Connecting the University of Havana:
Design and Analysis of a Degree-Constrained Fiber Optic Network

José Agustín Del Toro González

1 Introduction

The modernization of network infrastructure is a fundamental requirement for the academic and scientific development of modern universities. In this context, the University of Havana, with technical support from ETECSA, aims to interconnect its main buildings through a high-speed fiber optic network.

This work addresses the problem from the perspective of *Design and Analysis of Algorithms*, modeling it as a combinatorial optimization problem on graphs subject to real-world technical constraints. The goal is to design a minimum-cost network that connects all buildings, avoids unnecessary cycles, and respects the limited number of ports available at each network device.

2 Informal Problem Description

A set of university buildings must be interconnected using fiber optic links. Each possible connection has an associated installation cost, and each building is equipped with a network device that supports only a limited number of direct connections.

The objective is to select a subset of connections that:

- Connects all buildings.
- Contains no cycles.
- Respects degree constraints at each building.
- Minimizes the total installation cost.

3 Mathematical Formalization

3.1 Graph Model

The problem is modeled as an undirected weighted graph:

$$G = (V, E, w)$$

where:

- V represents the set of buildings.
- E represents the set of possible fiber connections.
- $w : E \to \mathbb{R}^+$ assigns a cost to each connection.

Each vertex $v \in V$ is associated with a degree bound $b(v)$ representing the maximum number of allowed connections.

3.2 Problem Definition

Find a subset of edges $T \subseteq E$ such that:

1. (V, T) is a spanning tree.
2. $\deg_T(v) \leq b(v)$ for all $v \in V$.
3. The total cost $\sum_{e \in T} w(e)$ is minimized.

This problem is known as the **Degree-Constrained Minimum Spanning Tree (DC-MST)** problem.

4 Computational Complexity Analysis

In this section, the computational complexity of the DC-MST problem is rigorously studied, with the aim of theoretically justifying the algorithmic decisions adopted throughout the remainder of this work.

4.1 Theoretical Framework

This section provides a rigorous analysis of the computational complexity of the *Degree-Constrained Minimum Spanning Tree* (DC-MST) problem. The objective is to justify, from the standpoint of computational complexity theory, why it is not reasonable to expect efficient exact algorithms for its resolution in the general case, and to provide a theoretical foundation for the use of heuristics and approximation methods.

4.2 Membership of DC-MST in NP

A decision problem belongs to the class **P** if there exists a deterministic algorithm that solves it in polynomial time with respect to the input size. A problem belongs to the class **NP** if, given a candidate solution, it can be verified in polynomial time.

A problem is said to be **NP-hard** if every problem in **NP** can be reduced to it through a transformation computable in polynomial time. If, in addition, the problem belongs to **NP**, it is said to be **NP-complete**.

The classical Minimum Spanning Tree (MST) problem belongs to the class **P**. However, as shown below, the introduction of constraints on the maximum degree of vertices radically alters its computational complexity.

4.3 Canonical Problem Chosen: Hamiltonian Path

To prove the NP-hardness of the DC-MST problem, a polynomial-time reduction is employed from the canonical problem **Hamiltonian Path (HP)**.

Definition (Hamiltonian Path). Given an undirected graph $G = (V, E)$, determine whether there exists a simple path that visits all vertices exactly once.

This problem is known to be **NP-complete**.

The choice of Hamiltonian Path is natural for the following structural reasons:

- A path is a connected and acyclic graph, that is, a tree.
- In a path, internal vertices have degree exactly 2.
- The endpoints of the path have degree exactly 1.

These properties directly match the notion of a spanning tree with degree constraints, particularly when the degree bound is equal to 2.

4.4 Construction of the Reduction

Let $G = (V, E)$ be an arbitrary instance of the Hamiltonian Path problem. From this instance, a corresponding instance of the DC-MST problem is constructed as follows:

- Define $V' = V$.
- Define $E' = E$.
- For every edge $e \in E'$, assign a unit weight: $w(e) = 1$.
- Define the degree bound for all vertices as:

$$b(v) = 2 \quad \forall v \in V'.$$

The resulting instance consists of the weighted graph $G' = (V', E', w)$ together with the degree bound function b.

4.5 Correctness of the Reduction

We now prove that the constructed instance of DC-MST has a solution if and only if the original instance of Hamiltonian Path has a solution.

Forward implication ($\Rightarrow$) Assume that a Hamiltonian path exists in the original graph G. By definition, such a path:

- Visits all vertices exactly once.
- Is connected and contains no cycles.
- Contains exactly $|V| - 1$ edges.

Moreover, in a Hamiltonian path each vertex has degree at most 2. Therefore, this path constitutes a valid spanning tree for the DC-MST instance, satisfies all imposed degree constraints, and its total cost is:

$$(|V| - 1) \cdot 1 = |V| - 1.$$

Consequently, a valid solution exists for the constructed DC-MST instance.

Backward implication ($\Leftarrow$) Now assume that a valid solution exists for the constructed DC-MST instance. Such a solution is a spanning tree T that connects all vertices and satisfies:

$$\deg_T(v) \leq 2 \quad \forall v \in V'.$$

A tree in which all vertices have degree at most 2 can only have a path structure. Acyclicity prevents alternative configurations, and the degree constraint excludes any branching. Therefore, the tree T defines a simple path that visits all vertices exactly once.

Consequently, a Hamiltonian path exists in the original graph G.

4.6 Complexity of the Transformation

The described transformation does not add vertices or edges to the original graph. The assignment of weights and degree bounds is performed by traversing the sets V and E, and thus its time complexity is:

$$O(|V| + |E|).$$

Therefore, the reduction is computable in polynomial time.

4.7 Conclusion

Since Hamiltonian Path is an NP-complete problem and there exists a polynomial-time reduction $HP \leq_p DC\text{-}MST$, it follows that the DC-MST problem is **NP-hard**. This result provides the theoretical foundation for the approach adopted in the remainder of this work, which is based on the use of heuristics and approximation methods.

5 Algorithm Design

This section presents the algorithms designed to solve the *Degree-Constrained Minimum Spanning Tree* (DC-MST) problem. Since in Section 4 it was shown that the problem is NP-hard, it is not reasonable to expect the existence of efficient exact algorithms for the general case. Therefore, a layered approach is adopted, combining exact methods, heuristics, and iterative improvement techniques.

This approach is consistent with the principles of Design and Analysis of Algorithms for complex combinatorial optimization problems.

5.1 General design considerations

The DC-MST problem generalizes the classical Minimum Spanning Tree (MST) problem by introducing degree constraints on the vertices. While the MST can be solved in polynomial time, the additional restriction prevents the direct application of classical algorithms such as Kruskal's or Prim's.

Within this context, the algorithmic design follows three fundamental goals:

- Guarantee optimal solutions for small instances (exact algorithm).
- Obtain feasible solutions efficiently (greedy heuristic).
- Improve solution quality without exploring the entire search space (local search).

—

5.2 Exact algorithm by exhaustive enumeration

5.2.1 General idea

The exact algorithm exhaustively explores the space of possible solutions by enumerating all spanning trees of the graph and selecting the one that satisfies the degree constraints

and has the minimum total cost. This approach guarantees global optimality of the obtained solution.

5.2.2 Theoretical foundation

A spanning tree over a set of n vertices contains exactly $n - 1$ edges. The number of possible spanning trees in a graph can grow exponentially with respect to n, which implies that this approach is not viable for large instances. Nevertheless, it is useful as a theoretical reference and for validating heuristic methods.

5.2.3 Pseudocode

Algorithm 1 Exact DC-MST by Brute Force

Require: Graph $G = (V, E, w)$, degree bounds $b(v)$
Ensure: Optimal spanning tree T^*

$T^* \leftarrow \emptyset$
$min\_cost \leftarrow +\infty$
for all spanning trees $T \subseteq E$ **do**
    if $\forall v \in V : \deg_T(v) \leq b(v)$ **then**
        $cost \leftarrow \sum_{e \in T} w(e)$
        if $cost < min\_cost$ **then**
            $min\_cost \leftarrow cost$
            $T^* \leftarrow T$
        end if
    end if
end forreturn T^*

—

5.3 Greedy heuristic with degree constraints

5.3.1 Motivation

Kruskal's algorithm efficiently solves the MST problem by selecting edges of minimum cost while avoiding cycles. However, this algorithm does not consider vertex degree limits. The proposed heuristic adapts this strategy by explicitly checking degree constraints at each step.

5.3.2 Greedy strategy

The heuristic iteratively selects the available edge with minimum weight, provided that its inclusion does not create a cycle and does not violate the degree bounds. Although this strategy does not guarantee optimality, it usually produces good-quality solutions in reasonable time.

5.3.3 Pseudocode

Algorithm 2 Greedy Heuristic for DC-MST

Require: Graph $G = (V, E, w)$, degree bounds $b(v)$
Ensure: Feasible spanning tree T

```
Sort E in non-decreasing order of w
Initialize UNION-FIND structure over V
T ← ∅
deg(v) ← 0  ∀v ∈ V
for all e = (u, v) ∈ E in increasing order do
    if Find(u) ≠ Find(v) then
        if deg(u) < b(u) and deg(v) < b(v) then
            T ← T ∪ {e}
            Union(u, v)
            deg(u) ← deg(u) + 1
            deg(v) ← deg(v) + 1
        end if
    end if
end forreturn T
```

—

5.4 Improvement through local search

5.4.1 Justification

Greedy heuristics may become trapped in suboptimal solutions due to early local decisions. Local search allows exploration of the neighborhood of a feasible solution, seeking incremental improvements without requiring exhaustive search.

5.4.2 Neighborhood definition

The neighborhood of a solution is defined through edge swaps, in which one edge is removed from the current tree and another external edge is added, preserving connectivity and degree constraints.

5.4.3 Pseudocode

—

5.5 Metaheuristics for the DC-MST problem

Since the Degree-Constrained Minimum Spanning Tree problem is NP-hard, even classical heuristics may become trapped in local optima when facing larger instances or unfavorable

Algorithm 3 Local Search for DC-MST

Require: Initial solution T, graph $G = (V, E, w)$, bounds $b(v)$
Ensure: Improved solution T

```
improved ← true
while improved do
    improved ← false
    for all e ∈ T do
        for all e' ∈ E \ T do
            T' ← T − {e} + {e'}
            if T' is a spanning tree then
                if ∀v : deg_{T'}(v) ≤ b(v) then
                    if cost(T') < cost(T) then
                        T ← T'
                        improved ← true
                        break
                    end if
                end if
            end if
        end for
        if improved then
            break
        end if
    end for
end whilereturn T
```

graph structures. For this reason, the incorporation of metaheuristics is considered, as they allow deeper and more flexible exploration of the solution space.

Metaheuristics are characterized by introducing global exploration mechanisms, such as controlled acceptance of worse solutions or the use of memory, while always maintaining feasibility of the generated solutions. In this work, two metaheuristics particularly suitable for combinatorial optimization problems on graphs are implemented: Simulated Annealing and Tabu Search.

5.5.1 Simulated Annealing

Simulated Annealing is a metaheuristic inspired by the physical process of thermal annealing. Its fundamental principle is to allow, with a certain probability, the acceptance of higher-cost solutions in order to escape local optima during the early stages of the search.

The algorithm maintains a current solution and generates neighboring solutions through edge swaps. If the neighboring solution improves the cost, it is accepted directly. Otherwise, it is accepted with a probability that depends on the cost increase and on a parameter called temperature, which gradually decreases throughout the execution.

This strategy enables broad exploration of the solution space at the beginning, favoring local exploitation as the temperature decreases.

Algorithm 4 Simulated Annealing for DC-MST

Require: Initial solution T, graph $G = (V, E, w)$, bounds $b(v)$
Ensure: Best solution found T^*

$T^* \leftarrow T$
Initialize temperature T_{emp}
while $T_{emp} > T_{min}$ **do**
    Generate neighboring solution T'
    if T' is feasible **then**
        if $cost(T') < cost(T)$ **then**
            $T \leftarrow T'$
        else
            Accept T' with probability $e^{-\Delta/T_{emp}}$
        end if
    end if
    if $cost(T) < cost(T^*)$ **then**
        $T^* \leftarrow T$
    end if
    Update temperature
end whilereturn T^*

5.5.2 Tabu Search

Tabu Search is a local-search-based metaheuristic that incorporates explicit memory to avoid cycles and redundant exploration. Unlike classical local search, Tabu Search allows the acceptance of non-improving solutions, provided that the performed move is not forbidden by the tabu list.

In the context of DC-MST, moves consist of edge swaps. Each recent move is temporarily stored in a tabu list, preventing its immediate reversal. This mechanism forces the algorithm to explore new regions of the solution space, enhancing its global exploration capability.

Algorithm 5 Tabu Search for DC-MST

Require: Initial solution T, graph $G = (V, E, w)$, bounds $b(v)$
Ensure: Best solution found T^*

$T^* \leftarrow T$
Initialize tabu list
for maximum iterations **do**
    Generate feasible neighborhood of T
    Select best non-tabu solution T'
    $T \leftarrow T'$
    Update tabu list
    if $cost(T) < cost(T^*)$ **then**
        $T^* \leftarrow T$
    end if
end forreturn T^*

5.6 Synthesis of the algorithmic approach

The proposed algorithmic design combines exactness, efficiency, and solution quality. The exact algorithm provides a theoretical reference, the greedy heuristic ensures scalability, and local search improves solution quality, achieving an appropriate balance between theoretical rigor and practical applicability.

6 Experimental Analysis

This section presents the experimental study carried out on the algorithms designed for the *Degree-Constrained Minimum Spanning Tree* (DC-MST) problem. The main objective is to empirically evaluate the behavior of the algorithms, contrasting the practical results with the theoretical conclusions obtained in previous sections.

Since the problem is NP-hard, experimental analysis is essential to understand the trade-off between optimality, efficiency, and scalability of the proposed solutions.

—

6.1 Experimental Methodology

The experiments were conducted on randomly generated instances using a parameterizable generator. Each instance is defined by the following parameters:

- Number of vertices $|V|$.
- Edge existence probability (Erdős–Rényi model).
- Edge weight range.
- Uniform maximum degree bound $b(v)$.

For each experimental configuration, the following algorithms were executed:

- Exact brute-force algorithm (only for small instances).
- Greedy heuristic with degree constraints.
- Local search applied to the greedy solution.
- Simulated Annealing, using the greedy heuristic solution as the initial solution.
- Tabu Search, applied to the same initial solution, with a fixed-size tabu list.

Each execution measures the computation time and the total cost of the obtained solution. The results were stored in CSV files to facilitate subsequent analysis and reproducibility.

—

6.2 Evaluation Metrics

The metrics used in the experimental analysis are the following:

- **Total cost**: sum of the weights of the edges in the generated tree.
- **Execution time**: computation time measured in seconds.
- **Approximation ratio**: ratio between the cost of a heuristic solution and the optimal cost, when the latter is available.

These metrics allow evaluating both the quality of the solutions and the computational efficiency of each approach.

—

6.3 Experimental Results

The obtained results confirm the theoretical predictions derived from the complexity analysis:

- The exact algorithm exhibits exponential growth in execution time, restricting its use to instances with a small number of vertices.
- The greedy heuristic scales efficiently even for larger instances, producing feasible solutions in very short times.
- Local search systematically improves the cost of the greedy solution, approaching the optimum when it is computable.
- Simulated Annealing successfully escaped local optima generated by classical local search.
- Tabu Search provided a more stable exploration of the solution space thanks to the use of memory.

In instances where it was possible to compute the optimal solution, it was observed that the approximation ratio of the greedy heuristic is acceptable, and that local search significantly reduces this gap. The introduced metaheuristics showed superior performance in terms of solution quality. Both approaches obtained lower-cost solutions than traditional heuristics, at the expense of higher execution times, although this increase remained within reasonable limits for medium-sized instances.

—

6.4 Discussion

The experimental analysis highlights the classic trade-off between optimality and efficiency in NP-hard problems. While the exact algorithm guarantees the best possible solution, its computational cost makes it impractical for real-world scenarios.

On the other hand, heuristics allow obtaining high-quality solutions in reasonable times, making them suitable tools for practical applications. Local search proves to be an effective strategy for refining initial solutions without incurring an exhaustive exploration of the search space.

These results empirically validate the algorithmic strategy adopted in this work.

—

6.5 Threats to Validity

As in any experimental study, certain limitations must be considered:

- The use of random instances may not reflect all possible real-world scenarios.
- Degree constraints were considered uniform, which simplifies some practical cases.
- The analysis is based on a finite number of instances and parameter configurations.

Despite these limitations, the experimental design is sufficient to draw valid conclusions regarding the relative behavior of the evaluated algorithms.

—

6.6 Experimental Conclusion

The conducted experiments confirm that the DC-MST problem exhibits a high level of computational complexity, consistent with its classification as NP-hard. At the same time, they demonstrate that the combination of greedy heuristics and local search allows obtaining efficient and high-quality solutions in practice.

These results support the adopted approach and justify the use of approximate techniques for solving complex network design problems under real-world constraints.

7 Future Work

The work highlights the gap between theoretical optimality and practical efficiency, and demonstrates the usefulness of heuristics for NP-complete problems of real-world relevance.