POLITÉCNICO DE PORTALEGRE

Escola Superior de Tecnologia, Gestão e Design

DEPARTAMENTO DE TECNOLOGIAS

Inteligência Artificial

Trabalho1 – 2024/2025 – 2º – Engenharia Informática

Docente: Luis Baptista

TSP – Procura Local

No problema do caixeiro-viajante (TSP – Traveling Salesman Problem) o objetivo é encontrar o caminho mais curto para visitar um conjunto de cidades (uma única vez) e regressar à cidade de origem. Um caminho é definido por uma sequência de cidades (implementado como uma lista de iCidades – ficha 3), cada uma com uma posição XY. A distância total do caminho (custo do caminho) é a soma das distâncias euclidianas entre as cidades consecutivas e entre a última e a primeira (função **distCircularIC** da ficha 3).
Foi colocado no PAE um conjunto de instâncias do TSP e código Python de apoio.

Pretende-se a implementação e o estudo empírico dos seguintes algoritmos de procura local:

- **optDistCircularIC** – Algoritmo já implementado na ficha 3, apenas o deve considerar para o estudo empírico.
- *Hill Climbing* (**Greedy**) – Em cada iteração este algoritmo gera todos os sucessores e escolhe o melhor.
- *Stochastic Hill Climbing* (**sGreedy**) – Igual ao **Greedy**, mas escolhe aleatoriamente de entre os melhores *b* sucessores (implemente o algoritmo de forma genérica, de forma a ser possível testar diferentes valores para *b*).
- *Partial Hill Climbing* (**pGreedy**) – Igual ao **Greedy**, mas em cada iteração o número de sucessores é *O(n)* em vez de $O(n^2)$ – *n* é o número de cidades.
- Um dos anteriores com *restarts* (**rGreedy**).

Considere ainda os seguintes pontos na implementação dos algoritmos greedy:

- Condição de paragem a usar:
  - quando todos os sucessores são piores;
  - número máximo de iterações;
  - nas últimas *k* iterações a função de custo melhora menos que ε (um valor pequeno).
- Os nomes das funções são exatamente os identificados a negrito e as suas interfaces são iguais à da função **optDistCircularIC**.
- A função *sucessor* é implementada com a função **trocaIC** (ficha 3).
- Os algoritmos começam sempre num estado inicial aleatório (por exemplo, pode fazer várias trocas aleatórias de cidades à lista de iCidades original).

Para os testes empíricos use pelo menos a instância *berlin52* e uma outra à escolha. Considere o tempo e o custo do caminho para avaliar os algoritmos para diferentes números *r* de iterações. Corra os algoritmos mais de uma vez e considere as médias.

Avançado: Proponha e implemente uma outra função sucessor. Avalie empiricamente esta função com um dos algoritmos aplicado à instância *berlin52*.

Além do código implementado deve ainda entregar um pequeno relatório crítico onde inclua as tabelas dos resultados empíricos e a respetiva interpretação/análise.