Problemas NP-Completos em Clojure

Trabalho Final — Teoria da Computação / Linguagens Formais e Autômatos

Prof. Dr. Jefferson O. Andrade

2023/2

1 Objetivo do Projeto

Desenvolver um programa em Clojure que implemente algoritmos para resolver um problema NP-Completo específico. O projeto visa fornecer uma compreensão prática dos desafios associados à resolução de problemas NP-Completos e explorar soluções heurísticas ou exatas.

2 Descrição Detalhada

1. **Seleção do Problema**:
   - Escolher um problema NP-Completo para o foco do projeto, como o Problema do Caixeiro Viajante (*TSP*), Problema da Mochila (*Knapsack Problem*), Problema de Cobertura de Conjuntos (*Set Cover Problem*), etc.
2. **Implementação de Algoritmos Exatos**:
   - Implementar uma solução exata para o problema escolhido. Por exemplo, para o TSP, isso pode ser a busca por força bruta ou o algoritmo de *Branch-and-Bound*.
3. **Implementação de Algoritmos Heurísticos**:
   - Implementar um ou mais algoritmos heurísticos, como algoritmos genéticos, busca tabu, ou algoritmos *greedy*.
4. **Análise de Complexidade**:
   - Analisar a complexidade de tempo e espaço dos algoritmos implementados.
5. **Interface do Usuário**:
   - Criar uma interface básica que permita ao usuário inserir dados de entrada para o problema e escolher qual algoritmo usar. Preferencialmente, permitir que os dados sejam lidos de arquivo de algum modo.
6. **Experimentos e Avaliação**:
   - Realizar experimentos para comparar o desempenho dos diferentes algoritmos, tanto em termos de qualidade da solução quanto de eficiência computacional.
7. **Visualização**:
   - Se aplicável, adicionar visualização para ilustrar a solução do problema, como um gráfico para o TSP.
8. **Relatório**:

- Preparar um relatório detalhando a implementação, os resultados dos experimentos, e uma discussão sobre a eficácia dos algoritmos. O relatório deve ser preparado em LaTeX, e deve ser apresentando seguindo a formatação de um artigo cientifico.

9. **Entrega**:
   - O trabalho deve ser entregue em um arquivo em formato **ZIP** (não é RAR, não é LHA, é ZIP). Deste modo devem ser entregues dois artefatos: O relatório, e os códigos fontes. O arquivo ZIP deve obrigatoriamente ter a seguinte estrutura:

```
tf
├── «nome-do-relatório».pdf
└── my_app
    ├── doc
    │   └── intro.md
    ├── resources
    ├── src
    │   └── my_app
    │       ├── core.clj
    │       └── [...]
    ├── test
    │   └── my_app
    │       ├── core_test.clj
    │       └── [...]
    ├── CHANGELOG.md
    ├── .gitignore
    ├── .hgignore
    ├── LICENSE
    ├── project.clj
    └── README.md
```

   Evidentemente, `my_app` é o nome da sua aplicação. Você pode escolher o nome que desejar. Note que dentro do diretório `my_app` a estrutura é a estrutura padrão para projetos Leiningen. Você também pode criar quantos *namespaces* desejar ou precisar. O arquivo `README.md` deve conter todas as informações necessárias para compilar e executar o seu programa.

3 Tópicos de Aprendizado Envolvidos

- Compreensão profunda de problemas NP-Completos e suas implicações.
- Desenvolvimento e aplicação de algoritmos exatos e heurísticos.
- Análise empírica e teórica de complexidade algorítmica.
- Prática de habilidades de programação em Clojure.

4 Desafios Potenciais

- Implementar algoritmos eficientes para problemas NP-Completos, especialmente as soluções exatas.
- Escolher e implementar adequadamente algoritmos heurísticos que ofereçam boas soluções em um tempo razoável.
- Lidar com a análise e visualização de dados para grandes instâncias do problema.

5 Avaliação do Projeto

- Correção dos algoritmos implementados.
- Eficiência e eficácia das soluções heurísticas.
- Qualidade do relatório, incluindo clareza, análise e discussão dos resultados.
- Qualidade e organização do código, incluindo documentação.

Este projeto não apenas reforça a compreensão teórica dos alunos sobre a complexidade dos algoritmos e os desafios dos problemas NP-Completos, mas também desenvolve habilidades práticas em programação avançada e resolução de problemas complexos.