UENF

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Curso: Ciência de Computação **Data:** 20./.10./2022
Trabalho: Trabalho2 **Período:** 7º **Disciplina:** Top. Esp.: Heurísticas e Complexidade
Professor: Fermín Alfredo Tang **Turno:** Diurno
Nome do Aluno: .....Matrícula: .....

1.- Com base nos problemas estudados no Trabalho 1, escolher um problema da categoria NP da sua preferência e realize as seguintes tarefas:

i) Propor e implementar uma heurística construtiva para achar uma solução inicial para o problema escolhido. Descrever a representação da sua solução e seu método construtivo e implemente.

ii) Propor e implementar uma heurística de busca local para achar uma solução de melhor qualidade que a encontrada no item i). Descrever o mecanismo de busca local adotado e implemente.

iii) Teste as suas duas heurísticas para instâncias pequenas do problema.

iv) Teste as suas duas heurísticas para instâncias mais difíceis. Por exemplo, procure conjuntos de dados de teste (*test data sets*) para o problema escolhido, em repositórios semelhantes a: http://people.brunel.ac.uk/~mastjjb/jeb/info.html e escolha um conjunto de pelo menos três problemas para resolver. Mostre os resultados em uma tabela. Se possível compare a sua solução com a solução ótima, caso disponível.

Traveling Salesman Problem

1.

i)

Este trabalho aplicou um algoritmo heurístico construtivo chamado Traveling Salesman Problem (TSP) que visa reduzir eficientemente a complexidade do cálculo e encontrar os resultados ótimos da melhor solução TSP de comprimentos de passeio.

Neste trabalho busquei aplicar o algoritmo de maneira a encontrar os resultados ótimos, pelos quais a aplicação será bastante simples e fácil.

De modo a solucionar o Problema do Caixeiro Viajante, foi usado uma modificação do Self-Organizing Maps: descrito como uma grade (geralmente bidimensional) de nós, inspirada em uma rede neural. Intimamente relacionada ao mapa, está a ideia do modelo, ou seja, a observação do mundo real que o mapa está tentando representar. O objetivo da técnica é representar o modelo com menor número de dimensões, mantendo as relações de similaridade dos nós nele contidos ("Using Self-Organizing Maps to solve the Traveling Salesman Problem").
O Problema do Caixeiro Viajante é um desafio bem conhecido na Ciência da Computação: consiste em encontrar a rota mais curta possível que percorre todas as cidades de um determinado mapa apenas uma vez.

Data and Code

Os dados usados neste trabalhos são oriundo da plataforma disponibilizada pelo Prof. Fermín Alfredo Tang. Para detalhes seguir os seguintes links Problem instances, TSP Test Data, **National TSP Collection**.

Podera encontrar todos os codigos para este trabalho neste link: https://github.com/ARRETdaniel/22-2_topicos_Especiais_Heuristicas_e_Complexidade

ii)

Como descrito na seção anterior, estaremos aplicando uma modificação do Self-Organizing Maps.

Esta modificação consiste no uso da rede para resolver o TSP, o principal conceito a ser entendido é como modificar a função de vizinhança. Se, em vez de uma grade, declararmos uma matriz circular de neurônios, cada nó só terá consciência dos neurônios à frente e atrás dele. Ou seja, a semelhança interna funcionará apenas em uma dimensão. Fazendo essa pequena modificação, o mapa auto-organizado se comporta como um anel elástico, aproximando-se das cidades, mas tentando minimizar o perímetro das mesmas graças à função de vizinhança. Para garantir a convergência do mesmo, podemos incluir uma *taxa de aprendizado*, $\boldsymbol{\alpha}$, para controlar a exploração e aproveitamento do algoritmo.

De maneira a obter uma alta exploração, devemos incluir um decaimento tanto na função de vizinhança quanto na taxa de aprendizado. Decair a taxa de aprendizado garantirá um deslocamento menos agressivo dos neurônios ao redor do modelo, e decair a vizinhança resultará em uma exploração mais moderada dos mínimos locais de cada parte do modelo ("Using Self-Organizing Maps to solve the Traveling Salesman Problem").

Portanto, a regressão pode ser expressa como:

$$n_{t+1} = n_t + \alpha_t \cdot g(w_e, h_t) \cdot \Delta(e, n_t)$$

Onde $\boldsymbol{\alpha}$ é a taxa de aprendizado em um dado momento, e (g) é a função gaussiana centrada em um vencedor e com dispersão de vizinhança de h . A função de decaimento consiste em simplesmente multiplicar os dois descontos dados, γ, pela taxa de aprendizado e pela distância da vizinhança.

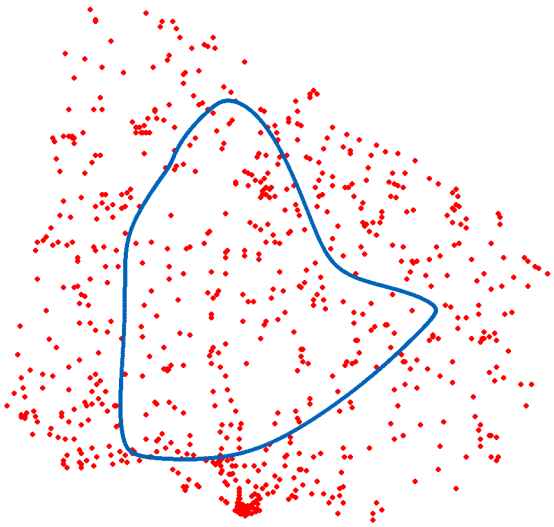
$$\alpha_{t+1} = \gamma_\alpha \cdot \alpha_t, \quad h_{t+1} = \gamma_h \cdot h_t$$

Esta expressão é de fato bastante semelhante à do Q-Learning, e a convergência é buscada de forma semelhante a esta técnica. Decair os parâmetros pode ser útil em tarefas de aprendizado não supervisionadas como as mencionadas acima.

Por fim, para obter a rota do SOM, basta associar uma cidade ao seu neurônio vencedor, percorrer o anel a partir de qualquer ponto e ordenar as cidades por ordem de aparecimento de seu neurônio vencedor no anel. Se várias cidades mapeiam para o mesmo neurônio, é porque a ordem de percorrer tais cidades não foi contemplada pelo SOM (por falta de relevância para a distância final ou por não ter precisão suficiente). Nesse caso, qualquer arranjo possível pode ser considerado para tais cidades ("Using Self-Organizing Maps to solve the Traveling Salesman Problem").

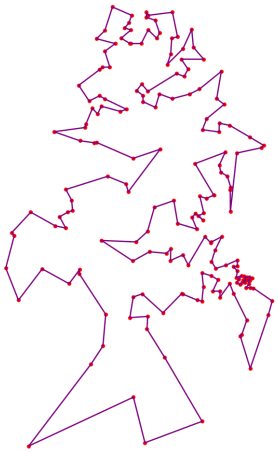
iii) Testando a heurísticas para instâncias pequenas do problema:

```
PS C:\Users\danie\Documents\Documents\Faculdade\22-2_topicos_Especiais_Heuristicas_e_Complexidade> python src/main.py assets/uy734.tsp
Problem with 734 cities read.
Network of 5872 neurons created. Starting the iterations:
Route found of length 101744.2685298634
```



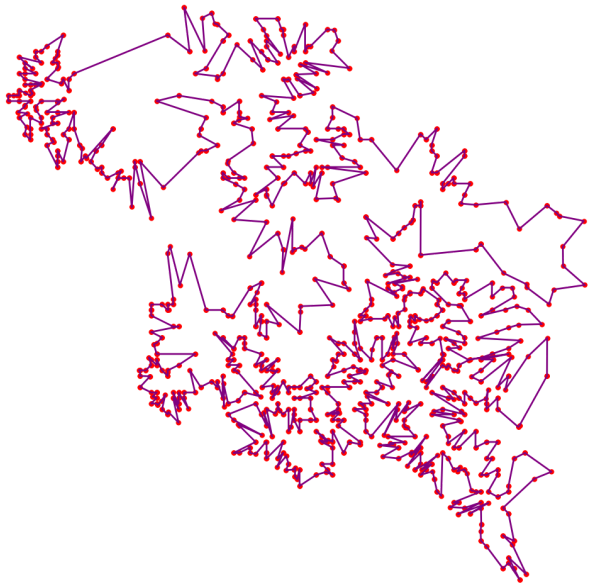
Uruguay 1000 iterations.

```
PS C:\Users\danie\Documents\Documents\Faculdade\22-2_topicos_Especiais_Heuristicas_e_Complexidade> python src/main.py assets/qa.tsp
Problem with 194 cities read.
Network of 1552 neurons created. Starting the iterations:
Completed 10000 iterations.000
Route found of length 10091.348673816037
```



Qatar 10.000 iterations.

```
PS C:\Users\danie\Documents\Documents\Faculdade\22-2_topicos_Especiais_Heuristicas_e_Complexidade> python src/main.py assets/rw.tsp
Problem with 1621 cities read.
Network of 12968 neurons created. Starting the iterations:
Completed 10000 iterations.000
Route found of length 36955.983996960786
```



Rwanda 10.000 iterations.
Optimal tour has length 26051.

iv) Testando a heurísticas para instâncias mais difíceis do problema:

Instance	Instance size	Iterations	Time (s)	Length	Optimal Tour Length	Quality
Qatar	194 Cities	14690	14.3	10233.89	93520.00	9.4%
Uruguay	734 Cities	17351	23.4	85072.35	79114.00	7.5%
Finland	10,639 Cities	37833	284.0	636580.27	520527.00	22.3%
Italy	16,862 Cities	39368	401.1	723212.87	557315.00	29.7%

A tabela acima reúne os resultados da estimativa, com o resultado médio de 5 execuções em cada uma das instâncias.

Para avaliar a implementação, usaremos algumas instâncias fornecidas pelo professor, mencionada anteriormente: National Traveling Salesman Problem. Essas instâncias são inspiradas em países reais e também incluem a rota ideal para a maioria deles, o que é parte

fundamental de nossa avaliação. A estratégia de avaliação consiste em executar várias instâncias do problema e estudar algumas métricas:

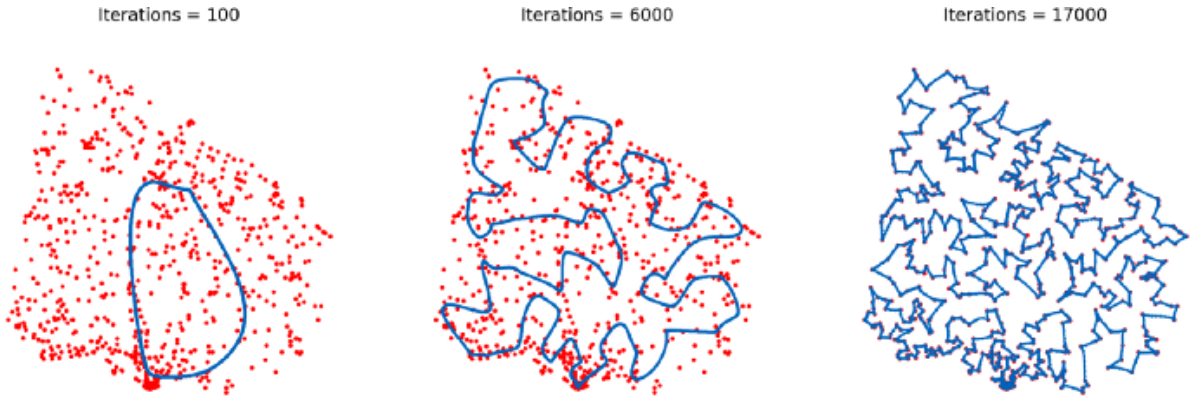
- **Tempo de execução** investido pela técnica para encontrar uma solução.
- **Qualidade** da solução, medida em função da rota ótima: uma rota que dizemos ser “10% mais longa que a rota ótima” é exatamente 1,1 vezes o comprimento da rota ótima.

Os parâmetros utilizados na avaliação são os encontrados pela parametrização da técnica. Esses parâmetros são:

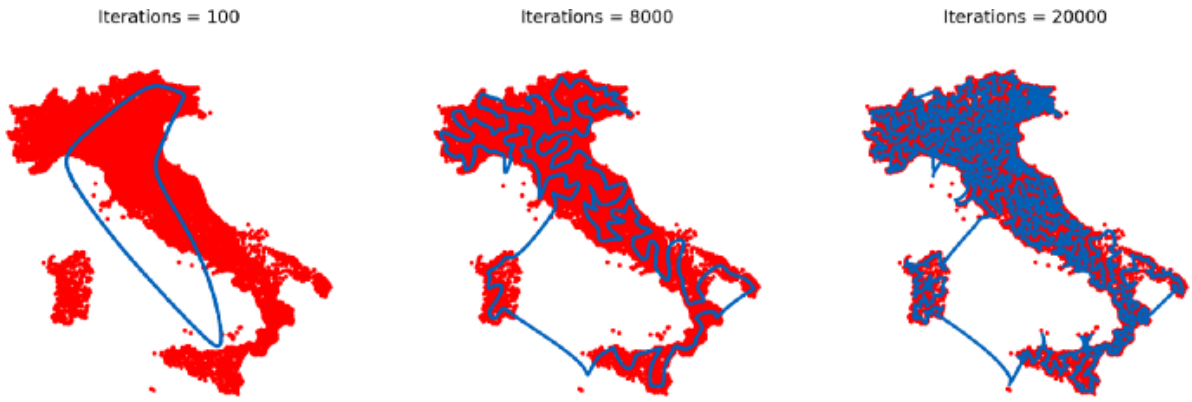
- Um tamanho populacional de 8 vezes as cidades do problema.
- Uma taxa de aprendizado inicial de 0,8 com uma taxa de desconto de 0,99997.
- Uma vizinhança inicial do número de cidades, decaída em 0,9997.

Esses parâmetros foram aplicados às seguintes instâncias:

- **Qatar**, contendo 194 cidades com um tour ótimo de 9352.
- **Uruguai**, contendo 734 cidades com um tour ótimo de 79114.
- **Finlândia**, contendo 10.639 cidades com umtour ótimo de 520.527.
- **Itália**, contendo 16.862 cidades com um tour ótimo de 557315.



Uruguay



Italy

Referências

Asrul Harun Ismail. “Domino algorithm: a novel constructive heuristics for traveling salesman problem.” *Domino algorithm: a novel constructive heuristics for traveling salesman problem*, IOP, 2018, https://iopscience.iop.org/article/10.1088/1757-899X/528/1/012043/pdf. Accessed 14 November 2022.

"Using Self-Organizing Maps to solve the Traveling Salesman Problem." *Diego Vicente*, https://diego.codes/post/som-tsp/. Accessed 14 November 2022.