Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada

Disciplina: DIM0406 — Algoritmos Avançados
Docente: Sílvia Maria Diniz Monteiro Maia
Discente: Felipe Cortez de Sá

GRASP-VNS aplicado ao problema de Steiner com rotulação mínima

1 Introdução

Neste relatório é apresentado um algoritmo metaheurístico para resolver o problema da árvore de Steiner com rotulação mínima. O algoritmo é uma combinação do *Greedy Randomized Adaptative Search Procedure* apresentado por Consoli et al [1] com sua fase de busca local substituída por uma *Variable Neighbourhood Search*. Os resultados são comparados com o algoritmo exato implementado na primeira unidade, é feita uma análise de complexidade procurando identificar os bottlenecks, é descrito como são gerados os casos de teste, uma tabela mostra os resultados obtidos para uma média de dez execuções de caso de teste. Além disso, é feito um adendo para o relatório do primeiro trabalho, apresentando os resultados previamente faltantes à conclusão.

2 Metaheurísticas utilizadas

2.1 GRASP

O *Greedy Randomized Adaptative Search Procedure* é comumente utilizado em problemas de otimização combinatória. A cada iteração, é realizada uma fase de construção, em que se gera uma solução para o problema e posteriormente uma fase de busca local, procurando um mínimo local na vizinhança da solução gerada. Se a melhor solução global é encontrada na iteração, atualiza-se a variável que contém a melhor solução. As duas fases são repetidas até o critério de parada ser satisfeito, podendo ser o número de iterações ou o tempo de execução, por exemplo. Na fase de construção, é criada uma lista de candidatos restritos, função gulosa, restricted candidate list possui os candidatos cujos elementos adicionados minimizam os custos incrementais. O elemento é selecionado aleatoriamente da RCL. inicialização múltipla.

2.2 VNS

O *Variable Neighbourhood Search* faz uso de múltiplas estruturas de vizinhança, explorando vizinhanças cada vez mais distantes e maiores assim que o ótimo local é obtido por uma dessas vizinhanças. As diferentes vizinhanças são K_n e a vizinhança mais distante (normalmente de cardinalidade maior) é K_{max}.

3 Metaheurística aplicada ao problema

3.1 GRASP

```
grasp() {
    col = {}
    col* = {}
    int no_improv = 0

    while(no_improv < limite) {
        col = {}
        construct(col)
        vns(col)
        if(card(col) < card(col*)) {
            col* = col
            no_improv = 0
        } else {
            ++no_improv
        }
    }
}

construct() {
    if(iteration > 2) {
        rcl = {1, 1, ... , 1}
        col[random()] = 1
    } else {
        rcl = {0, 0, ... , 0}
    }

    while(comp(c) > 1) {
        rcl = argmin comp(c)
        col = col ∪ rcl[random()]
    }
}
```

Listing 1: Pseudocódigo para GRASP

4 Complexidade

5 Casos teste utilizados

Os casos teste utilizados são gerados automaticamente por um programa **`generate.c`** de acordo com parâmetros de entrada. Os parâmetros são **`SIZE`**, a quantidade de nós do grafo, **`COLORS`**, o número de rótulos, **`DENSITY`**, a proporção de arestas para cada nó, e **`BASIC`**, a quantidade de nós básicos. **`DENSITY`** funciona percorrendo a matriz de adjacência que representa o grafo e de acordo com a probabilidade definida (sendo 0 e 100 equivalentes a 0% e 100%, respectivamente) adicionando ou não uma aresta de rotulação aleatória ligando dois nós. O arquivo gerado é então passado para o programa principal.

A fim de comparar os resultados com o trabalho realizado por Consoli [1], os parâmetros dos testes são os mesmos, isto é, tem-se uma combinação entre **`DENSITY`** $\in \{80, 50, 20\}$, e para **`SIZE`** $= 100$, **`COLORS`** $\in \{25, 50, 100, 125\}$, para **`SIZE`** $= 500$, **`COLORS`** $\in \{125, 250, 500, 625\}$. **`BASIC`** recebe 20% de **`SIZE`**, isto é, 20 e 100, respectivamente.

O código que gera os arquivos de caso teste para os parâmetros desejados está em **`generate.py`**.

6 Resultados

7 Correções do primeiro trabalho

7.1 Técnica utilizada

7.2 Resultados

7.3 Conclusão

Referências

[1] S. Consoli, K. Darby-Dowman, N. Mladenovic, J.A. Moreno-Perez. *Variable neighbourhood search for the minimum labelling Steiner tree problem*. Annals of Operations Research, 2009.

https://www.researchgate.net/publication/225327721_Variable_neighbourhood_search_for_the_minimum_labelling_Steiner_tree_problem

[2] *Graphviz — Graph Visualization Software*.
http://www.graphviz.org/

[3] Glover, F., Kochenberger, G. A. et al. *Handbook of Metaheuristics*. Kluwer Academic Publishers

[4] *Stack Overflow — Execution time of a C program*.
http://stackoverflow.com/questions/5248915/execution-time-of-c-program