Utilizando Recozimento Simulado e NPE para resolver o problema da Árvore Geradora Mínima com Restrição de Grau

José Flávio Lopes

Fevereiro 2023

Abstract

Este artigo discute o uso do recozimento simulado para resolver o problema DC-MST. As soluções candidatas foram representadas usando a estrutura NPE, com os operadores SPRN e TBRN usados para gerar novas soluções durante o processo de busca. O operador SPRN gerou novas soluções mudando, aleatoriamente, o local de uma subárvore da solução atual, enquanto o operador TBRN gerou novas soluções trocando, além do local, também a raíz de uma subárvore da solução atual.

1 Introdução

Dado um grafo $G(V, E)$ conectado, não-direcionado e ponderado, com V sendo o conjunto de vértices e E o conjunto de arestas; o problema da Árvore Geradora Mínima com Restrição de Grau (DC-MST) consiste em encontrar um árvore geradora T de G, na qual os vértices possuam grau menor ou igual a um valor D específico. Além disso, a soma do peso das aresta de T precisa ser mínima.

O DC-MST é um problema considerado NP-Completo e isto foi provado em (Lovász and Plummer 1986), o que significa que não há uma solução exata para ele que execute em tempo polinomial.

A estrutura do restante deste artigo é a seguinte: na Seção 2, é apresentado o NPE, juntamente com seus operadores; na Seção 3, a metaheurística proposta; na Seção 4, avaliamos o desempenho da heurística proposta, comparando-a com heurísticas de outros autores; finalmente, na Seção 5, apresentamos as conclusões deste estudo.

2 NPE

O NPE (Node-depth Phylogenetic-Based Encoding) é uma representação de árvores, comumente utilizada em algoritmos evolucionários. A representação NPE de uma árvore consiste em duas listas, N e D, onde N_x é um vértice e D_x é a profundidade do vértice na árvore. Há mais sobre o NPE em (Lima et al. 2016) e em (Carvalho and Ribeiro 2019).Dois operadores de mutação foram utilizados, e serão apresentados na seção 2.2 e seção 2.3, sendo, respectivamente, o operador SPRN e o operador TBRN. Na seção 2.1, o algoritmo de decodificação do NPE é apresentado.

2.1 Algoritmo de Decodificação

O algoritmo 1 faz a decodificação do NPE para um conjunto de arestas.

Algorithm 1 Pseudocódigo para a decodificação do NPE

```
//Seja N o NPE, com N_i send o par (n_i, d_i) na posicao i de N
//Seja |N| o tamanho de N
se |N| == 0:
  retorna nulo
fimse
Arvore ← {}
raizes_0 ← n_0
j ← 1
enquanto j < |N|:
   k ← raizes_{d_j − 1}
   Arvore ← Arvore ∪ (k, n_j)
   raizes_{dj} ← n_j
   j ← j + 1
fimenquanto
retorna Arvore
```

O arranjo *raizes* se encarrega de armazenar o ultimo vértice encontrado em cada profundidade, de forma que $raizes_i = x$ indica que o último vértice encontrado na profundidade i foi o x.

2.2 Operador SPRN

O operador SPRN (Subtree Pruning and Regrafting Operator on NPE - SPRN) modifica o a árvore ao escolher uma subárvore e realoca-lá para uma outra subárvore; para tanto, o procedimento necessita da escolha de um vértice p, que é a raiz da subárvore a ser realocada, um intervalo rp, que se trata da sequência no NPE dos vértices contidos na subárvore com raiz em p; além disso, também é necessária a escolha de um vértice a, que seja adjacente de p e não esteja contido em rp; a será a nova raiz da subárvore sendo realocada.

Algorithm 2 Pseudocódigo para o SPRN

```
// Seja N o NPE e |N| o tamanho do NPE
// Seja ip o índice do vértice p em N, com 0 < ip < |N|
fp ← ip + 1
enquanto (fp < |N|) && (d_fp > d_ip):
   fp ←fp + 1
fimenquanto
// Seja rp o intervalo [ip, fp)
// Seja ia o indice do vértice a em N, ia ∉ rp
N' ← {}
D' ← {}
para cada i ∈ [0, ia]:
   se i ∉ rp:
     N' ← N' ∪ n_i
     D' ← D' ∪ d_i
   fimse
fimpara
para cada i ∈ rp:
   N' ← N' ∪ n_i
   D' ← D' ∪ d_i − d_ip + d_ia + 1 // recalculando a profundidade do vértice em n_i
fimpara
para cada i ∈ [ia + 1, |N|):
   se i ∉ rp:
     N' ← N' ∪ n_i
     D' ← D' ∪ d_i
   fimse
fimpara
retorna (N', D')
```

2.3 Operador TBRN

O operador TBRN modifica a árvore ao escolher uma subárvore com raíz em um vértice p, mudar a raiz dessa subárvore para algum vértice r que está contido na própria subárvore, e então realocar a subárvore modificada para alguma outra subárovre com raíz em algum vértice a, tal que a seja adjacente de r e não esteja na subárvore que está sendo realocada. O algoritmo 3 apresenta um pseudocódigo para o TBRN.

Algorithm 3 Pseudocódigo para o TBRN

// Seja N a lista de vértices do NPE, com $|N|$ sendo seu tamanho
// Seja D a lista de profundidades do NPE, tal que o vértice N_i está na profundidade D_i na árvore
// Seja ip o índice do vértice p em N, com $0 < ip < |N|$
$fp \leftarrow ip + 1$
enquanto $(fp < |N|)$ && $(D_{fp} > D_{ip})$:
  $fp \leftarrow fp + 1$
fimenquanto
// Seja rp o intervalo $[ip, fp)$
// Seja ir o índice do vértice r em N, tal que $ir \in rp$
$fr \leftarrow ir + 1$
enquanto $(fr < |N|)$ && $(D_{fr} > D_{ir})$:
  $fr \leftarrow fr + 1$
fimenquanto
// Seja rr o intervalo $[ir, fr)$
// Seja ia o índice do vértice a em N, tal que a é adjacente de r e $ia \notin rp$
$Ntmp \leftarrow \{\}$
$Dtmp \leftarrow \{\}$
para cada $i \in rr$:
  $Ntmp \leftarrow N_i$
  $Dtmp \leftarrow D_i - D_{ir} + D_{ia} + 1$
fimpara
$N' \leftarrow \{\}$
$D' \leftarrow \{\}$
para cada $i \in [0, ia]$:
  se $i \notin rp$
    $N' \leftarrow N' \cup N_i$
    $D' \leftarrow D' \cup D_i$
  fimse
fimpara
$ix \leftarrow ir$
enquanto $ix \neq ip$:
  $i \leftarrow ix - 1$
  enquanto $i \geq ip$:
    se $D_i == D_{ix} - 1$:
      break
    fimse
    $i \leftarrow i - 1$
  fimenquanto
  $fx \leftarrow i + 1$
  enquanto $(fx < |N|)$ && $(D_{fx} > D_i$:
    $fx \leftarrow fx + 1$
  fimenquanto
  // Seja aux_ix o índice para qual $N_{ix} == Ntmp_{aux\_ix}$
  para cada $j \in (i, fx]$:
    se $j \notin rr$:
      $Ntmp \leftarrow Ntmp \cup N_j$
      $Dtmp \leftarrow Dtmp \cup D_j - D_i + Dtmp_{aux\_ix} + 1$
    fimse
  fimpara
  $ix \leftarrow i$
fimenquanto
$N' \leftarrow N' \cup Ntmp$
$D' \leftarrow D' \cup Dtmp$
para para $i \in (ia + 1, |N|]$:
  se $i \notin rp$:
    $N' \leftarrow N' \cup N_i$
    $D' \leftarrow D' \cup D_i$
  fimse
fimpara
retorna (N', D')

2.4 Escolha dos vértices

O operador SPRN necessita da escolha de 2 vértices do grafo para sua execução, sendo um deles o vértice p, que é a raiz da subárvore a ser realocada, e o outro sendo o vértice a, que é a raiz para onde a subárvore será realocada; para que a operação ocorra sem erros, o vértice p não pode ser a raiz da árvore sendo representada pelo NPE, ou seja, o vértice p não pode estar no índice 0 da lista de vértices do NPE; em relação ao vértice a, é necessário que ele não esteja contido na subárvore que tem p como raiz, e que a e e sejam adjacentes, ou seja, $(e, a) \in G.E$. Dito isso, o vértice p pode ser escolhido aleatoriamente, sendo necessária apenas a verificação de sua posição na lista N; o vértice a também pode ser escolhido aleatoriamente, mas para ele há duas verificações: a não pode estar contido na subárvore que tem p como raiz e a deve ser adjacente a p.

Já o operador TBRN necessita da escolha de 3 vértices para realizar suas ações; o vértice p segue os mesmos requisitos que o vértice p do operador SPRN; os outros dois vértices são o r e o a: o vértice r necessita estar contido na subárvore que possui p como raiz, pois essa subárvore agora terá r como sua nova raiz; o vértice a, da mesma forma que o vértice a do operador SPRN, não deve estar contido na subárvore que possui p como raiz, e deve ser adjacente a r. Note que após modificar a subárvore para que sua raiz agora seja o vértice r, a operação subsequente é idêntica ao SPRN.

3 Recozimento Simulado

O recozimento simulado é uma técnica de otimização que visa a melhorar uma solução através de pequenas perturbações (mutações) em sua forma atual. O resultado pós perturbação pode ser aceito ou rejeitado, dependendo de algum critério de aceitação; é comum que um resultado com avaliação inferior ao da solução pré perturbação seja aceito, com o objetivo de escapar de mínimos ou máximos locais. Há mais sobre o recozimento simulado em (Kirkpatrick, Gelatt Jr, and Vecchi 1983)

3.1 Função de avaliação

A forma proposta para avaliar uma solução é baseada no algoritmo de decodificação apresentado na seção 2, com algumas modificações. O Algoritmo 4 apresenta o pseudocódigo com tais modificações.

Algorithm 4 Pseudocódigo para a função de avaliação

//Seja N o NPE, com N_i send o par (n_i, d_i) na posicao i de N
//Seja $|N|$ o tamanho de N
//Seja $grau$ o grau máximo permitido para todo vértice
//Seja $listaGrau$ a lista de grau para cada vértice, de modo que $listaGrau_i$ seja o grau do vértice i
//Seja w a função que retorna o peso da aresta (x, y), para todo $(x, y) \in G.E$
se $|N| == 0$:
  retorna $NULL$
fimse
$custo \leftarrow 0$
$penalizacao \leftarrow 0$
$raizes_0 \leftarrow n_0$
$j \leftarrow 1$
enquanto $j < |N|$:
  $k \leftarrow raizes_{d_j - 1}$
  $listaGrau_k \leftarrow listaGrau_k + 1$
  $listaGrau_{n_j} \leftarrow listaGrau_{n_j} + 1$
  se $(listaGrau_k > grau)$ ||| $(listaGrau_{n_j} > grau)$:
    $penalizacao \leftarrow penalizacao + 1$
  fimse
  $custo \leftarrow custo + w(k, n_j) + penalizacao * custo$
  $raizes_{dj} \leftarrow n_j$
  $j \leftarrow j + 1$
fimenquanto
retorna $custo$

3.2 Metaheurística proposta

O Algoritmo 5 ilustra um pseudocódigo para a metaheurística proposta.

Algorithm 5 Pseudocódigo para o recozimento simulado

```
//Seja x0 a solução inicial
//Seja temp a temperatura do sistema
//Seja cf o fator de resfriamento
//Seja f a função que avalia uma solução
x_atual ← x0
f_atual ← f(x_atual)
T ← temp
x_melhor ← x_atual
f_melhor ← f_atual
enquanto criterio_de_parada == FALSO:
   x_novo ← x_atual.copia()
   // aplicar SPRN e/ou TBRN em x_novo
   f_novo ← f(x_novo)
   p_aceita ← exp(−(f_novo − f_atual)/T)
   se aleatorio < p_aceita:
      x_atual ← x_novo
      f_atual ← f_novo
   fimse
   se f_atual < f_melhor:
      x_melhor ← x_atual
      f_melhor ← f_atual
   fimse
   T ← T * cf
fimenquanto
retorna (x_melhor, f_melhor)
```

O algorimo recebe como entrada uma solução inicial, que é uma árvore geradora codificada como um NPE. Para as perturbações na solução, os operadores SPRN e TBRN são utilizados. A decisão de qual operador utilizador pode ser tomada de maneira empírica. Para escapar de mínimos ou máximos locais, há uma probabilidade de aceitação para soluções que pioraram após serem perturbadas.

4 Resultados

Os testes foram feitos em um computador com 8gb de RAM, com um processador Intel Core i5-7200U @ 2.5GHz x 4. Várias instâncias foram testadas, e nesta seção serão apresentados alguns resultados para grafos completos com 500, 700 e 900 vértices. Foram executados testes com 1000 iterações, 5000 iterações e 15000 iterações.

4.1 Gráficos

Os gráficos a seguir foram obtidos com o parâmetro de grau máximo igual a 5.

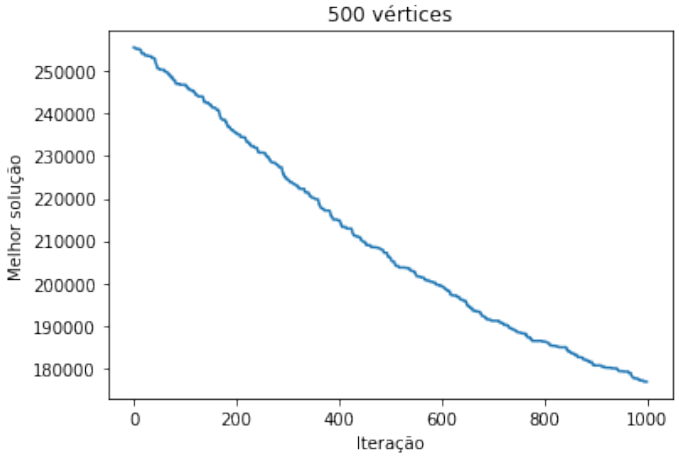

Figure 1: Grafo completo com 500 vértices e 1000 iterações

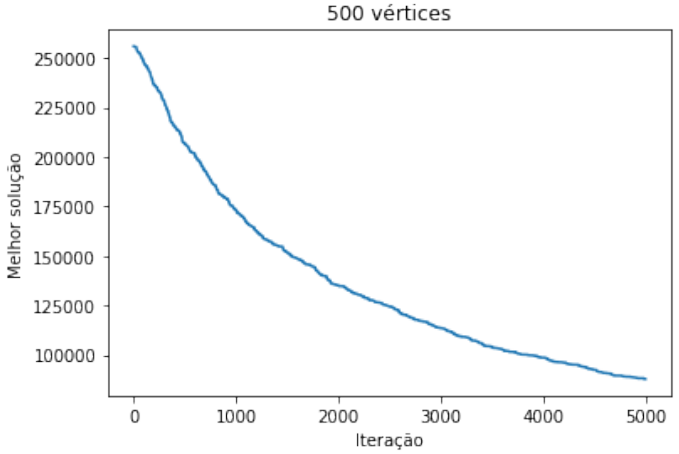

Figure 2: Grafo completo com 500 vértices e 5000 iterações

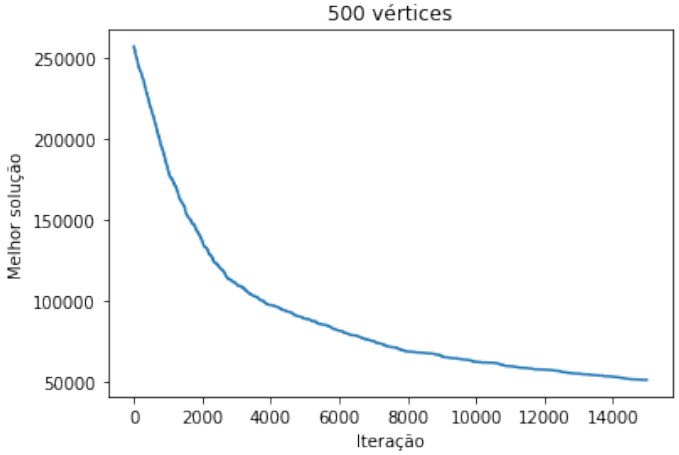

Figure 3: Grafo completo com 500 vértices e 15000 iterações

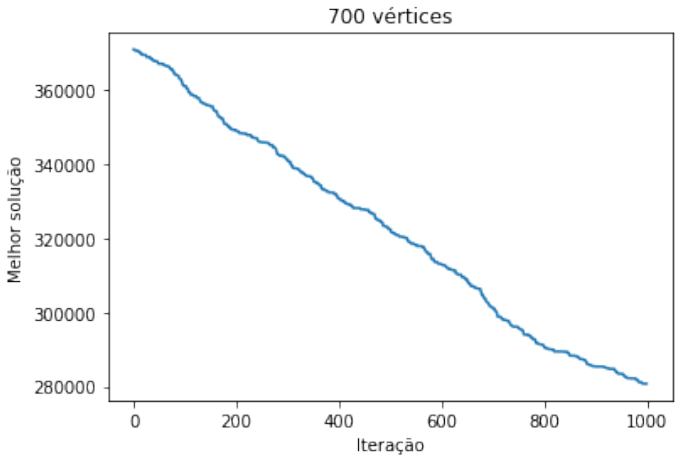

Figure 4: Grafo completo com 700 vértices e 1000 iterações

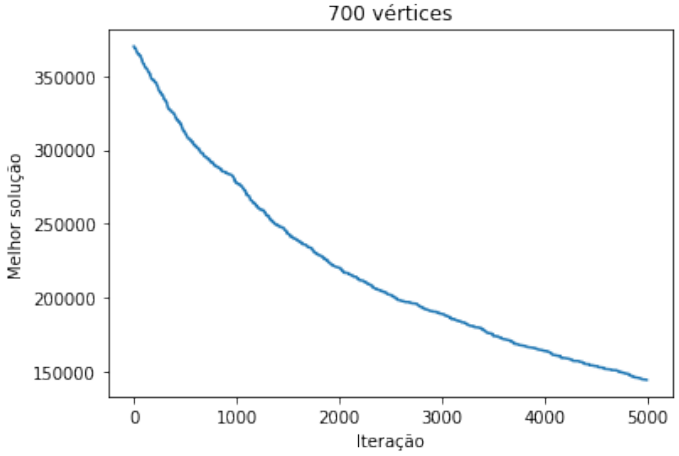

Figure 5: Grafo completo com 700 vértices e 5000 iterações

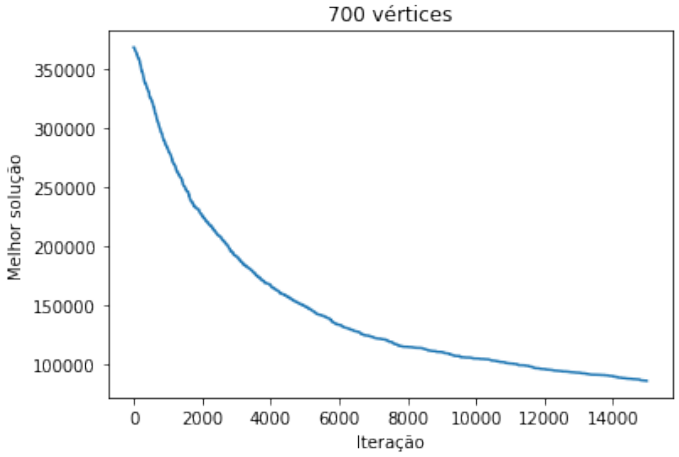

Figure 6: Grafo completo com 700 vértices e 15000 iterações

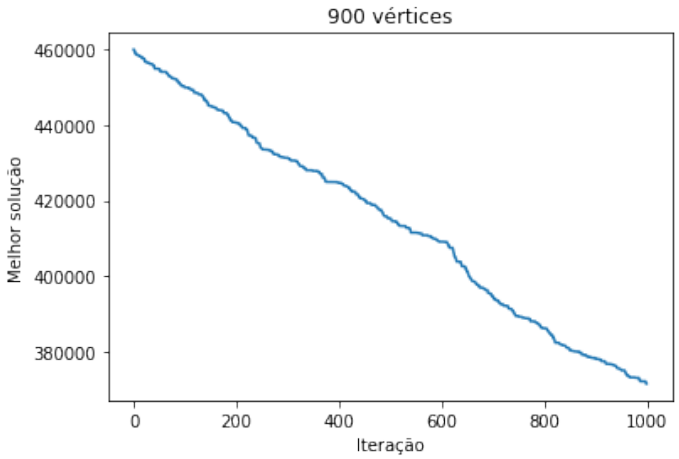

Figure 7: Grafo completo com 900 vértices e 1000 iterações

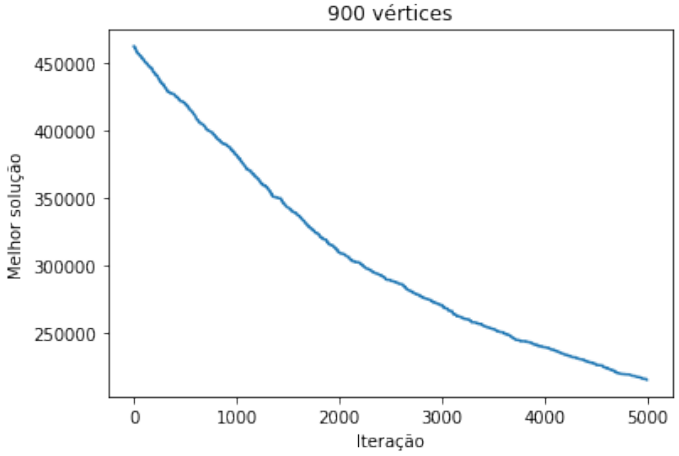

Figure 8: Grafo completo com 900 vértices e 5000 iterações

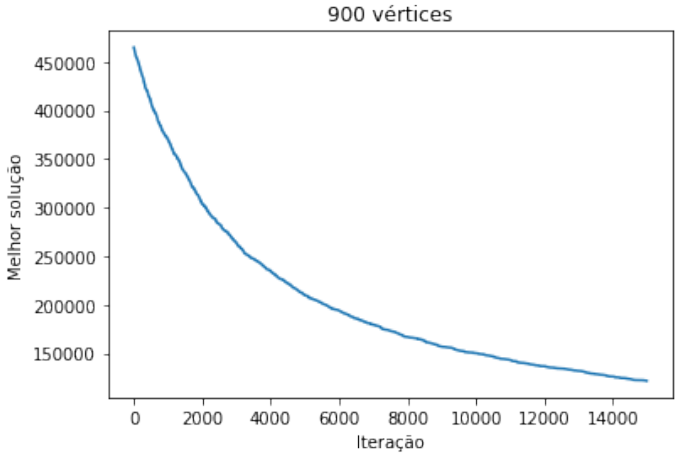

Figure 9: Grafo completo com 900 vértices e 15000 iterações

4.2 Comparação

Os resultados a seguir foram feitos em comparação com uma heurística baseada na proposta feita por (Singh and Sundar 2020).

Table 1: Resultados medidos por número de iterações

Heurística	**v = 500, d = 5, iter = 1000**	**v = 700, d = 5, iter = 1000**	**v = 900, d = 5, iter = 1000**
Recozimento	176850.76	280743.69	371362.84
Hybrid GA	238413.81	339889.63	443304.16

Table 2: Resultados medidos por tempo (30s)

Heurística	**v = 500, d = 5, tempo = 30s**	**v = 700, d = 5, tempo = 30s**	**v = 900, d = 5, tempo =**
Recozimento	29119.54	53253.70	95373.31
Hybrid GA	240453.72	349266.37	452767.86

Table 3: Resultados medidos por tempo (60s)

Heurística	**v = 500, d = 5, tempo = 60s**	**v = 700, d = 5, tempo = 60s**	**v = 900, d = 5, tempo =**
Recozimento	23131.65	42482.78	64864.06
Hybrid GA	229721.26	342425.14	450629.17

5 Conclusão

Este artigo prôpos o uso da metaheurística "Recozimento Simulado" para a resolução do problema da Árvore Geradora Mínima com Restrição de Grau. Para representar as soluções, a estrutura NPE foi utilizada e, embora esta seja comumente empregada em algoritmos evolucionários, o fato de possuir dois operadores com ótima diversificação serviu muito bem para seu uso na abordagem proposta. Os resultados apresentados mostram que as técnicas utilizadas não se prenderam em mínimos locais, sempre melhorando a função objetivo ao obter mais iterções ou tempo de execução.

References

Carvalho, Iago and Marco Ribeiro (Nov. 2019). "A Node-depth Phylogenetic-based Artificial Immune System for Multi-objective Network Design Problems". In: *Swarm and Evolutionary Computation* 50, p. 100491. DOI: 10.1016/j.swevo.2019.01.007.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi (1983). "Optimization by simulated annealing". In: *Science* 220.4598, pp. 671–680.

Lima, Telma et al. (May 2016). "Node-depth Phylogenetic-Based Encoding, a Spanning-Tree Representation for Evolutionary Algorithms. Part I: Proposal and Properties Analysis". In: *Swarm and Evolutionary Computation* 31. DOI: 10.1016/j.swevo.2016.05.001.

Lovász, L. and M.D. Plummer (1986). "Matching theory". In: *North-Holland Mathematics Studies* 121, pp. 1–45.

Singh, Kavita and Shyam Sundar (2020). "A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem". In: *Soft Computing* 24.3, pp. 2169–2186.