Using Markov chains and Monte Carlo simulations to solve Minimum Diameter Spanning Tree Problem

Amanda Ferreira de Azevedo[1], **Wanderson Douglas Lomenha Pereira**[1]
https://github.com/afazevedo/project_mcmc

[1] Universidade Federal do Rio de Janeiro
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia
Programa de Engenharia de Sistemas e Computação

{afazevedo,wlomenha}@cos.ufrj.br

Abstract. *Given an undirected graph G=(V, E), we investigate the Budget Minimum Diameter Spanning Tree Problem, seeking the minimum diameter spanning tree of G. The diameter is given by the number of edges in the most extensive path of a tree. Moreover, costs are associated with all the edges of G, and the sum of the edge costs of this tree may not exceed a given budget value. Thus, feasible trees must necessarily be spanning. Barely investigated in the literature, these problems have much application, mainly in the telecommunications area. Furthermore, in the literature it has been proven to be an NP-Hard problem. Our approach uses the simulated annealing algorithm that solves the problem through a Markov chain simulation. This chain induces a lazy random walk between spanning trees, and the probability transition to each neighbor is uniformly equal. The results show good approximations when compared to the optimal solution. Thus, to our knowledge, is the first use of random heuristics algorithms proposed for this problem. Therefore, we show that the chain takes many transitions to converge, and the algorithm performance is related to the sparseness of G.*

1. Problem definition and literature survey

Let $G = (V, E)$ be a finite undirected connected graph with a set V of vertices and a set E of edges. Assume that a cost c_e is associated to each edge $e = \{i, j\} \in E$. A *tree* $T = (V_T, E_T)$, with $V_T \subseteq V$ and $E_T \subseteq E$, is a connected and acyclic subgraph of G. A *spanning tree* is a tree that contains all the vertices of the graph, i.e., when $V_T = V$. Among all the spanning trees of G, the one with the lowest cost is the optimal solution to the *Minimum Spanning Tree Problem* (MST).

A *path* in G is called *simple* when it does not visit the same node more than once. Denote by d_{ij} the length of the shortest simple path connecting the vertices $i, j \in V$, i.e., the *distance* between them. Finally, the largest distance between any pair of nodes in G is called the *diameter* of G. Let B be an upper limit for the total cost of a given tree edges. Thus, the *Budget Minimum Diameter Spanning Tree Problem* (BDSTP) aims to find a spanning tree T with minimum diameter, where $\sum_{e \in E} c_e \leq B$.

BMDSTP was introduced by [Plesnik 1981], who proved to be an NP-Hard problem. As far as we know, there are no non-exact algorithms that solve it in the literature. The first exact algorithms, however, were proposed this year by [Amanda Azevedo 2021]. When all the costs c_e are all equal to 1 and $B = |V| - 1$, the problem is equivalent to *1-center problem* (see [Hakimi et al. 1978] for details). Thus, under these conditions exists a polynomial-time algorithm that solves it despite the instance size. Therefore, the budget constraint becomes redundant, and the problem could just be named *Minimum Diameter Spanning Tree Problem* (MDSTP). For example, consider the graph of the states of Brazil shown in Figure 1.

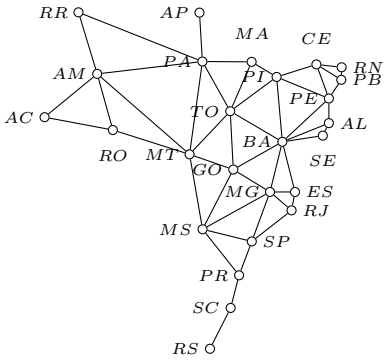

Figure 1. Graph of the states of Brazil

An illustration of a comparison between the optimal spanning tree of MDSTP, MST, and BMDSTP is given in Figure 2.

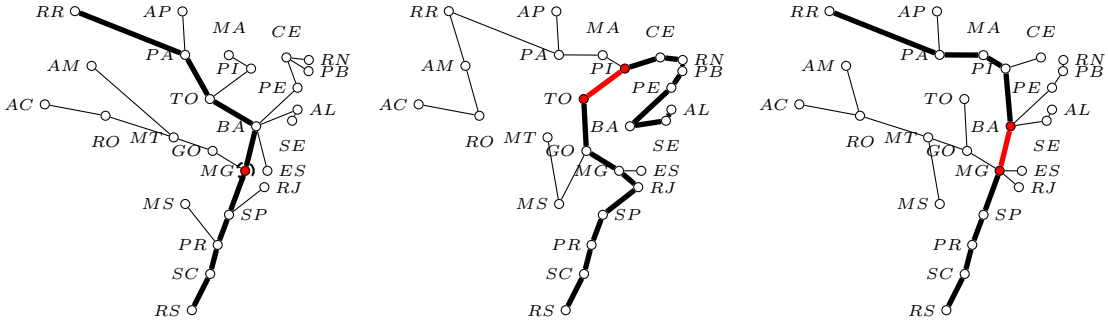

Figure 2. Comparison between the optimal spanning tree of MDSTP, MST, and BMDSTP respectively

2. Methodology

The *simulated annealing algorithm* is a meta-heuristic founded in the model of [Metropolis et al. 1953], which was adapted in the computational context by [Kirkpatrick et al. 1983, Černý 1985]. Moreover, is a *Monte Carlo* technique inspired by an analogy with annealing, a physical phenomenon of thermodynamics. *Annealing* consists of raising the temperature of the metal by injecting energy into the structure and then slowly cooling the metal. Thus, the general idea of the algorithm is to solve a minimization problem in analogy to this heating and cooling scheme.

To decide whether a given movement is accepted or not, a probabilistic process controlled by [Hastings 1970] technique is used. One of the essential parameters for this process is the temperature, which controls the balance between intensification and diversification of the algorithm. When a new randomly generated solution has a cost worse than the current solution, the solution is accepted by a simplification of the probability distribution proposed by Maxwell-Boltzmann, and later simplified by Metropolis-Hastings [Maxwell 1860]. Thus, the probability of an energy variation $\Delta(s)$ of a transition between configurations in this distribution is given by:

$$P(\Delta(s)) = e^{-\frac{\Delta(s)}{T}} \tag{1}$$

If we are in a minimization problem, when $\Delta(s) < 0$, accepts with probability 1 uniformly between neighbors better than s. Additionally, when $\Delta(s) > 0$, accept with probability given by equation 1. Therefore, when $\Delta = 0$, the acceptance is indifferent, since there is no change.

2.1. Simulated Annealing and Markov Chains

Let S be a finite set of states in a Markov Chain with a function $f : S \to \mathbb{R}$ associated to them. Formally, the general optimization problem in this context aim to find a state $s \in S$ that minimizes $f(s)$. Intuitively, let us take the inverse of f as the stationary distribution of this chain and only accept states that provides improvement. In a sufficiently long time, we are likely to end up in a state s where $f(s)$ is minimum. Using this strategy, however, we can end up at a local minimum. Then, if we control this "greed" with randomness and gradually increase it, we can highly get closer to the optimal solution. Hence, if we create a *family of Markov chains*, each of which with an increasingly greedy stationary distribution, we can get even more close to the optimum. Thus, we will use the so-called *Boltzmann* distribution to deal with this phenomenon, using the cooling idea seen in the last section. Therefore, we will create a Markov chain using Boltzmann distribution as its stationary distribution, as follows:

$$\pi_s = \frac{e^{\frac{-f(s)}{T}}}{\sum_{s \in S} e^{\frac{-f(s)}{T}}}$$

For a fixed and non-negative T, the smaller $f(s)$, the greater the stationary distribution. Thus, smaller values of $f(s)$ have exponentially larger probability. The guarantee that in a certain number of steps this chain will converge to the optimal solution is given by the following theorem:

Theorem: Let S be a finite set and let $f : S \to \mathbb{R}$ be arbitrary. For $T > 0$, let $\alpha(T)$ denote the probability that a random element s' chosen according to the *Boltzmann* distribution π_s satisfies: $f(s') = min_{s \in S} f(s)$ then $\lim_{T \to 0} \alpha(T) = 1$. Proof: See [Häggström 2002].

Hence, the Markov chain using π_s as a stationary distribution and decreasing T, increases the chances of choosing a minimum. The choice of how many iterations on each temperature should be done is called the annealing schedule, and their choice is crucial since we know that converges "sufficiently slowly". As [Häggström 2002] said, *"The choice of annealing schedule in practice is therefore a highly delicate balance: On one hand, we want it to be fast enough to get convergence in reasonable time, and on the other hand, we want it to be slow enough to avoid converging to an element which is not an f-minimizer."*

2.2. Our work

Let $S = \{s_1, s_2, \dots\}$ be a state space. Every state $s_i \in S$ is a spanning tree of a given graph $G = (V, E)$. The other main ingredients are two functions, which we call the *diameter function* and *cost function*. The diameter function, $f : S \to \mathbb{Z}$, is a function from the state space S to any integer value in $\mathbb{Z}$, which we use to calculate the diameter of given tree. Moreover, the cost function $g : S \to \mathbb{R}$ is a function from the state space S to any real value, which we use to calculate the total cost of a given tree. The transitions between any spanning trees is given by the algorithm named as *cycle transition*, described below:

1. Uniformly choose an edge in $E(G) - E(T)$ where T is the current tree;
2. Uniformly choose an edge in the cycle and remove;
3. Return T.

Consequently, the number of possible neighbors of each state is given by $|C| \cdot (m - n + 1)$, where $|C|$ is the size of the imposed cycle, and n, m is the number of nodes and edges, respectively. If the chosen edge was the same added edge, we stay in the same position, inducing a self-loop. Moreover, if G is a complete graph with n vertices,

the total number of spanning trees is $n^{(n-2)}$, namely *Cayley's* formula. Otherwise, if G is not a complete graph, the number of spanning trees can be calculated by *Kirchhoff's Theorem*. For example, applying to the graph of the states of Brazil given as an example, the total number of states in our chain would be 5310501721.

To generate an initial solution for the chain, we use an algorithm proposed by [Broder 1989], named **generate**, described as follows:

1. Simulate a random walk in G starting at s until every node is visited, storing the first visit to each node.
2. To each node $i \in V - s$, we take the edge $\{j, i\}$ and we add to the tree, T, where j corresponds the first visit to i.
3. Return T

The algorithm complexity is $O(n \log n)$ in most graphs and in the worst case, however, is $O(n^3)$.

To manage the budget constraint, we combined the simulated annealing algorithm with a local search. This approach is reported in several works, like [Martin and Otto 1996]. These works show a visible improvement in results when used simultaneously with the annealing process.

We implemented a local search that changes nodes or subtrees positions to reduce the cost, trying to make a current solution viable. The pseudo-code is described below.

Algorithm 1 Local Search

procedure LS(s')
  $level = 1$
  if total cost $\leq B$ **then**
    Stop ▷ Viable Solution
  else
    Identify the node or center edge of the tree, s'
    Put a label on each node of s' corresponding to its distance from the center
    Create a list of nodes, *LCR*, where the label of i is equal to level
    for $i \in LCR$ **do**
      Identify i's neighbors and take the one, v, that has a label smaller
      Choose a node j such that $c_{vi} > c_{ji}$, $min\{c_{ji} - c_{vi}\}$, with $i \neq j$, and the label j smaller or equal to i
      Add the edge $\{j, i\}$ and remove $\{v, i\}$ from s'
      Update the cost adding $c_{ji} - c_{vi}$
    end for
    if $level = \frac{n}{2}$ **then**
      if not improve **then** ▷ that is, made no move
        Stop
      else
        Back to step 2
      end if
    else
      Do $level = level + 1$ and back to step 3
    end if
  end if
  return s'
end procedure

Note that it induces a new chain that transitions are a combine of cycle transition algorithm and the local search. This new chain, however, will have more neighbors than the previous one. In the next section, we will demonstrate that this chain is irreducible.

2.3. Irreducibility

To force all states to be visited, we will make local search be used according to Boltzmann distribution. Consequently, we will do many local searches at the beginning of the annealing process, and as the temperature drops, we will transition using mainly the cycle transition.

We say that state s_i intercommunicates with state s_j if there is a path from s_i to s_j for some $t > 0$. If s_i intercommunicates with s_j then $s_i \rightarrow s_j$. Note that if s_i intercommunicates with s_j and s_j intercommunicates with s_i, then $s_i \leftrightarrow s_j$. If this condition is valid for all states then the chain is called *irreducible*.

We need to ensure that our chain is irreducible to guarantee that every state can be visited and possibly reach the optimal solution. Suppose we go from state s_i to state s_j. Note that the state s_j is a spanning tree that differs from the state s_i by at least one edge when performing the local search. Furthermore, the state s_j will cost less than or equal to the state s_i. Thus, it will rarely return from state s_j to state s_i through a single transition. However, it is possible to return from state s_j to state s_i through a path with positive probability. The reason is that we use the Boltzmann distribution to decide if we use local search or not. Consequently, there is a non-zero probability that the local search will not be used on the path between s_i and s_j. Hence, by cycle transition, the annealing process can allow transitions between states with a diameter and cost worse than the current state. Therefore, our CM is irreducible. ■

2.4. Cooling schedule

This section will show the strategy adopted to decrease the temperature T throughout the iterations. First, we need to set the initial temperature in order to initiate the annealing process. We created a iterative algorithm called *initial temperature* that starts at a low temperature and adjusts itself each iteration until the initial temperature is ideal, described as follows:

1. Generate an initial solution using **generate** algorithm;
2. Starting from any low temperature;
3. Count the proportion of neighbors that are accepted in the maximum number of iterations at this temperature;
4. If that proportion is high, return the current temperature as initial temperature;
5. Otherwise, increase the temperature and repeat the process.

In this way, we will start the annealing process with a high temperature, having a reasonable neighbor acceptance rate. Now we need to define a stopping criterion, the final temperature. To stop the cooling process, we set the final temperature, eps, to 10^{-3} in our experiments.

Another critical parameter to set is the cooling rate. Thereby, with higher temperatures, it is allowed to accept many inadequate solutions. For that reason, we start the procedure with a slower cooling rate, and as the temperature drops, we increase it. In our experiments, we used exponential cooling rate, also known as geometric, proposed by [Kirkpatrick et al. 1983, Wilhelm and Ward 1987]. At the beginning of the process, we set the cooling rate to 0.7. Moreover, as the temperature drops, we will gradually add 0.01.

Finally, we must set the path's size associated with each temperature in the annealing process, i.e., the number of iterations. We set the maximum number of iterations with the difference between edges and the number of nodes. The idea is to have fewer iterations when the graph is sparse.

2.5. Simulated Annealing Algorithm

The procedure starts with an initial solution given by the **generate** algorithm. Then, it chooses a new neighbor from **cycle transition**. Next, we apply the **local search** to this neighbor according to the Boltzmann distribution. Now, suppose that the difference between the current diameter with the candidate diameter is less than zero. In that case, we must accept the transition and move to the new state. Otherwise, we accept the new state according to the Boltzmann distribution. Therefore, we cool down the temperature

through the cooling schedule and repeat the process until we reach the tolerance eps given. Below is our pseudo code of the simulated annealing procedure.

Algorithm 2 Simulated Annealing

procedure SA(annealing parameters)
  Initialize the chain with s_0 given by **generate**
  Update currrent solutions $s = s_0$
  Choose new state s' with **cycle transition**
  Let $\Delta = f(s) - f(s')$
  Do the **local search** in s' with probability $e^{-\frac{\Delta}{T}}$ and update s'
  Recalculate $\Delta = f(s) - f(s')$ if local search was applied
  if $\Delta < 0$ **then**
    Accept transition to new state and update current solution $s = s'$
  else
    Accept transition with probability $e^{-\frac{\Delta}{T}}$
  end if
  Cool the temperature by cooling schedule, repeating this step and step 4
end procedure

3. Experiments and Results

Here are the test execution environment settings. The CPU is an Intel(R) Core(TM) i7-8565U, with 20GB DDR4 RAM. The operating system used was Windows 10 and the programming language used was Python 3.9.5.

To check the efficiency of the algorithm we used four instances, two of them are complete graphs, and the other two are sparse graphs. Most instances have been taken out of a related classic diameter problem, except the states of Brazil, which was our creation. The states of Brazil instance has 26 nodes and 49 edges.

3.1. Annealing Parameters

For these instances, we used the initial temperature algorithm to calculate $T_{initial}$ for each of them. As we saw above, the maximum interation, $iter_{max}$, is the difference between edges and vertices. The column instances show the number of nodes and edges, respectively, for each of them.

Instances	$T_{initial}$	$iter_{max}$
10_45	3194799.99	35
15_105	515978.03	90
20_50	120000	30
states_brazil	1848842.59	23

3.2. Results

In the table below, Z and W, respectively, represents the diameter and weight of the best solution found Moreover, Z^* is the optimal solution to BMDSTP. Then, D_0 and W_0 represent the initial diameter and initial weight, respectively. Finally, $T(s)$ represents the total execution time in seconds from the simulated annealing algorithm.

Instancias	B	Z*	Z	W	D_0	W_0	T(s)
10_45	261	4	4	261	6	437	20.17
15_105	463.2	4	4	440	6	814	158.09
20_50	600	4	4	593	9	517	5.85
states_brazil	13000	9	9	12872	14	14484	91.75

Now, let us look at the performance when compared to other algorithms. As it is a problem that has not been much investigated in the literature, we used the exact algorithm, **EA**. Also, we made a rejection algorithm, **RS**, to compare. We chose the instance of the Brazilian states to show this comparison.

	SA	EA	RS
Z	9	9*	12
T(s)	466.13	>3600	480.71

Table 1. *Best upper bound found: Exact algorithm did not finish process

The optimal solution, when $B = 13000$, is 9, achieved successfully by our SA algorithm in the shortest time. To show the behavior of this solution through the annealing process, we present the graphics below. Figure 3 shows diameter performance during the procedure. At the beginning of the algorithm, we accept many poor states. Consequently, with high temperatures, the graphic shows a high variation of the diameter. Note that we tend to accept fewer bad states as the cooling procedure goes, so the diameter tends to converge to a good solution. Next, in Figure 4, we show the evolution of the weight during the algorithm. We see that it has a more varied behavior when compared to the other one. The reason is that the only modification made on weight depends strictly on the local search. Thus, we perform it mainly in high temperatures, and as the temperature goes down, we can see minor variation.

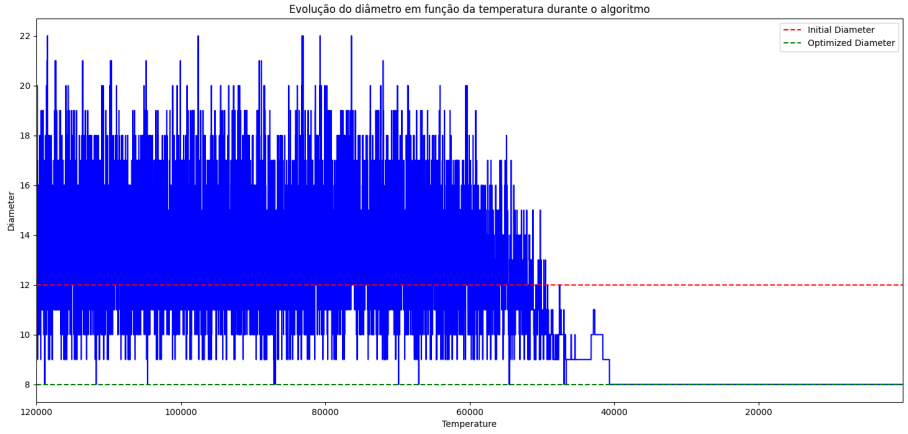

Figure 3. Evolution of diameter

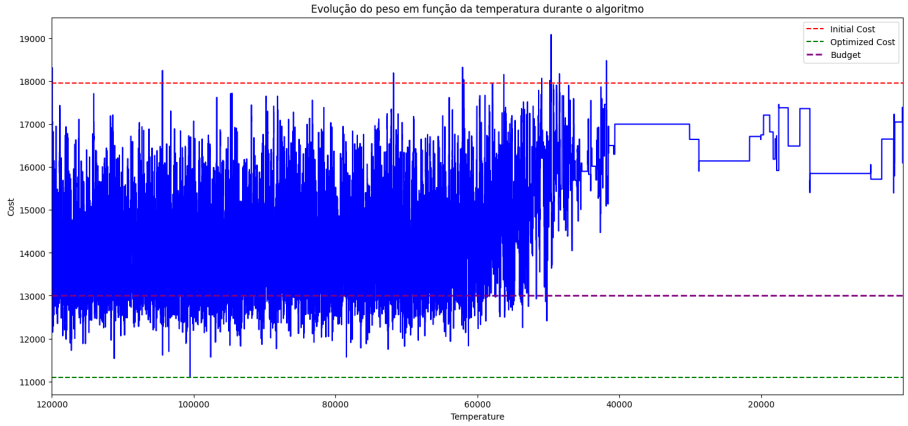

Figure 4. Evolution of weight

4. Discussion and Conclusion

Simulated Annealing is an exciting procedure and has much room for improvement. Thus, there are several ways to combine and create parameters, heuristics, and chains. Our work showed a new and exciting perspective of solving the problem. Although it still needs some adjustments to get better, we believe it was an attractive way to deal with the problem. As a result, we got optimality certificate in all tested instances, even with high times. Moreover, it has unique literary importance.

We noticed that the performance was better for instances that are sparse graphs, as expected. In addition, for graphs with many nodes and edges, the algorithm takes longer to converge. For future work, we would like to research better ways to handle transitions. Also, test other cooling strategies and local searches to try to reduce the execution time.

References

Amanda Azevedo, A. L. (2021). Árvores de diâmetro mínimo sujeitas a restrição de orçamento.

Broder, A. Z. (1989). Generating random spanning trees. In *FOCS*, volume 89, pages 442–447.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

Häggström, O. (2002). *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press.

Hakimi, S. L., Schmeichel, E. F., and Pierce, J. G. (1978). On p-Centers in Networks. *Transportation Science*, 12(1):1–15.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Martin, O. C. and Otto, S. W. (1996). Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63(1):57–75.

Maxwell, J. C. (1860). V. illustrations of the dynamical theory of gases.—part i. on the motions and collisions of perfectly elastic spheres. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 19(124):19–32.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Plesnik, J. (1981). The complexity of designing a network with minimum diameter. *Networks*, 11(1):77–85.

Wilhelm, M. R. and Ward, T. L. (1987). Solving quadratic assignment problems by 'simulated annealing'. *IIE Transactions*, 19(1):107–119.