SOP com Busca Tabu

Augusto Bennemann
Fabrício Mazzola

Introdução

- Estudar um problema NP-completo
- Desenvolver a formulação matemática em PI
- Implementar uma meta-heurística
- Avaliação comparativa da resolução do problema entre os métodos Programação Linear e Meta-Heurística.
- Conclusão

Introdução

- Estudar um problema NP-completo (Sequential Ordering Problem)
- Desenvolver a formulação matemática em PI
- Implementar uma meta-heurística
- Avaliação comparativa da resolução do problema entre os métodos Programação Linear e Meta-Heurística.
- Conclusão

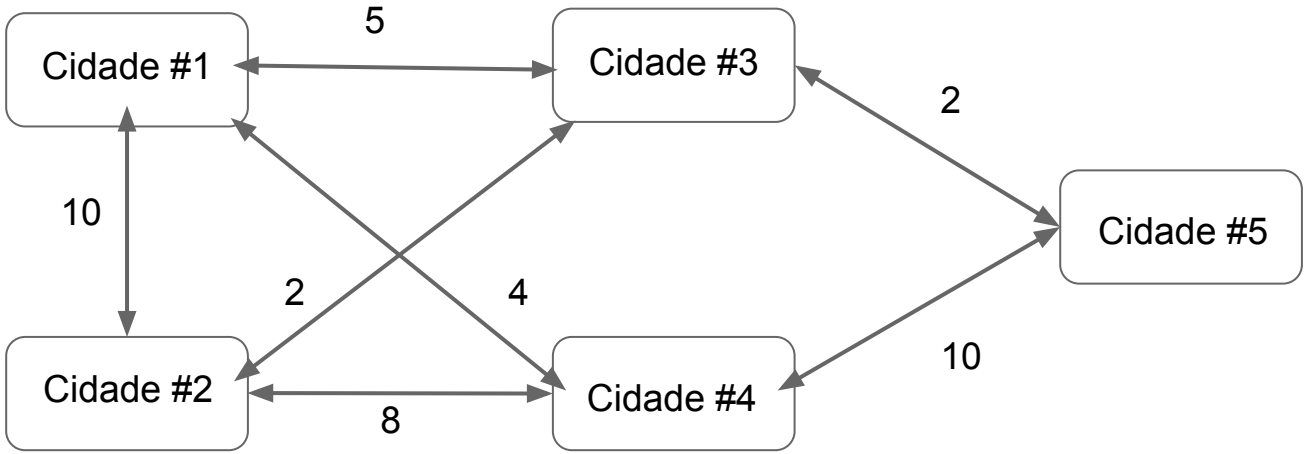
Introdução

- Estudar um problema NP-completo (Sequential Ordering Problem)
- Desenvolver a formulação matemática em PI
- Implementar uma meta-heurística (Busca Tabu)
- Avaliação comparativa da resolução do problema entre os métodos Programação Linear e Meta-Heurística.
- Conclusão

Problema - SOP

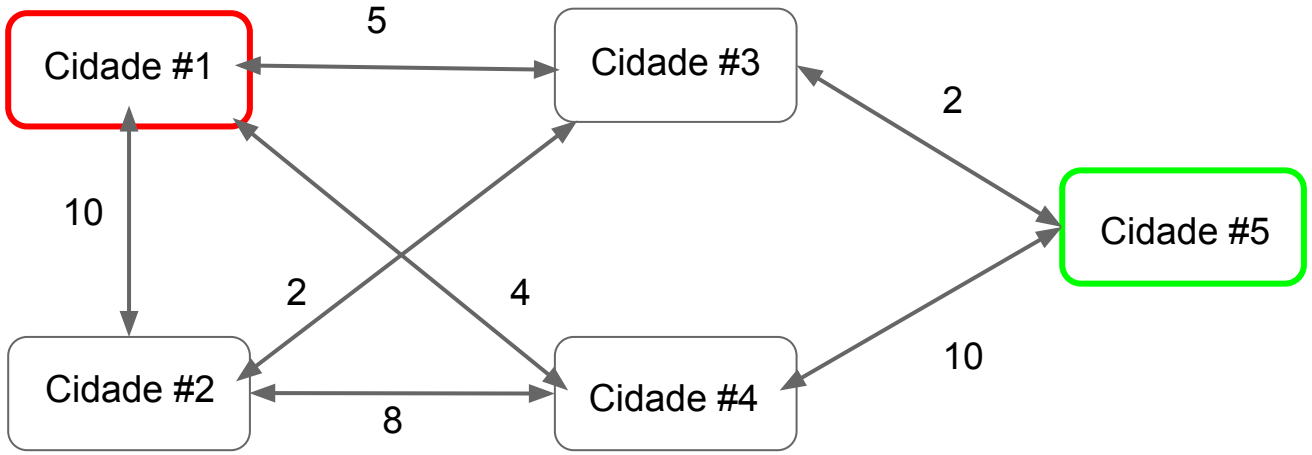
- Variação do Caixeiro Viajante Assimétrico (ATSP)
- Objetivos:
  - Encontrar o caminho de menor custo.
- Restrições
  - Respeitar restrições de precedência entre os vértices.
  - Visitar todos os vértices do grafo exatamente uma vez.
  - Partir do vértice 1
  - Chegar no vértice N

Pares de precedência: (#2, #3), (#2, #4), (#3, #5)



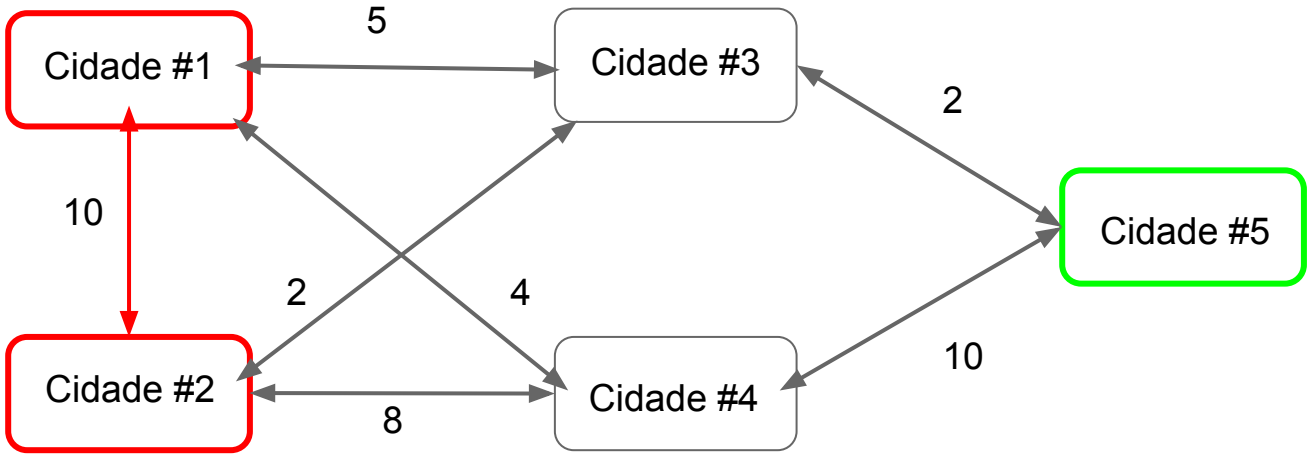
Exemplo de instância

Pares de precedência: (#2, #3), (#2, #4), (#3, #5)



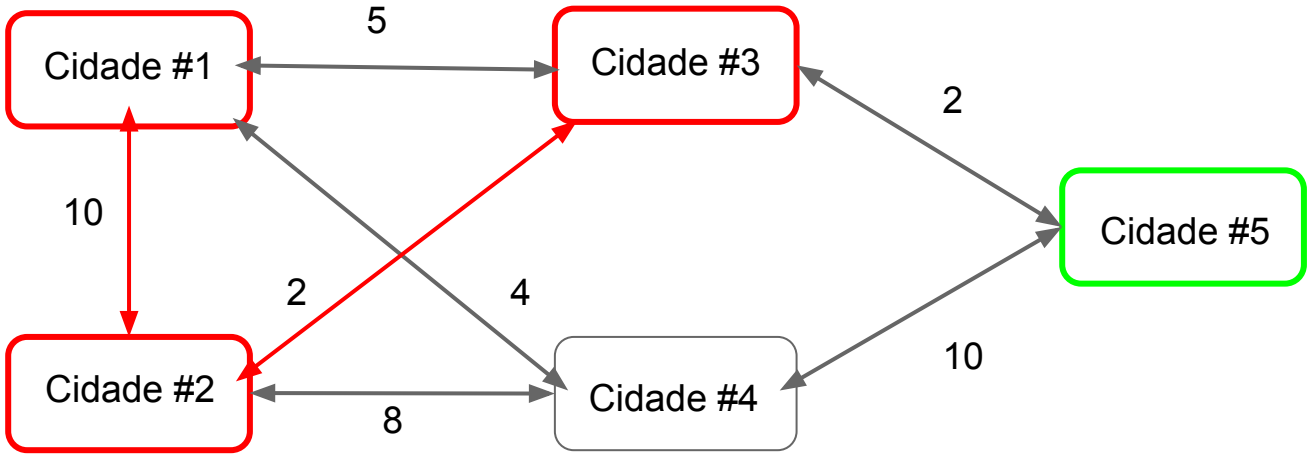
Exemplo de instância

Pares de precedência: (#2, #3), (#2, #4), (#3, #5)



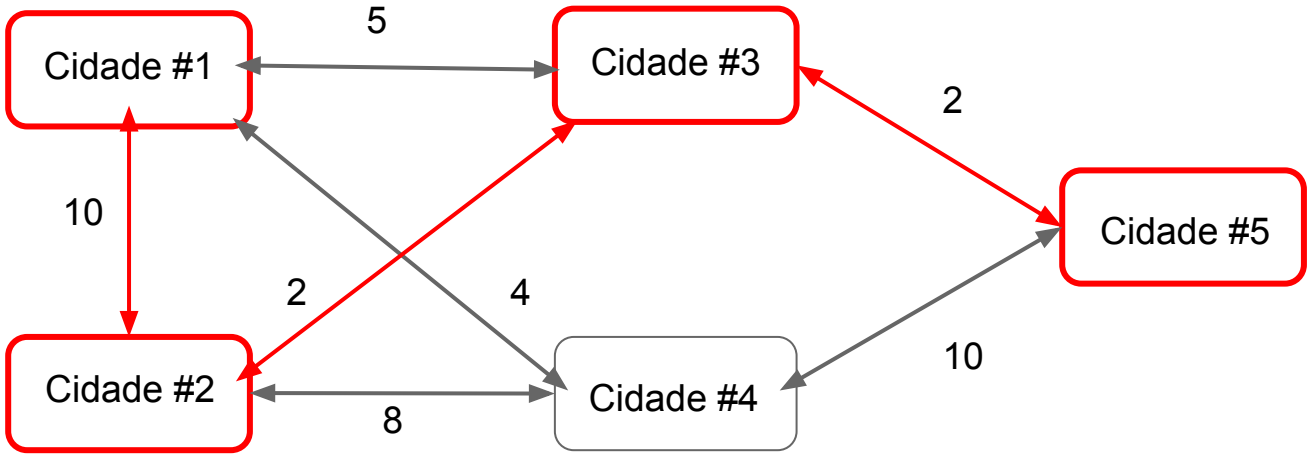
Exemplo de instância

Pares de precedência: (#2, #3), (#2, #4), (#3, #5)



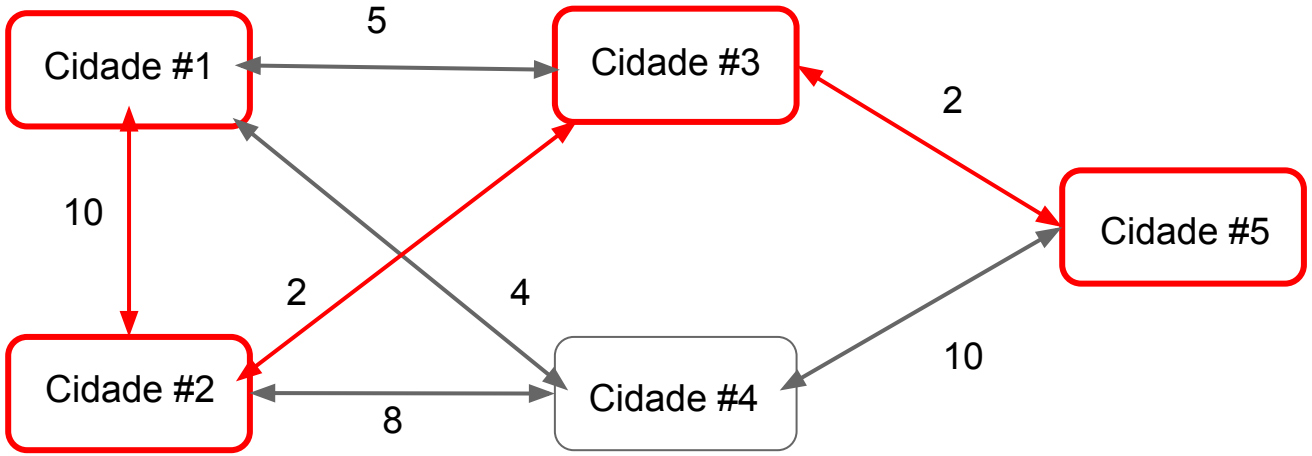
Exemplo de instância

Pares de precedência: (#2, #3), (#2, #4), (#3, #5)



Exemplo de instância

Caminho ótimo = 14



Pares de precedência: (#2, #3), (#2, #4), (#3, #5)

Exemplo de instância

Formulação matemática

Conjuntos

V conjunto de vértices (1 a n)

Parâmetros

c_{ij} $i \in V, j \in V$ custo do vértice i ao vértice j

$$p_{ij} \quad i \in V, j \in V \quad \begin{cases} 1, & \text{se vértice } i \text{ precede o vértice } j \\ 0, & \text{caso contrário} \end{cases}$$

Variáveis

$$x_{ij} \quad i \in V, j \in V \quad \begin{cases} 1, & \text{se arco } (i,j) \text{ presente na solução} \\ 0, & \text{caso contrário} \end{cases}$$

o_i $i \in V$ ordem de visitação $[0, n-1]$

Formulação matemática

Conjuntos

V conjunto de vértices (1 a n)

Parâmetros

c_{ij} $i \in V, j \in V$ custo do vértice i ao vértice j

p_{ij} $i \in V, j \in V$

$$\begin{cases} 1, & \text{se vértice } i \text{ precede o vértice } j \\ 0, & \text{caso contrário} \end{cases}$$

Variáveis

x_{ij} $i \in V, j \in V$

$$\begin{cases} 1, & \text{se arco } (i,j) \text{ presente na solução} \\ 0, & \text{caso contrário} \end{cases}$$

o_i $i \in V$ ordem de visitação $[0, n-1]$

Formulação matemática

Conjuntos

V conjunto de vértices (1 a n)

Parâmetros

c_{ij} $i \in V, j \in V$ custo do vértice i ao vértice j

p_{ij} $i \in V, j \in V$ $\begin{cases} 1, & \text{se vértice } i \text{ precede o vértice } j \\ 0, & \text{caso contrário} \end{cases}$

Variáveis

x_{ij} $i \in V, j \in V$ $\begin{cases} 1, & \text{se arco } (i,j) \text{ presente na solução} \\ 0, & \text{caso contrário} \end{cases}$

o_i $i \in V$ ordem de visitação $[0, n-1]$

Formulação matemática

Conjuntos

V conjunto de vértices (1 a n)

Parâmetros

c_{ij} $i \in V, j \in V$ custo do vértice i ao vértice j

p_{ij} $i \in V, j \in V$ $\begin{cases} 1, & \text{se vértice } i \text{ precede o vértice } j \\ 0, & \text{caso contrário} \end{cases}$

Variáveis

x_{ij} $i \in V, j \in V$ $\begin{cases} 1, & \text{se arco } (i,j) \text{ presente na solução} \\ 0, & \text{caso contrário} \end{cases}$

o_i $i \in V$ ordem de visitação $[0, n-1]$

Formulação matemática

Função objetivo

$$\min \sum_{i \in V} \sum_{j \in V - \{i\}} x_{ij} c_{ij}$$

Formulação matemática

Restrições

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \tag{2}$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \tag{3}$$

$$x_{n,j} = 0, \ \forall j \in V. \tag{4}$$

$$x_{i,1} = 0, \ \forall i \in V. \tag{5}$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \tag{6}$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \tag{7}$$

$$x_{i,j} \in \{0, 1\} \tag{8}$$

$$o_i \in R^+ \tag{9}$$

Formulação matemática

Restrições

Todos vértices, exceto o último, devem ter exatamente um arco de saída

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \quad (2)$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \quad (3)$$

$$x_{n,j} = 0, \ \forall j \in V. \quad (4)$$

$$x_{i,1} = 0, \ \forall i \in V. \quad (5)$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \quad (6)$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \quad (7)$$

$$x_{i,j} \in \{0, 1\} \quad (8)$$

$$o_i \in R^+ \quad (9)$$

Formulação matemática

Restrições

Todos vértices, exceto o primeiro, devem ter exatamente um arco de entrada

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \quad (2)$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \quad (3)$$

$$x_{n,j} = 0, \ \forall j \in V. \quad (4)$$

$$x_{i,1} = 0, \ \forall i \in V. \quad (5)$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \quad (6)$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \quad (7)$$

$$x_{i,j} \in \{0, 1\} \quad (8)$$

$$o_i \in R^+ \quad (9)$$

Formulação matemática

Restrições

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \quad (2)$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \quad (3)$$

$$x_{n,j} = 0, \ \forall j \in V. \quad (4)$$

$$x_{i,1} = 0, \ \forall i \in V. \quad (5)$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \quad (6)$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \quad (7)$$

$$x_{i,j} \in \{0, 1\} \quad (8)$$

$$o_i \in R^+ \quad (9)$$

Não há nenhum arco saindo do último vértice.

Não há nenhum arco chegando no primeiro vértice

Formulação matemática

Restrições

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \quad (2)$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \quad (3)$$

$$x_{n,j} = 0, \ \forall j \in V. \quad (4)$$

$$x_{i,1} = 0, \ \forall i \in V. \quad (5)$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \quad (6)$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \quad (7)$$

$$x_{i,j} \in \{0, 1\} \quad (8)$$

$$o_i \in R^+ \quad (9)$$

Garante que a ordem de precedência é obedecida

Formulação matemática

Restrições

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \quad (2)$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \quad (3)$$

$$x_{n,j} = 0, \ \forall j \in V. \quad (4)$$

$$x_{i,1} = 0, \ \forall i \in V. \quad (5)$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \quad (6)$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \quad (7)$$

$$x_{i,j} \in \{0, 1\} \quad (8)$$

$$o_i \in R^+ \quad (9)$$

Se aresta entre *i* e *j* é usada, *i* deve preceder *j*.

Formulação matemática

Restrições

$$\sum_{j \in V - \{i\}} x_{ij} = 1, \ \forall i \in V \setminus \{n\} \tag{2}$$

$$\sum_{i \in V - \{j\}} x_{ij} = 1, \ \forall j \in V \setminus \{1\} \tag{3}$$

$$x_{n,j} = 0, \ \forall j \in V. \tag{4}$$

$$x_{i,1} = 0, \ \forall i \in V. \tag{5}$$

$$o_i + 1 \leq o_j, \ \forall i, j \in V, \ i \neq j, \ p_{ij} = 1 \tag{6}$$

$$o_i + 1 \leq o_j - M * (x_{i,j} - 1), \ \forall i, j \in V, \ i \neq j. \tag{7}$$

$$x_{i,j} \in \{0, 1\} \tag{8}$$

$$o_i \in R^+ \tag{9}$$

Busca Tabu

- Objetivo:
  - Evitar que partes recentemente visitadas sejam retornadas, criando ciclos na solução.
- Lista Tabu armazena movimentos feitos até o presente momento.
- Serve como uma memória de curto prazo dos movimentos recentes dentro da área pesquisável.

Algoritmo proposto

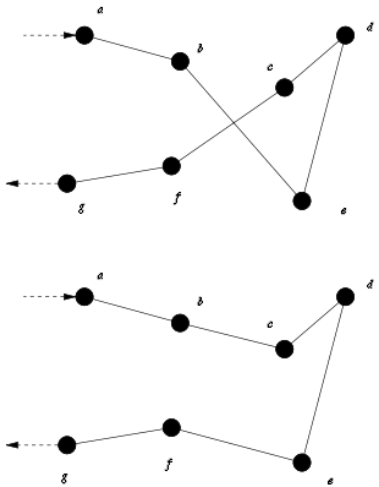
- Representação do problema e Estruturas de Dados
  - Grafo completo G = <V, A>
    - Uso de uma matriz de adjacência NxN, em que N = |V|
  - Lista Tabu
    - Vetores
  - Caminhos
    - Vetores

Algoritmo proposto

- Geração da solução inicial
  - Algoritmo guloso
  - Partindo do vértice inicial, escolhe a aresta de menor custo cujo vértice conectado (i) ainda não foi visitado e (ii) todos vértices que o precedem já foram visitados.
  - Uma vez escolhido o próximo vértice do caminho, o processo é repetido a partir desse vértice, até que não haja mais vértices para se escolher.

Algoritmo proposto

- Vizinhança e a estratégia para seleção dos vizinhos;
  - Busca local: 2-opt
  - Calcula o custo 2-opt para cada combinação de vizinhos de um vértice
  - Escolhe a combinação de menor custo e realiza a troca (best improvement)
  - Coloca na lista tabu o movimento realizado



Algoritmo proposto

- Parâmetro(s) do método, com os valores utilizados nos experimentos;
  - Tamanho lista Tabu
  - Critério de Aspiração
    - Movimentos na lista Tabu considerados válidos, com determinada probabilidade
  - Diversificação espaço de busca
    - Válidas trocas entre vértices mais distantes, quando nenhuma movimentação têm solução melhor

Algoritmo proposto

- Parâmetro(s) do método, com os valores utilizados nos experimentos;

Tabela 1. Variação dos parâmetros

Parâmetro	Variações
Tamanho lista Tabu	N/2, **N/4**, N/8
Critério de aspiração	50%, **20%**, 10%
Diversificar espaço de busca	**3**, 5, 6

Algoritmo proposto

- Critério de parada do algoritmo.
  - Número de iterações sem encontrar melhora na solução.
  - Cada busca local decresce o contador de iterações
  - Quando acha uma solução melhor, o contador de iterações é reiniciado

Ambiente de avaliação

- Intel i5-4570 CPU @ 3.20GHz
- 32 Gb de RAM
- SO: OS X
- C++
- Compilador LLVM version 8.0.0 (clang-800.0.42.1)

Resultados

1. Valor da melhor solução encontrada pelo GLPK
2. Tempo de execução do GLPK;
3. Valor médio da solução inicial do seu algoritmo;
4. Valor médio da melhor solução;
5. Desvio padrão das melhores soluções;
6. Tempo de execução médio (em segundos) do algoritmo;
7. Desvio percentual médio das soluções;

Análise de resultados obtidos

Tabela 2. Tabela de resultados - 10 iterações

Instância	1	2	3	4	5	6	7
ESC07	2125	0	2700	2550.0	0.0	0.00	20.00
ESC12	1675	1.2	2034	1751.0	0.0	0.01	4.53
ESC25	1681	129.6	3360	3360.0	0.0	0.08	99.88
ESC47	3152	3600	3843	3553.0	0.0	0.85	175.85
ESC78	–	3600	22600	22120.0	430.0	6.74	21.33
ft70.1	–	3600	45255	44459.0	0.0	3.14	13.08
prob.100	–	3600	2921	2755.0	0.0	10.84	136.88
rbg109a	–	3600	1443	1330.4	4.8	122.46	28.16
rbg150a	–	3600	2168	2079.8	1.8	417.97	18.84
rbg174a	–	3600	2444	2295.7	0.9	1009.22	12.92

Análise de resultados obtidos

Tabela 3. Tabela de resultados - 50 iterações

Instância	1	2	3	4	5	6	7
ESC07	2125	0	2700	2550.0	0.0	0.01	20.00
ESC12	1675	1.2	2034	1743.4	22.8	0.06	4.08
ESC25	1681	129.6	3360	3179.8	143.1	0.65	89.16
ESC47	3152	3600	3843	3553.0	0.0	3.61	175.85
ESC78	–	3600	22600	21812.0	519.0	27.82	19.64
ft70.1	–	3600	45255	44459.0	0.0	12.26	13.08
prob.100	–	3600	2921	2755.0	0.0	42.15	136.88
rbg109a	–	3600	–	–	–	120.00	–
rbg150a	–	3600	–	–	–	120.00	–
rbg174a	–	3600	–	–	–	120.00	–

Conclusões

- Nosso algoritmo conseguiu bons resultados para instâncias pequenas e com um alto número de iterações
- Para instâncias grandes, o algoritmo se torna muito demorado mesmo com poucas iterações
- É difícil encontrar os parâmetros certos e bons
- A aleatoriedade influencia pouco nos resultados da busca tabu