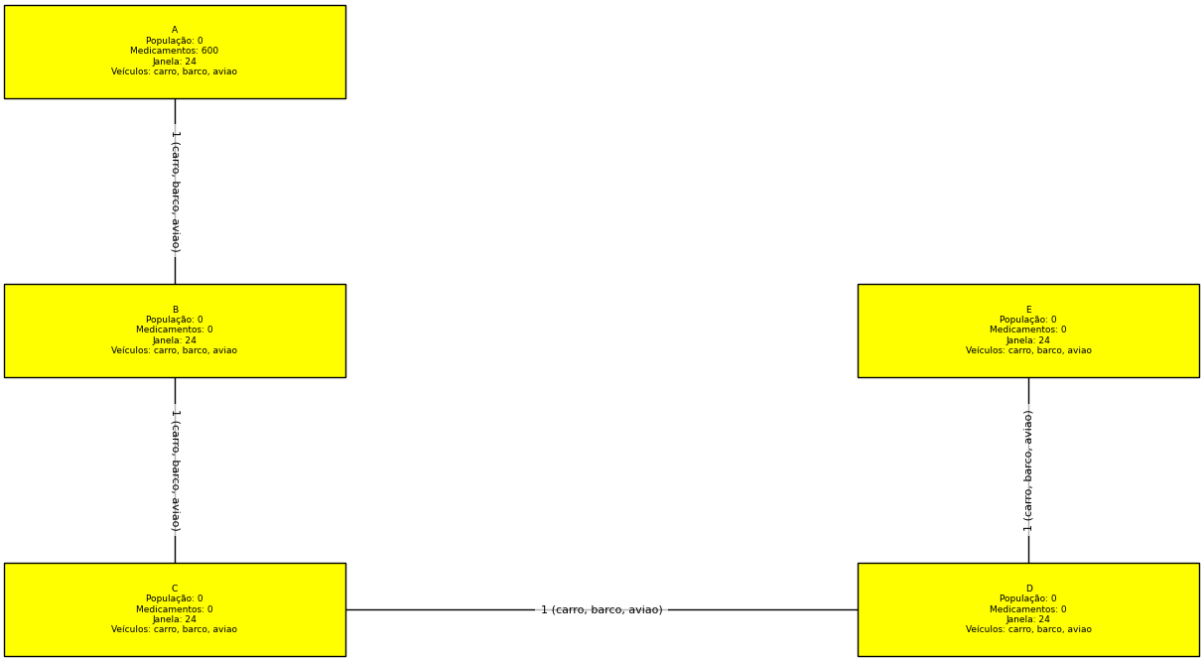

Figura 2 - Grafo com campo "Populacao" a 0

As heurísticas de cada nó também são apresentadas na interface, no canto inferior esquerdo.

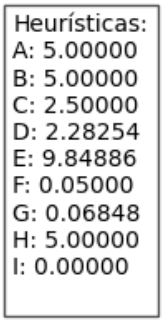

Figura 3 - Heurísticas

Adicionalmente, a prioridade de cada nó também é apresentada na interface, no canto superior esquerdo.

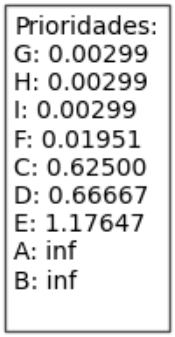

Figura 4 - Prioridades

5. Testes realizados e resultados obtidos

Nos nós visitados que vamos apresentar mais à frente, só iremos apresentar para o caminho de menor custo entre veículos, ou seja, o algoritmo refaz a pesquisa N vezes, sendo N o número de veículos. Por outro lado, o tempo de execução é o tempo total que inclui as tentativas para todos os veículos.

Para nó de origem, utilizamos sempre o nó A. Poderia-se usar outros nós.

Além disso, assumimos as seguintes características dos veículos, excepto para o último teste:

"carro": {"custo": 1, "limite_carga": 500, "combustivel_disponivel": 100, "velocidade": 50},
"barco": {"custo": 3, "limite_carga": 1000, "combustivel_disponivel": 60, "velocidade": 70},
"aviao": {"custo": 5, "limite_carga": 1500, "combustivel_disponivel": 500, "velocidade": 200}

No algoritmo Simulated Annealing, a temperatura inicial é um parâmetro importante para o desempenho do algoritmo. Se a temperatura inicial fosse muito baixa, a exploração seria muito reduzida, o que levaria a que o algoritmo aceitasse soluções piores com menor frequência. Com uma temperatura inicial muito alta, o comportamento levaria a uma exploração excessiva, o que o faria aceitar soluções piores com mais facilidade. Assim, no nosso problema, assumimos a temperatura inicial igual a 10 por ser uma temperatura moderada, ou seja, o algoritmo aceita soluções piores de forma controlada. Além disso, tendo em atenção ao problema que estamos a analisar e ao tamanho do espaço de soluções, escolhemos o número de iterações igual a 10, para garantir que o algoritmo explore corretamente as soluções vizinhas.

De forma semelhante, o bom funcionamento do algoritmo Hill-Climbing também depende de determinados parâmetros. Acima de tudo, tanto no algoritmo Hill-Climbing como o Simulated Annealing, é necessário ter “sorte” na escolha do nó inicial pois este é escolhido aleatoriamente e o Hill-Climbing corre o risco de ficar preso em mínimos ou máximos locais. Além disso, no Hill-Climbing temos também um número máximo de iterações e um número máximo de restarts. Assim como no Simulated Annealing, foi escolhido um número de iterações igual a 10 no algoritmo Hill-Climbing. Quanto ao número de restarts, este define o máximo de restarts que o algoritmo pode fazer, isto é, o número máximo de vezes que se pode voltar a executar o algoritmo, começando num novo nó inicial. Decidimos colocar esse número máximo de iterações igual a 6 visto que, para situações de teste, nos é conveniente que o algoritmo consiga escolher um bom nó inicial.

1. Grafo linear sem restrições

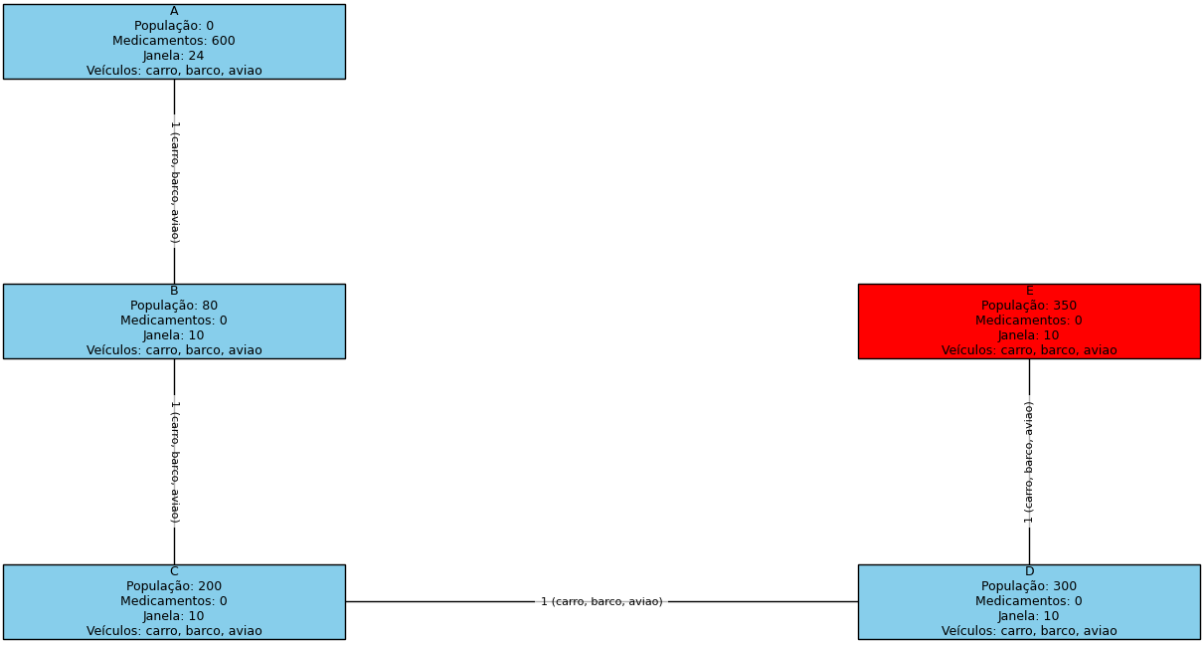

Figura 5 - Grafo linear sem restrições

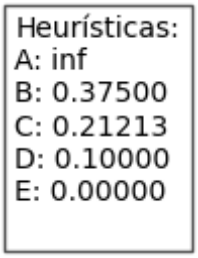

Figura 6 - Heuristicas do Grafo linear sem restrições

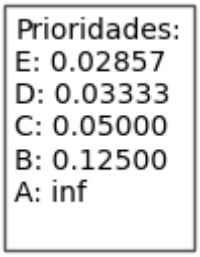

Figura 7 - Prioridades do Grafo linear sem restrições

Um **grafo linear sem restrições**, como o apresentado, é extremamente útil para os testes devido às suas características estruturais e simplificadas. A linearidade do grafo ajuda a identificar discrepâncias em estratégias de exploração e cálculo de custos em situações previsíveis e controladas, uma vez que há apenas um caminho possível entre os nós. Isto permite avaliar com precisão o comportamento dos algoritmos de procura, garantindo que seguem a ordem esperada e produzem os resultados correctos. Além disso, a ausência de bloqueios, todos os veículos estarem autorizados a usar todas as arestas, entre outros, eliminam restrições externas, permitindo uma análise focada exclusivamente nos algoritmos. Este grafo também é útil para validar heurísticas, já que a estrutura linear simplifica os

cálculos, permitindo verificar se estão a ser corretamente aplicadas. Assim, o grafo linear não só serve como ponto de partida, mas também como uma ferramenta essencial para testar a consistência e eficiência dos algoritmos em condições favoráveis e previsíveis. Além disso, o nó A é o único nó com medicamentos e os demais possuem população que necessita de ser socorrida.

grafo_linear_sem_restricoes	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.011429	0.011429	0.011429	0.011429	0.011429	0.011429	0.011429	0.011429
Veículo escolhido	carro	carro	carro	carro	carro	carro	carro	carro
Tempo de execução (ms)	0.599	1.442	2.265	0.408	1.104	3.764	1.47	380
Nós solução	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	[‘A’, ‘B’. ‘C’, ‘D’, ‘E’]	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	[‘A’, ‘B’, ‘C’, ‘D’, ‘E’]	['A', 'B', 'C', 'D', 'E']
Nós visitados	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	[‘A’, ‘B’. ‘C’, ‘D’, ‘E’]	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	['A', 'B', 'C', 'D', 'E']	[‘A’, ‘B, ‘C’, ‘D’, ‘E’]	['A', 'B', 'C', 'D', 'E']
Distância do percurso (km)	4	4	4	4	4	4	4	4
Pessoas socorridas	500	500	500	500	500	500	500	500

Percorreram a mesma distância uma vez que o percurso era linear e socorreram 500 pessoas, pois o limite de carga era 500 e o número de medicamentos do nó de origem era 600. Como o nó de destino apenas precisava de 350 medicamentos, todos os algoritmos demonstraram-se funcionais no que toca à otimização de socorrer outros nós no caminho até ao destino, por ordem crescente de prioridade.

O resultado da aplicação dos algoritmos foi a seguinte:

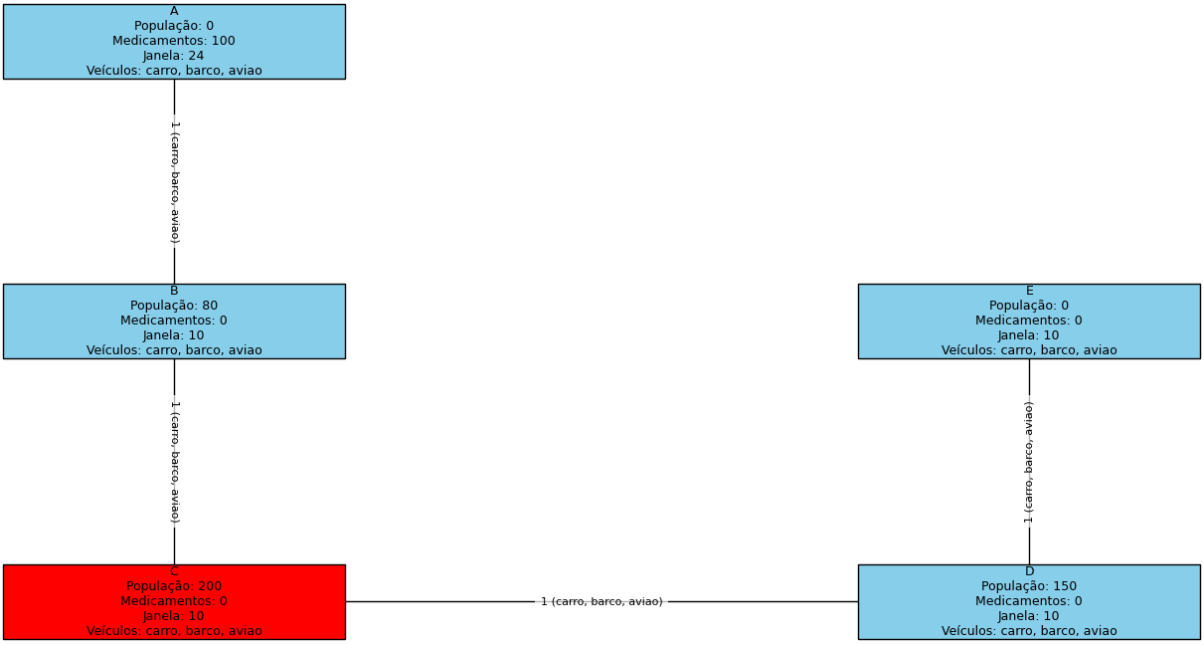

Figura 8 - Grafo linear sem restrições

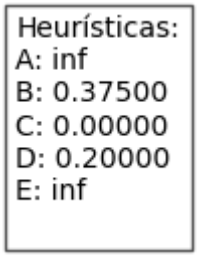

Figura 9 - Heurísticas do Grafo linear sem restrições

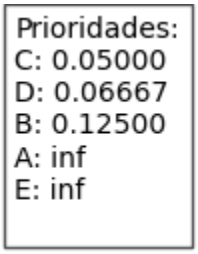

Figura 10 - Prioridades do Grafo linear sem restrições

Como é possível verificar, todos os algoritmos têm o mesmo custo (0.011429) e todos os algoritmos escolhem o veículo carro, pois é o veículo de menor custo. Quanto ao tempo de execução, o algoritmo Hill Climbing (380 ms) foi o mais lento e o Iterativo (0.408 ms) foi o mais rápido. Os nós solução e os nós visitados são os mesmo para todos os algoritmos [‘A’,’B’,’C’,’D’,’E’]. Em relação à distância do percurso e às pessoas socorridas os valores também são os mesmos para todos os algoritmos sendo a distância 4 km e as pessoas socorridas 500.

2. Grafo com ciclo sem restrições

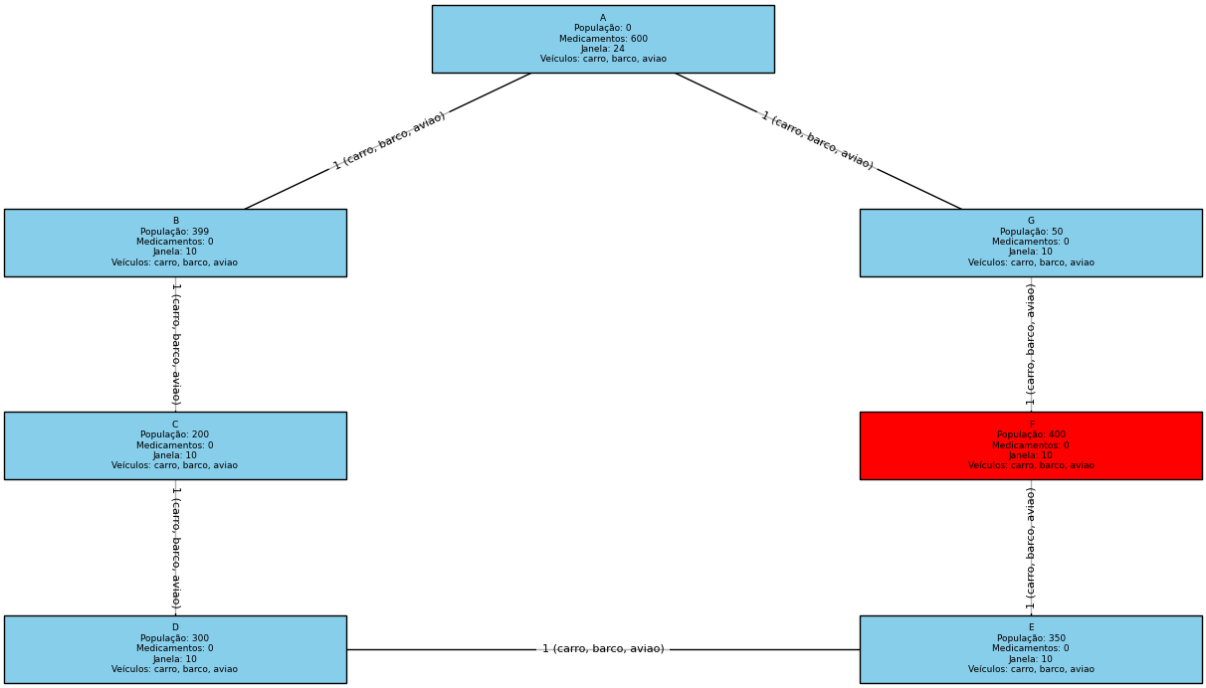

Figura 11 - Grafo com ciclo sem restrições

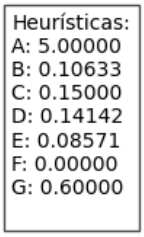

Figura 12 - Heurísticas do Grafo com ciclo sem restrições

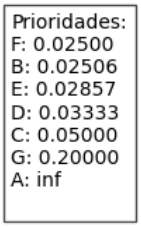

Figura 13 - Prioridades do Grafo com ciclo sem restrições

O grafo em forma de **ciclo** apresentado é de extrema importância para avaliar e validar algoritmos de procura em sistemas distribuídos. Este tipo de estrutura introduz mais complexidade do que um grafo linear, pois oferece mais do que 1 caminho para chegar ao mesmo nó de destino. Isso permite testar a eficiência dos algoritmos na escolha de caminhos ótimos, considerando custos acumulados e heurísticas.

Além disso, a ausência de bloqueios nas arestas e a possibilidade de todos os veículos utilizarem qualquer trajeto asseguram que o foco da análise permaneça nos próprios algoritmos, eliminando interferências externas. O nó A, que possui medicamentos, contrasta com os demais nós que necessitam de recursos, simulando cenários de alocação em rede. Esta configuração é ideal para verificar como os algoritmos lidam com ciclos e decidem entre

múltiplas alternativas viáveis, garantindo que a solução encontrada seja eficiente em termos de custo e tempo.

A forma cíclica também é útil para testar a robustez dos algoritmos contra loops infinitos e validar sua capacidade de calcular corretamente os custos totais em trajetos mais complexos. Assim, este grafo contribui significativamente para uma análise abrangente e eficaz do desempenho e da precisão dos algoritmos.

grafo_ciclo_sem_restricoes	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.0125	0.005	0.005	0.005	0.0125	0.005	0.0125	---
Veículo escolhido	carro	carro	carro	carro	carro	carro	carro	---
Tempo de execução (ms)	0.394	1.104	0.897	0.690	0.920	1.102	0.969	---
Nós solução	[‘A’,’B’,’C’,’D’,’E’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	['A', 'B', 'C', 'D', 'E', 'F']	['A', 'G', 'F']	[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’]	---
Nós visitados	[‘A’,’B’,’C’,’D’,’E’,’F’]	[‘A’,’G’,’B’,’F’]	[‘A’,’B’,’G’,’C’,’F’]	[‘A’,’G’,’F’]	['A', 'B', 'C', 'D', 'E', 'F']	['A', 'B', ‘C’, 'G', 'F']	[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’]	[‘A’, ‘B’]
Distância do percurso (km)	5	2	2	2	5	2	5	---
Pessoas socorridas	500	450	450	450	500	450	500	---

Como se pode verificar, os algoritmos BFS, Uniforme, Iterativo e A* obtiveram um menor custo (0.005). O veículo escolhido foi o carro para todos os algoritmos, pois é o de menor custo. Em relação ao tempo de execução, os mais lentos são o BFS (1.104 ms) e o A* (1.102 ms) e o mais rápido é o DFS (0.394 ms). Para os algoritmos BFS, Uniforme, Iterativo e A* os nós solução são [‘A’,’G’,’F’], para os outros algoritmos a lista dos nós solução é maior o que indica que esses algoritmos passam por mais nós. Em relação aos nós visitados, o algoritmo que visita menos nós é o Iterativo [‘A’,’G’,’F’], ou seja, explora só o essencial para encontrar a solução. Todos os outros algoritmos visitam maior número de nós. A distância do percurso pode ser 2 km ou 5 km. Os algoritmos que cuja distância é 2 km, são os algoritmos que tiveram um menor custo, ou seja, BFS, Uniforme, Iterativo e A*. Assim os outros algoritmos não são tão eficazes a encontrar o percurso ideal. Em relação às pessoas socorridas, os algoritmos DFS, Greedy e Simulated Annealing são onde são socorridas mais pessoas (500). Os algoritmos que socorreram menos pessoas (450), pois o caminho que consideraram ótimo não tinha mais pessoas para socorrer tanto no nó destino como intermédios.

Neste grafo, o algoritmo de Hill-Climbing não consegue alcançar o destino desejado, ou seja, o nó F, pois este desloca-se apenas se o seu vizinho possuir uma heurística menor do que a sua. Assim sendo, neste grafo, tendo como nó inicial o nó A, este verifica que o seu vizinho de menor heurística é o nó B. De seguida, como o único vizinho de B, C, possui heurística maior do que B, o algoritmo fica “preso” no nó B, que se qualifica como um mínimo local.

3. Grafo com árvore sem restrições

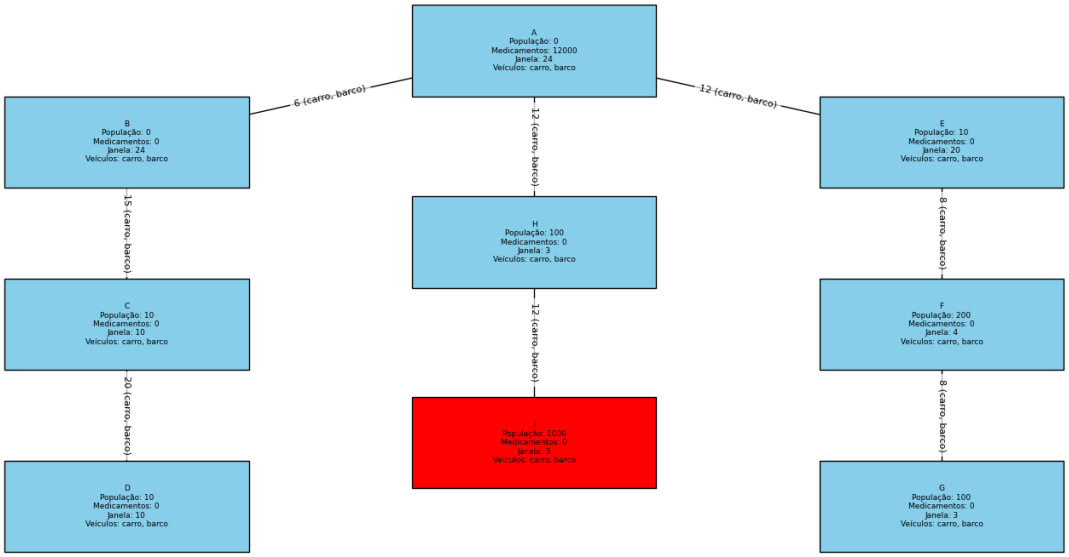

Figura 14 - Grafo com árvore sem restrições

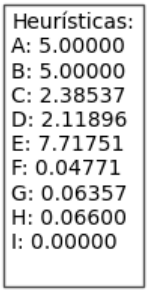

Figura 15 - Heurísticas do Grafo com árvore sem restrições

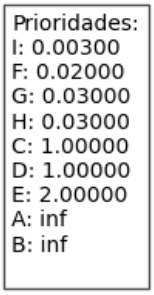

Figura 16 - Prioridades do Grafo com árvore sem restrições

Este grafo em formato de **árvore** permite testar algoritmos de procura em cenários com ramificações múltiplas, avaliando a sua eficiência na escolha dos melhores caminhos. Com o nó A a fornecer medicamentos e os restantes a necessitar de assistência, é possível analisar a distribuição de recursos de forma otimizada.

A ausência de restrições nas arestas garante que o foco permaneça nos algoritmos, permitindo validar a sua capacidade de evitar redundâncias, priorizar nós críticos e calcular trajetos eficientes.

grafo_arvore_sem_restricoes	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048
Veículo escolhido	carro	carro	carro	carro	carro	carro	carro	carro
Tempo de execução (ms)	0.768	0.820	0.975	0.863	0.654	1.239	0.738	460
Nós solução	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	['A', 'H', 'I']	[‘A’, ‘H’, ‘I’]	['A', 'H', 'I']
Nós visitados	[‘A’,’B’,’C’,’D’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’B’,’E’,’H’,’C’,’F’,’I’]	[‘A’,’B’,’E’,’H’,’F’,’C’,’I’]	[‘A’,’B’,’C’,’D’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’, ‘B’, ‘E’, ‘H’, ‘I’, ‘F’, ‘G’]	[‘A’, ‘H’, ‘I’]	['A', 'H', 'I']
Distância do percurso (km)	24	24	24	24	24	24	24	24
Pessoas socorridas	500	500	500	500	500	500	500	500

Podemos verificar que todos os algoritmos têm o mesmo custo (0.048) e que todos os algoritmos utilizam o veículo carro, pois o carro é o veículo com um custo menor. O algoritmo mais rápido é o Greedy (0.654 ms) e o mais lento é o Hill Climbing (460 ms). Os nós solução são iguais para todos os algoritmos [‘A’,’H’,’I’]. Já os nós visitados variam de algoritmo para algoritmo sendo os algoritmos Greedy, Simulated Annealing e Hill Climbing que visitam menos nós [‘A’,’H’,’I’]. Em relação à distância do percurso e às pessoas socorridas os valores também são os mesmos para todos os algoritmos sendo a distância 24 km e as pessoas socorridas 500.

4. Grafo grande sem restrições

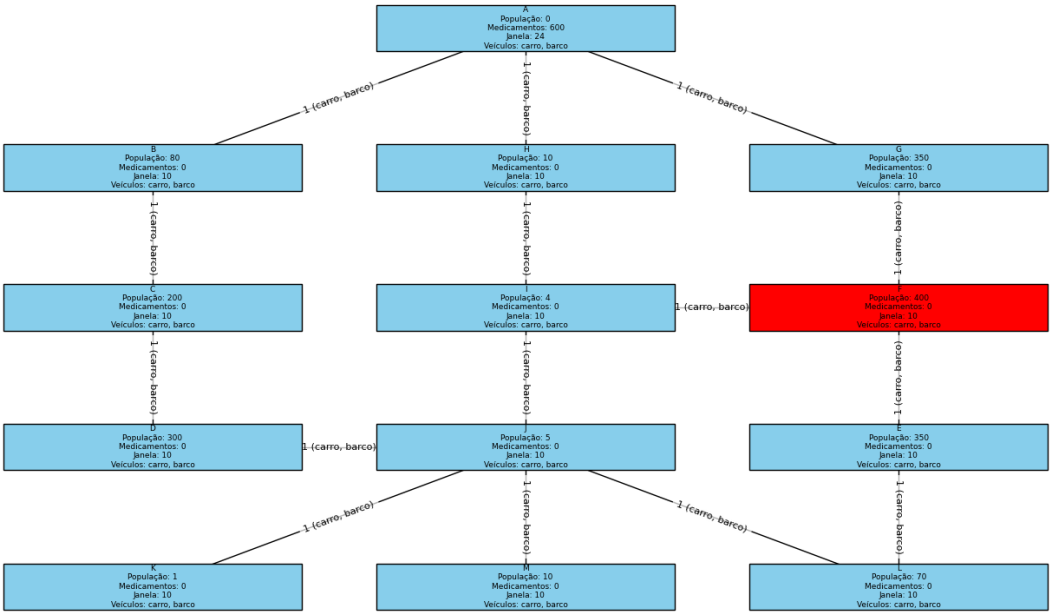

Figura 17 - Grafo grande sem restrições

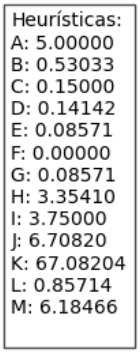

Figura 18 - Heurísticas do Grafo grande sem restrições

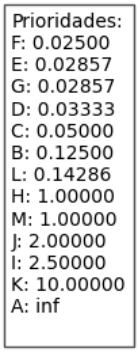

Figura 19 - Prioridades do Grafo grande sem restrições

Este grafo, com **maior número de nós** e várias ramificações, é ideal para testar algoritmos de procura em cenários que simulam redes moderadamente complexas. A estrutura proporciona múltiplos caminhos possíveis, permitindo avaliar a eficiência na escolha de trajetos e a capacidade de priorizar zonas críticas.

O nó A, com medicamentos disponíveis, destaca-se como ponto central de distribuição, enquanto os nós descendentes apresentam diferentes níveis de população e

janelas de tempo. A ausência de restrições nas arestas garante um ambiente controlado, focado exclusivamente no comportamento dos algoritmos. Este grafo é uma excelente ferramenta para validar a eficiência e consistência das soluções em redes de dimensão moderada.

grafo_grande_sem_restricoes	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.015	0.005	0.005	0.005	0.005	0.005	0.005	0.005
Veículo escolhido	carro	carro	carro	carro	carro	carro	carro	carro
Tempo de execução (ms)	4.584	3.407	3.830	0.512	1.730	0.581	0.803	490
Nós solução	[‘A’,’B’,’C’,’D’,’J’,’I’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	['A', 'G', 'F']	[‘A’, ‘G’, ‘F’]	['A', 'G', 'F']
Nós visitados	[‘A’,’B’,’C’,’D’,’J’,’I’,’F’]	[‘A’,’G’,’H’,’F’]	[‘A’,’B’,’G’,’H’,’C’,’F’]	[‘A’,’G’,’F’]	[‘A’,’G’,’F’]	[‘A’, ‘G’, ‘B’, ‘H’, ‘F’, ‘C’]	[‘A’, G’, F’]	['A', 'G', 'F']
Distância do percurso (km)	6	2	2	2	2	2	2	2
Pessoas socorridas	500	500	500	500	500	500	500	500

Como podemos verificar todos os algoritmos têm o mesmo custo (0.005) excepto o DFS que tem um custo superior aos outros (0.015). Todos os algoritmos utilizam como veículo o carro, visto ser o veículo com um custo menor. O algoritmo mais lento é o Hill Climbing (490 ms) e o mais rápido é o Iterativo (0.512 ms). Os nós solução também são iguais para todos os algoritmos [‘A’, ‘G’, ‘F’], excepto para o DFS. Os algoritmos que visitam menos nós são o Iterativo, o Greedy, o Simulated Annealing e o Hill Climbing [‘A’, ‘G’, ‘F’] todos os outros algoritmos visitam mais nós à procura da solução. Para todos os algoritmos a distância do percurso é 2 km exceto para o DFS cuja distância é 6 km. Em relação às pessoas socorridas, todos os algoritmos socorrem o mesmo número de pessoas, 500.

O DFS ao explorar o grafo de forma profunda faz com que os caminhos sejam mais longos, ou seja, ao ser utilizado o custo irá aumentar e a distância do percurso também vai aumentar.

5. Restrição com bloqueio de aresta e de veículos

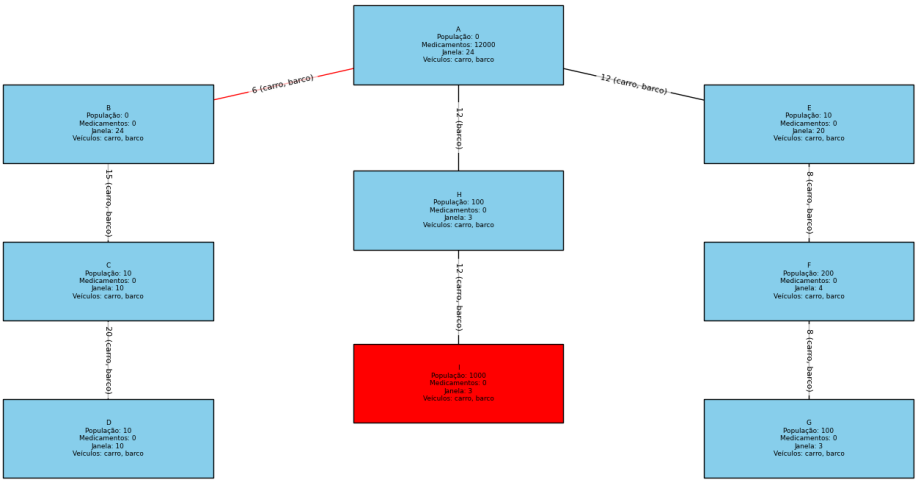

Figura 20 - Grafo com Restrição com bloqueio de aresta e de veículos

Heurísticas:
A: 5.00000
B: 5.00000
C: 4.00000
D: 4.47214
E: 4.00000
F: 0.00000
G: 0.06000
H: 0.06580
I: 5.00000

Figura 21 - Heurísticas do Grafo com Restrição com bloqueio de aresta e de veículos

Prioridades:
F: 0.02000
G: 0.03000
H: 0.03000
C: 1.00000
D: 1.00000
E: 2.00000
A: inf
B: inf
I: inf

Figura 22 - Prioridades do Grafo com Restrição com bloqueio de aresta e de veículos

O objetivo deste grafo permite testar como os algoritmos lidam com arestas bloqueadas e com arestas que não permitem a passagem de todos os veículos.

grafo_bloqueio_veiculos	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.072	0.072	0.072	0.072	0.072	0.072	0.072	0.072
Veículo escolhido	barco	barco	barco	barco	barco	barco	barco	barco
Tempo de execução (ms)	1.686	1.819	1.912	0.101	0.269	0.830	0.750	260
Nós solução	[‘A’,’H’,’I’,]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	['A', 'H', 'I']	[‘A’, ‘H’, ‘I’]	[‘A’, ‘H’, ‘I’]
Nós visitados	[‘A’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’E’,’H’,’F’,’I’]	[‘A’,’E’,’H’,’F’,’I’]	[‘A’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’, ‘E’, ‘F’, ‘I’]	[‘A’, ‘H’, ‘I’]	[‘A’, ‘H’, ‘I’]
Distância do percurso (km)	24	24	24	24	24	24	24	24
Pessoas socorridas	1000	1000	1000	1000	1000	500	1000	1000

Como podemos verificar todos os algoritmos têm o mesmo custo (0.072). Todos os algoritmos utilizam o veículo barco, pois existe uma aresta onde só é possível atravessar de barco. O algoritmo mais rápido é o Iterativo (0.101 ms) e o mais lento é o Hill Climbing (260 ms). Os nós solução são os mesmos para todos os algoritmos [‘A’,’H’,’I’] . Os algoritmos que visitam menos nós são o Greedy e o Simulated Annealing que visitam apenas os nós essenciais [‘A’,’H’,’I’], todos os outros algoritmos visitam mais nós do que os necessários para a solução. Em relação à distância do percurso e às pessoas socorridas os valores também são os mesmos para todos os algoritmos sendo a distância 24 km e as pessoas socorridas 1000 excepto o algoritmo A* que a distância é 2 km e as pessoas socorridas 500.

6. Restrição com carro de combustível insuficiente, barco de velocidade insuficiente e nó com população por socorrer, mas janela de tempo esgotou

Este teste foi projetado para analisar o comportamento dos algoritmos com veículos que não tem combustível suficiente para chegar ao destino ou não tem velocidade suficiente para chegar lá antes que a janela se esgote.

Para este teste, foi alterado o ficheiro de características dos veículos, em que o carro ficou com o valor de combustível 20 e o barco com um valor de velocidade 19.

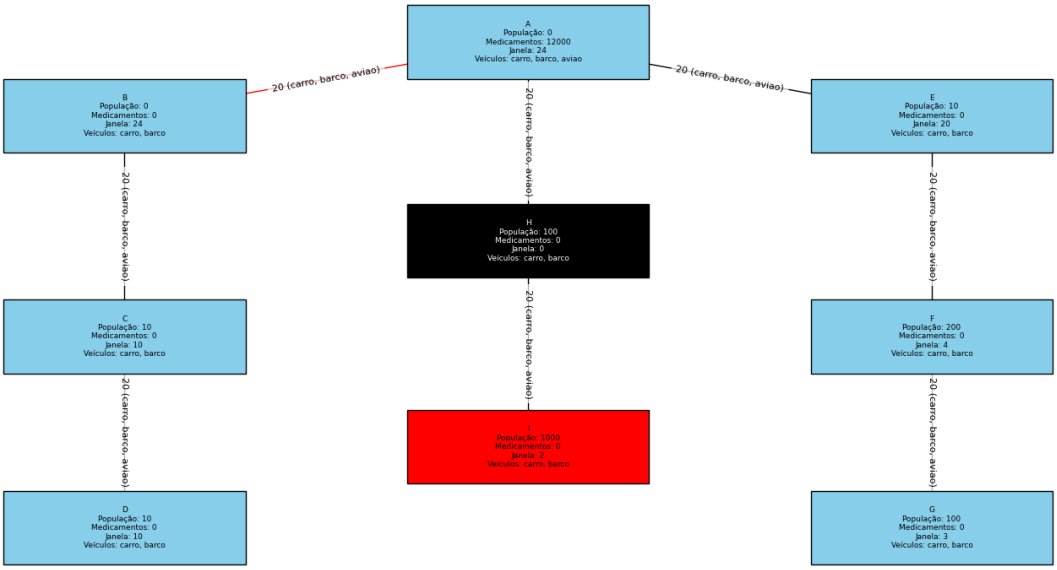

Figura 23 - Grafo com Restrição com carro de combustível insuficiente, barco de velocidade insuficiente e nó com população por socorrer, mas janela de tempo esgotou

Heurísticas:
A: 5.00000
B: 5.00000
C: 2.50000
D: 2.28254
E: 9.84886
F: 0.05000
G: 0.06848
H: 5.00000
I: 0.00000

Figura 24 - Heurística do Grafo com Restrição com carro de combustível insuficiente, barco de velocidade insuficiente e nó com população por socorrer, mas janela de tempo esgotou

Prioridades:
I: 0.00200
F: 0.02000
G: 0.03000
C: 1.00000
D: 1.00000
E: 2.00000
A: inf
B: inf
H: inf

Figura 25 - Prioridades do Grafo com Restrição com carro de combustível insuficiente, barco de velocidade insuficiente e nó com população por socorrer, mas janela de tempo esgotou

grafo_no_janela_esgotada	**DFS**	**BFS**	**Uniforme**	**Iterativo**	**Greedy**	**A***	**Simulated Annealing**	**Hill Climbing**
Melhor custo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Veículo escolhido	aviao	aviao	aviao	aviao	aviao	aviao	aviao	aviao
Tempo de execução (ms)	2.902	2.529	1.234	0.080	1.602	1.267	9.180	430
Nós solução	['A', 'H', 'I']	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’,’H’,’I’]	['A', 'H', 'I']	[‘A’,’H’,’I’]	['A', 'H', 'I']
Nós visitados	[‘A’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’E’,’H’,’F’,’I’]	[‘A’,’E’,’H’,’F’,’I’]	[‘A’,’E’,’F’,’G’,’H’,’I’]	[‘A’,’H’,’I’]	[‘A’, ‘E’, ‘H’, ‘F’, ‘G’, ‘I’]	[‘A’,’H’,’I’]	['A', 'H', 'I']
Distância do percurso (km)	40	40	40	40	40	40	40	40
Pessoas socorridas	1000	1000	1000	1000	1000	1000	1000	1000

Como podemos verificar, todos os algoritmos têm o mesmo custo (0.2) e todos utilizam o mesmo veículo, o aviao. O algoritmo mais rápido é o Iterativo (0.080 ms) e o mais lento é o Hill Climbing (430 ms). Os nós solução são os mesmos para todos os algoritmos [‘A’,’H’,’I’]. Os algoritmos que visitam menos nós são o Greedy e o Simulated Annealing que visitam apenas os nós essenciais [‘A’,’H’,’I’], todos os outros algoritmos visitam mais nós do que os necessários para a solução. Em relação à distância do percurso e às pessoas socorridas os valores também são os mesmos para todos os algoritmos sendo a distância 40 km e as pessoas socorridas 1000. Ressalvamos que o nó de origem e que o avião tem um limite de carga que permitia socorrer mais pessoas, mas o único nó intermediário tem janela com valor 0 e, por isso, já não é socorrido.

6. Conclusão

O objetivo deste projeto foi otimizar a distribuição de recursos por várias zonas afetadas por uma catástrofe natural, usando um grafo para representar o nosso mapa, tendo em conta vários fatores limitantes. Para atingir o objetivo deste projeto, aplicamos e adaptamos os vários algoritmos de procura e de otimização lecionados nas aulas.

Nesse sentido, realizamos seis testes, cada um com um grafo distinto, para aplicar os diferentes algoritmos e analisar o seu desempenho. Cada grafo apresentado contém uma nova característica em relação ao anterior. Assim, este projeto permitiu-nos perceber em que circunstâncias é mais benéfico usar cada algoritmo como visto no capítulo anterior. Através da testagem dos algoritmos para os diferentes grafos e dos seus resultados, foi-nos possível compreender melhor o desempenho de cada algoritmo ao analisar métricas como o custo, o veículo utilizado, o tempo de execução, a solução, os nós visitados, a distância percorrida e o número de pessoas que foram socorridas.

Por fim, ao aplicar os algoritmos em um cenário mais realista, foi-nos possível compreender de forma mais profunda a relevância destes algoritmos em situações críticas do mundo real.