My Project

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 calculos Namespace Reference

Functions

- calculos_auxiliares (G, otimo, calcular_erro_medio)

  Realiza cálculos auxiliares para comparar os algoritmos TSA e Suurballe.
- calculo_taxa_resolusao (G)

  Calcula e exibe a taxa de resolução dos algoritmos TSA e Suurballe.
- calculo_taxa_resolusao_otima (G)

  Calcula e exibe a taxa de resolução ótima do TSA em comparação com o Suurballe.
- calculo_erro (G)

  Calcula e exibe o erro médio percentual do custo do TSA em relação ao Suurballe.

3.1.1 Function Documentation

3.1.1.1 calculo_erro()

```
calculos.calculo_erro (
            G)
```

Calcula e exibe o erro médio percentual do custo do TSA em relação ao Suurballe.

O erro é calculado para pares de nós onde ambos os algoritmos encontraram uma solução. O erro percentual para um par é ((custo_TSA - custo_Suurballe) / custo_Suurballe) $*$ 100. A função exibe o erro médio acumulado sobre todos esses pares válidos.

Parameters

G	O grafo (NetworkX DiGraph) para análise.

Note

Utiliza `calculos_auxiliares`. Espera que o utilizador pressione Enter para continuar.

3.1.1.2 calculo_taxa_resolusao()

```
calculos.calculo_taxa_resolusao (
              G)
```

Calcula e exibe a taxa de resolução dos algoritmos TSA e Suurballe.

Utiliza a função `calculos_auxiliares` para obter o número total de pares de nós e o número de pares resolvidos por cada algoritmo. Em seguida, imprime a taxa de resolução percentual para cada um.

Parameters

G	O grafo (NetworkX DiGraph) para análise.

Note

Espera que o utilizador pressione Enter para continuar após a exibição.

3.1.1.3 calculo_taxa_resolusao_otima()

```
calculos.calculo_taxa_resolusao_otima (
              G)
```

Calcula e exibe a taxa de resolução ótima do TSA em comparação com o Suurballe.

Considera uma solução do TSA como ótima se o seu custo total for igual ao custo total da solução do Suurballe para o mesmo par de nós. A taxa é calculada como a percentagem de soluções ótimas encontradas pelo TSA em relação ao total de soluções encontradas pelo Suurballe (que é considerado o benchmark ótimo).

Parameters

G	O grafo (NetworkX DiGraph) para análise.

Note

Utiliza `calculos_auxiliares`. Espera que o utilizador pressione Enter para continuar.

3.1.1.4 calculos_auxiliares()

```
calculos.calculos_auxiliares (
              G,
              otimo,
              calcular_erro_medio)
```

Realiza cálculos auxiliares para comparar os algoritmos TSA e Suurballe.

Itera sobre todos os pares de nós únicos no grafo, executa ambos os algoritmos (TSA e Suurballe) para cada par e recolhe estatísticas como o número de soluções encontradas por cada um, soluções ótimas (se aplicável) e o erro percentual do TSA em relação ao Suurballe.

Parameters

G	O grafo (NetworkX DiGraph) sobre o qual os cálculos são realizados.
otimo	Booleano. Se True, verifica se o custo do TSA é igual ao do Suurballe (considerado como ótimo) e conta essas ocorrências.
calcular_erro_medio	Booleano. Se True, calcula o erro percentual médio do custo total do TSA em relação ao custo total do Suurballe.

Returns

Tuple contendo:

- pares (list): Lista de todos os pares de nós (origem, destino) no grafo.
- pares_validos (int): Número de pares para os quais tanto o TSA como o Suurballe encontraram uma solução (ambos os caminhos).
- resolvidos_tsa (int): Número total de pares para os quais o TSA encontrou ambos os caminhos.
- resolvidos_sur (int): Número total de pares para os quais o Suurballe encontrou ambos os caminhos.
- resolvidos_otimos (int): Se 'otimo' for True, o número de pares válidos onde o custo do TSA foi igual ao custo do Suurballe. Caso contrário, este valor não é significativo.
- erro_medio (float): Se 'calcular_erro_medio' for True, o erro percentual médio do TSA em relação ao Suurballe para os 'pares_validos'. Caso contrário, é 0.0.

3.2 draw Namespace Reference

Functions

- draw_network (G, node_mapping, origem, destino, caminho_tsa, caminho2, caminho_sur, caminho3, algoritmo)

  Desenha o grafo destacando até quatro caminhos e os nós de origem e destino.
- draw_empty_network (G, node_mapping)

  Desenha uma rede vazia, sem destacar caminhos ou nós específicos.
- draw_suurballe (G, origem_split, destino_split, caminho1_split, caminho2_split, filename)

  Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

3.2.1 Function Documentation

3.2.1.1 draw_empty_network()

```
draw.draw_empty_network (
              G,
              node_mapping)
```

Desenha uma rede vazia, sem destacar caminhos ou nós específicos.

Esta função gera uma visualização gráfica do grafo, mostrando apenas os nós com os seus rótulos (índice e nome) e as arestas. Não há destaque para origem, destino ou caminhos.

Parameters

G	O grafo NetworkX direcionado a ser desenhado.
node_mapping	Mapa de nós, associando índices numéricos (ou identificadores internos) aos nomes reais dos nós (strings) no grafo.

Note

Esta função é útil para exibir apenas a estrutura do grafo.

A imagem gerada é guardada em "output/Rede Original.png".

3.2.1.2 draw_network()

```
draw.draw_network (
              G,
              node_mapping,
              origem,
              destino,
              caminho_tsa,
              caminho2,
              caminho_sur,
              caminho3,
              algoritmo)
```

Desenha o grafo destacando até quatro caminhos e os nós de origem e destino.

Esta função visualiza o grafo, colorindo os nós de origem e destino e desenhando os caminhos fornecidos com cores distintas. A legenda é ajustada conforme o algoritmo selecionado.

Parameters

G	O grafo NetworkX direcionado.
node_mapping	Dicionário que mapeia o índice numérico (ou identificador interno) ao nome real do nó (string) a ser exibido.
origem	O nome (string) do nó de origem.
destino	O nome (string) do nó de destino.
caminho_tsa	Lista de nós (nomes) representando o primeiro caminho encontrado pelo Two-Step Approach (geralmente o mais curto). Pode ser None.
caminho2	Lista de nós (nomes) representando o segundo caminho disjunto encontrado pelo Two-Step Approach. Pode ser None.
caminho_sur	Lista de nós (nomes) representando o primeiro caminho encontrado pelo algoritmo de Suurballe. Pode ser None.
caminho3	Lista de nós (nomes) representando o segundo caminho disjunto encontrado pelo algoritmo de Suurballe. Pode ser None.
algoritmo	Inteiro que indica o(s) algoritmo(s) cujos resultados devem ser exibidos: 1 - Apenas Two-Step Approach (usa `caminho_tsa` e `caminho2`). 2 - Apenas Suurballe (usa `caminho_sur` e `caminho3`). 3 - Ambos os algoritmos (TSA e Suurballe), exibidos em subplots separados. (usa `caminho_tsa`, `caminho2`, `caminho_sur`, `caminho3`).

Note

Se `algoritmo` for 3, a função cria dois subplots. Caso contrário, um único plot.

A imagem gerada é guardada em "output/Rede Final.png".

3.2.1.3 draw_suurballe()

```
draw.draw_suurballe (
              G,
              origem_split,
              destino_split,
              caminho1_split,
              caminho2_split,
              filename)
```

Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

A função visualiza o grafo após a aplicação de uma etapa do algoritmo Suurballe, destacando os caminhos encontrados (se fornecidos) e as arestas internas (no formato '_in'/'_out').

Parameters

G	Grafo NetworkX direcionado (potencialmente dividido/transformado), representando a rede.
origem_split	Nó de origem no formato dividido (p.ex., 'A_out') ou original.
destino_split	Nó de destino no formato dividido (p.ex., 'B_in') ou original.
caminho1_split	Lista de nós representando o primeiro caminho encontrado nesta etapa (pode ser None). Será desenhado a verde.
caminho2_split	Lista de nós representando o segundo caminho encontrado nesta etapa (pode ser None). Será desenhado a azul.
filename	Nome base do arquivo (sem extensão) usado para salvar a imagem gerada, representando o estado atual do grafo (p.ex., "Step 1 - Primeiro Caminho").

Note

A função cria uma visualização do grafo, destacando os caminhos e as arestas em diferentes cores para facilitar a compreensão do algoritmo.

As arestas internas (de um nó dividido, ligando `X_in` a `X_out`) são desenhadas com uma linha tracejada e de cor prata.

A imagem é guardada em "output/{filename}.png".

3.3 functions Namespace Reference

Functions

- retrieve_data (data)

  Processa os dados de entrada (string) para criar um grafo direcionado NetworkX.
- find_best_paths (G, origem, destino, algoritmo)

  Encontra os dois melhores caminhos disjuntos em termos de nós (exceto origem/destino) entre dois nós, com base no menor custo, usando a abordagem Two-Step.
- suurballe (G, origem_orig, destino_orig, algoritmo, option, calculo)

  Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em arestas entre nós de origem e destino num grafo.
- split_nodes (G, source_orig, target_orig, path=None)

  Divide os nós especificados de um grafo G em nós de entrada (`_in`) e saída (`_out`).
- merge_split_path (split_path)

  Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.
- is_valid_path (path, original_graph)

  Verifica se um caminho (potencialmente de um grafo dividido) corresponde a um caminho válido no grafo original.
- path_cost (path, G)

  Calcula o custo de um caminho no grafo G.

3.3.1 Function Documentation

3.3.1.1 find_best_paths()

```
functions.find_best_paths (
            G,
            origem,
```

```
            destino,
            algoritmo)
```

Encontra os dois melhores caminhos disjuntos em termos de nós (exceto origem/destino) entre dois nós, com base no menor custo, usando a abordagem Two-Step.

Esta função primeiro calcula o caminho mais curto (path1) entre os nós de origem e destino usando Dijkstra (baseado no atributo 'cost' das arestas). Em seguida, cria uma cópia do grafo e remove todas as arestas de path1 e todos os nós intermediários de path1 (nós que não são nem a origem nem o destino). Depois, calcula o caminho mais curto (path2) nesta cópia modificada do grafo. Se não existir um segundo caminho, notifica e retorna None para path2 e cost2.

Parameters

G	O grafo NetworkX direcionado.
origem	O nome (string) do nó de origem.
destino	O nome (string) do nó de destino.
algoritmo	Inteiro que indica o contexto do algoritmo (p.ex., 1 para TSA, 3 para Ambos). Usado para controlar mensagens de impressão específicas do algoritmo. Se None, impressões genéricas ou nenhumas são feitas.

Returns

Tuple (path1, cost1, path2, cost2), onde:

- path1 (list/None): O primeiro caminho mais curto (lista de nós). None se não houver caminho.
- cost1 (float/None): O custo total do primeiro caminho. None se não houver caminho.
- path2 (list/None): O segundo caminho mais curto disjunto em nós. None se não existir.
- cost2 (float/None): O custo total do segundo caminho. None se não existir.

3.3.1.2 is_valid_path()

```
functions.is_valid_path (
            path,
            original_graph)
```

Verifica se um caminho (potencialmente de um grafo dividido) corresponde a um caminho válido no grafo original.

Primeiro, o `path` (que pode conter nós com sufixos `_in`/`_out`) é convertido para um caminho com nomes de nós originais usando `merge_split_path`. Depois, verifica-se se cada par consecutivo de nós no caminho merged forma uma aresta existente no `original_graph`.

Parameters

path	A lista de nós do caminho, possivelmente com sufixos `_in`/`_out`.
original_graph	O grafo NetworkX original (sem nós divididos).

Returns

True se o caminho merged for válido (não vazio, com pelo menos 2 nós, e todas as arestas existem no `original_graph`), False caso contrário.

3.3.1.3 merge_split_path()

```
functions.merge_split_path (
            split_path)
```

Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.

Esta função recebe uma lista de nós que podem ter sufixos '_in' ou '_out' (indicando que vieram de um processo de "node splitting") e remove esses sufixos para retornar uma lista de nomes de nós originais. Nós consecutivos no caminho que mapeiam para o mesmo nó original são consolidados (apenas uma instância do nó original é mantida).

Parameters

split_path	Lista de strings, onde cada string é um nome de nó do grafo dividido. Pode conter nomes como 'A_in', 'A_out', ou 'B' (se 'B' não foi dividido).

Returns

Uma lista de strings com os nomes dos nós originais, sem sufixos e sem duplicatas consecutivas. Retorna lista vazia se `split_path` for None ou vazio.

Note

Exemplo: ['S_out', 'A_in', 'A_out', 'B_in', 'B_out', 'T_in'] -> ['S', 'A', 'B', 'T']

3.3.1.4 path_cost()

```
functions.path_cost (
            path,
            G)
```

Calcula o custo de um caminho no grafo G.

Parameters

path	Lista de nós representando o caminho.
graph_G	O grafo NetworkX original.

Returns

O custo total do caminho, ou None se o caminho for inválido/vazio ou uma aresta não existir.

3.3.1.5 retrieve_data()

```
functions.retrieve_data (
            data)
```

Processa os dados de entrada (string) para criar um grafo direcionado NetworkX.

Esta função analisa uma string contendo informações sobre nós (com coordenadas) e links (arestas com custos) para construir um grafo. Os nós são adicionados com um atributo 'pos' para as suas coordenadas, e as arestas são adicionadas com um atributo 'cost'. As arestas são consideradas bidirecionais (adiciona target->source com o mesmo custo).

Parameters

data	A string de entrada que contém os dados da rede a serem analisados. Deve seguir um formato específico com seções "NODES (...)" e "LINKS (...)".

Returns

Tuple (G, node_mapping):

- G (nx.DiGraph): O grafo direcionado criado a partir dos dados.
- node_mapping (dict): Um dicionário que mapeia um índice numérico (baseado na ordem de leitura dos nós) para o nome (string) de cada nó.

3.3.1.6 split_nodes()

```
functions.split_nodes (
              G,
              source_orig,
              target_orig,
              path = None)
```

Divide os nós especificados de um grafo G em nós de entrada (`_in`) e saída (`_out`).

Cria um novo grafo H. Para cada nó no `path` fornecido (ou todos os nós se `path` for None e a lógica for ajustada, mas atualmente foca no `path`), o nó `N` é substituído por `N_in` e `N_out`. Uma aresta interna de `N_in` para `N_out` com custo zero é adicionada. As arestas originais `(U, V)` em G são remapeadas em H:

- Se U foi dividido: de `U_out`
- Se V foi dividido: para `V_in` Os nós de origem e destino (`source_orig`, `target_orig`) são mapeados para suas versões `_out` e `_in` respectivamente, se foram divididos.

Parameters

G	Grafo NetworkX direcionado original a ser transformado.
source_orig	O nome (string) do nó de origem no grafo G.
target_orig	O nome (string) do nó de destino no grafo G.
path	Opcional. Uma lista de nós (strings, nomes dos nós) que devem ser divididos. Se um nó está neste `path`, ele será dividido. Nós não presentes no `path` não são divididos e mantêm o seu nome original em H. Se `path` for `None` ou vazio, nenhum nó é dividido desta forma específica, mas `source_orig` e `target_orig` podem ainda ser tratados como `_out`/`_in` se fizerem parte do conjunto de nós a dividir (que neste caso seria vazio). A lógica atual implica que APENAS nós em `path` são divididos.

Returns

Tuple (H, s, t):

- H (nx.DiGraph): O novo grafo com nós divididos.
- s (str): O nó de origem em H (p.ex., `source_orig_out` ou `source_orig`).
- t (str): O nó de destino em H (p.ex., `target_orig_in` ou `target_orig`).

3.3.1.7 suurballe()

```
functions.suurballe (
              G,
              origem_orig,
              destino_orig,
              algoritmo,
              option,
              calculo)
```

Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em arestas entre nós de origem e destino num grafo.

O algoritmo segue vários passos:

1. Encontra um caminho inicial P1_original no grafo G. 0.5. Realiza "node splitting" nos nós de P1_original, criando um grafo H.
2. Encontra o primeiro caminho P1_split no grafo transformado H.
3. Transforma a rede H: calcula custos reduzidos, remove/inverte arcos de P1_split.
4. Encontra o segundo caminho P2_split no grafo residual H_residual.
5. Remove arcos opostos (desentrelaçamento) para formar os caminhos finais P1 e P2. Finalmente, os caminhos P1 e P2 são mapeados de volta para os nós do grafo original.

Parameters

G	O grafo NetworkX direcionado original.
origem_orig	O nome (string) do nó de origem no grafo original.
destino_orig	O nome (string) do nó de destino no grafo original.
algoritmo	Inteiro que indica o contexto (p.ex., 2 para Suurballe, 3 para Ambos). Controla impressões e, potencialmente, chamadas de desenho.
option	Booleano. Se True, salta o desenho dos passos intermédios do Suurballe. Usado para acelerar quando apenas o resultado final é desejado.
calculo	Booleano. Se True, suprime a maioria das mensagens de impressão. Útil quando a função é chamada em loop para cálculos estatísticos.

Returns

Tuple (P1, cost1, P2, cost2), onde:

- P1 (list/None): O primeiro caminho disjunto em arestas (lista de nós). None se nenhum caminho for encontrado.
- cost1 (float/None): O custo total do primeiro caminho. None se P1 for None.
- P2 (list/None): O segundo caminho disjunto em arestas. None se não existir.
- cost2 (float/None): O custo total do segundo caminho. None se P2 for None.

3.4 menus Namespace Reference

Functions

- show_ask_network ()

  Apresenta um menu para o utilizador selecionar uma rede de uma lista predefinida ou inserir o caminho para um novo ficheiro de rede.
- ask_network ()

  Solicita ao utilizador o nome do ficheiro da rede.
- ask_origin_destiny (node_mapping)

  Solicita ao utilizador os nós de origem e destino com base num mapeamento fornecido.
- clear_screen ()

  Limpa a tela do terminal.
- ask_which_algorithm ()

  Exibe um menu para o utilizador escolher um algoritmo ou combinação a ser utilizada.
- ask_skip_forward ()

  Pergunta ao utilizador se deseja visualizar todos os passos intermédios de um algoritmo (especificamente pensado para o Suurballe) ou apenas o resultado final.
- ask_which_calculus ()

  Apresenta um menu para o utilizador escolher qual cálculo estatístico realizar.

Variables

- list networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.txt", "networks/nobel-germany.txt"]

3.4.1 Function Documentation

3.4.1.1 ask_network()

```
menus.ask_network ()
```

Solicita ao utilizador o nome do ficheiro da rede.

Esta função pede ao utilizador para digitar o nome de um ficheiro de rede (que se espera ter a extensão '.txt'). Constrói o caminho completo para o ficheiro, assumindo que ele reside no diretório 'networks/'.

Returns

str: O caminho completo para o ficheiro da rede (p.ex., "networks/nome_ficheiro.txt"). Retorna uma string vazia ou pode levantar erro se a entrada for inválida, dependendo da gestão de erros implementada posteriormente. A versão atual retorna o caminho mesmo que o ficheiro não exista ou a extensão seja incorreta, a validação é feita no chamador.

Note

A função valida se o nome do ficheiro termina com '.txt'. Se não, imprime uma mensagem de erro e, na implementação atual, o comportamento de retorno para nomes inválidos não impede a continuação (o chamador deve validar). Também informa sobre FileNotFoundError, mas não o trata diretamente aqui.

3.4.1.2 ask_origin_destiny()

```
menus.ask_origin_destiny (
            node_mapping)
```

Solicita ao utilizador os nós de origem e destino com base num mapeamento fornecido.

Limpa o ecrã e exibe uma lista numerada de nós disponíveis (baseada no `node_mapping`). O utilizador é solicitado a inserir os números correspondentes aos nós de origem e destino. A função valida se os números são válidos (existem no mapeamento) e se origem e destino são diferentes. Um gráfico da rede (presumivelmente aberto por `draw_empty_network` antes desta chamada) é mencionado como auxílio visual, e a função fecha qualquer figura Matplotlib aberta após a seleção.

Parameters

node_mapping	Dicionário que mapeia um identificador numérico (int) para o nome do nó (str). Ex: '{0: 'NóA', 1: 'NóB', ...}`.

Returns

Tuple (origem_nome, destino_nome):

- origem_nome (str): O nome do nó de origem selecionado.
- destino_nome (str): O nome do nó de destino selecionado.

3.4.1.3 ask_skip_forward()

```
menus.ask_skip_forward ()
```

Pergunta ao utilizador se deseja visualizar todos os passos intermédios de um algoritmo (especificamente pensado para o Suurballe) ou apenas o resultado final.

Apresenta as opções:

1. Passar todos os passos à frente (visualização final).

1. Ver grafos passo a passo. Valida a entrada do utilizador.

Returns

bool: True se o utilizador escolher passar à frente (opção 1), False se escolher ver passo a passo (opção 0).

3.4.1.4 ask_which_algorithm()

```
menus.ask_which_algorithm ()
```

Exibe um menu para o utilizador escolher um algoritmo ou combinação a ser utilizada.

Apresenta as opções:

1. Two-Step Approach
2. Suurballe
3. Usar ambos os métodos (para comparação visual lado a lado) Valida a entrada do utilizador para garantir que a escolha é uma das opções válidas.

Returns

int: A opção escolhida pelo utilizador (1, 2 ou 3).

3.4.1.5 ask_which_calculus()

```
menus.ask_which_calculus ()
```

Apresenta um menu para o utilizador escolher qual cálculo estatístico realizar.

As opções são:

1. Calcular taxa de resolução (percentagem de pares de nós para os quais os algoritmos TSA e Suurballe encontram soluções).
2. Calcular taxa de resolução ótima (percentagem de vezes que o TSA encontra uma solução com custo igual ao do Suurballe, considerando Suurballe como ótimo).
3. Calcular erro médio do custo (erro percentual médio do custo do TSA em relação ao Suurballe). Valida a entrada do utilizador.

Returns

int: A opção escolhida pelo utilizador (1, 2 ou 3).

3.4.1.6 clear_screen()

```
menus.clear_screen ()
```

Limpa a tela do terminal.

Utiliza o comando 'cls' para Windows ('nt') e 'clear' para outros sistemas operativos (Linux, macOS).

3.4.1.7 show_ask_network()

```
menus.show_ask_network ()
```

Apresenta um menu para o utilizador selecionar uma rede de uma lista predefinida ou inserir o caminho para um novo ficheiro de rede.

A função limpa o ecrã, exibe as redes disponíveis na lista `networks`. O utilizador pode escolher uma rede pelo número ou optar por inserir o nome de um novo ficheiro de rede (que deve estar na pasta 'networks/' e ter extensão '.txt'). Se um novo ficheiro for adicionado com sucesso, ele é acrescentado à lista `networks` para a sessão atual. Finalmente, lê os dados da rede escolhida, cria o grafo usando `retrieve_data` e retorna o grafo e o mapeamento de nós.

Returns

Tuple (G, node_mapping):

- G (nx.DiGraph): O grafo NetworkX criado a partir do ficheiro de rede selecionado.
- node_mapping (dict): Um dicionário que mapeia índices numéricos (da ordem de leitura) para os nomes dos nós (strings).

Note

Se o ficheiro não for encontrado ou houver erro na leitura/formato, a função pode levantar exceções (p.ex., FileNotFoundError, ValueError de retrieve_data).

3.4.2 Variable Documentation

3.4.2.1 networks

```
list menus.networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.↩
txt", "networks/nobel-germany.txt"]
```

3.5 task Namespace Reference

Functions

- main ()

  Função principal que executa o ciclo do menu da aplicação.

3.5.1 Function Documentation

3.5.1.1 main()

```
task.main ()
```

Função principal que executa o ciclo do menu da aplicação.

Apresenta um menu principal com opções para determinar caminhos, realizar cálculos estatísticos ou sair do programa.

Chapter 4

File Documentation

4.1 calculos.py File Reference

Namespaces

- namespace calculos

Functions

- calculos.calculos_auxiliares (G, otimo, calcular_erro_medio)

  Realiza cálculos auxiliares para comparar os algoritmos TSA e Suurballe.
- calculos.calculo_taxa_resolusao (G)

  Calcula e exibe a taxa de resolução dos algoritmos TSA e Suurballe.
- calculos.calculo_taxa_resolusao_otima (G)

  Calcula e exibe a taxa de resolução ótima do TSA em comparação com o Suurballe.
- calculos.calculo_erro (G)

  Calcula e exibe o erro médio percentual do custo do TSA em relação ao Suurballe.

4.2 draw.py File Reference

Namespaces

- namespace draw

Functions

- draw.draw_network (G, node_mapping, origem, destino, caminho_tsa, caminho2, caminho_sur, caminho3, algoritmo)

  Desenha o grafo destacando até quatro caminhos e os nós de origem e destino.
- draw.draw_empty_network (G, node_mapping)

  Desenha uma rede vazia, sem destacar caminhos ou nós específicos.
- draw.draw_suurballe (G, origem_split, destino_split, caminho1_split, caminho2_split, filename)

  Desenha um grafo dividido ou transformado e destaca os caminhos encontrados pelo algoritmo Suurballe.

4.3 functions.py File Reference

Namespaces

- namespace functions

Functions

- functions.retrieve_data (data)

  Processa os dados de entrada (string) para criar um grafo direcionado NetworkX.
- functions.find_best_paths (G, origem, destino, algoritmo)

  Encontra os dois melhores caminhos disjuntos em termos de nós (exceto origem/destino) entre dois nós, com base no menor custo, usando a abordagem Two-Step.
- functions.suurballe (G, origem_orig, destino_orig, algoritmo, option, calculo)

  Implementa o algoritmo de Suurballe para encontrar dois caminhos disjuntos em arestas entre nós de origem e destino num grafo.
- functions.split_nodes (G, source_orig, target_orig, path=None)

  Divide os nós especificados de um grafo G em nós de entrada (`_in`) e saída (`_out`).
- functions.merge_split_path (split_path)

  Converte um caminho do grafo dividido (com nós '_in'/'_out') de volta para os nomes originais dos nós.
- functions.is_valid_path (path, original_graph)

  Verifica se um caminho (potencialmente de um grafo dividido) corresponde a um caminho válido no grafo original.
- functions.path_cost (path, G)

  Calcula o custo de um caminho no grafo G.

4.4 menus.py File Reference

Namespaces

- namespace menus

Functions

- menus.show_ask_network ()

  Apresenta um menu para o utilizador selecionar uma rede de uma lista predefinida ou inserir o caminho para um novo ficheiro de rede.
- menus.ask_network ()

  Solicita ao utilizador o nome do ficheiro da rede.
- menus.ask_origin_destiny (node_mapping)

  Solicita ao utilizador os nós de origem e destino com base num mapeamento fornecido.
- menus.clear_screen ()

  Limpa a tela do terminal.
- menus.ask_which_algorithm ()

  Exibe um menu para o utilizador escolher um algoritmo ou combinação a ser utilizada.
- menus.ask_skip_forward ()

  Pergunta ao utilizador se deseja visualizar todos os passos intermédios de um algoritmo (especificamente pensado para o Suurballe) ou apenas o resultado final.
- menus.ask_which_calculus ()

  Apresenta um menu para o utilizador escolher qual cálculo estatístico realizar.

Variables

- list menus.networks = ["networks/abilene.txt", "networks/atlanta.txt", "networks/nobel-eu.txt", "networks/nobel-germany.txt"]

4.5 task.py File Reference

Namespaces

- namespace task

Functions

- task.main ()

  Função principal que executa o ciclo do menu da aplicação.

Index