Discrete Optimization and Integer Programming.

Minimum Spanning Tree

“Nothing takes place in the world whose meaning is not that of some maximum or minimum.” (E. Euler)

HUAWEI

Contents

I. Minimum Spanning Tree

A tree is undirected graph which is connected and has no cycles.

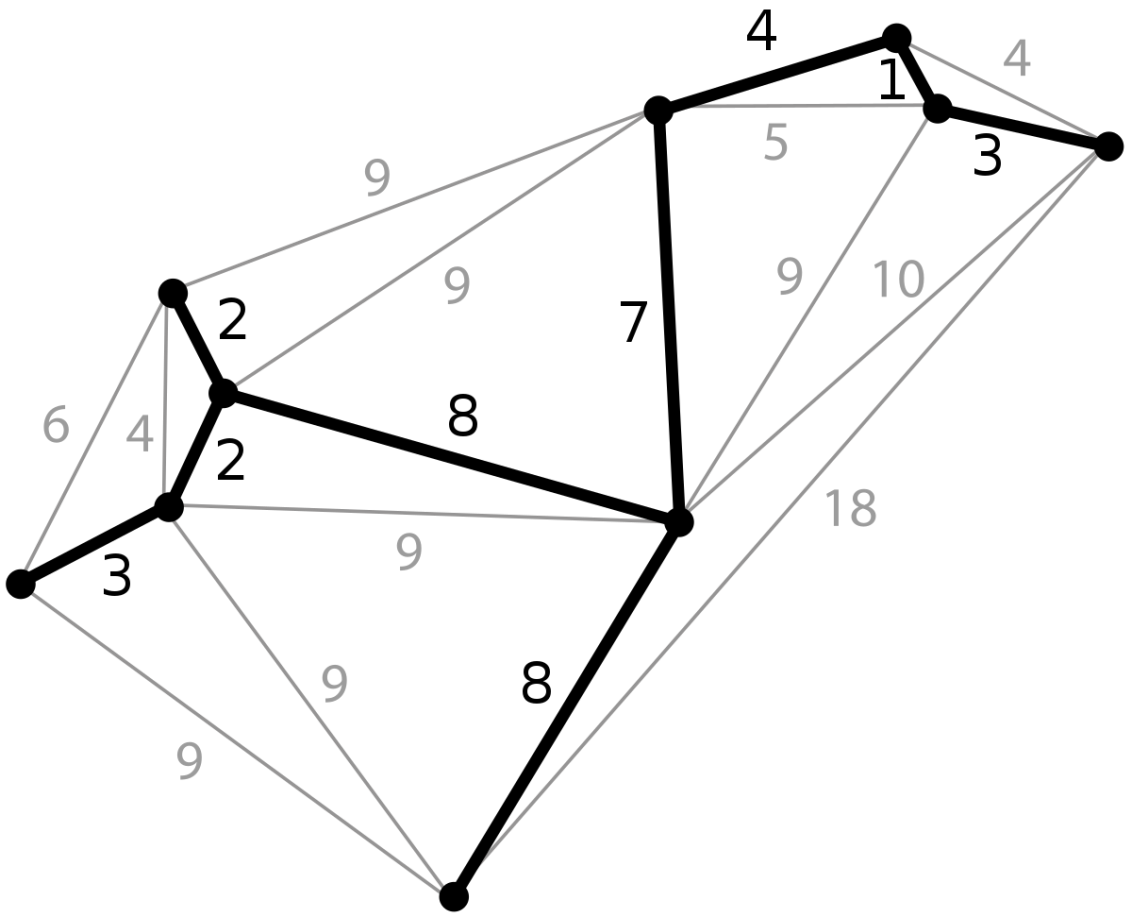
A forest is a disjoint union of trees.

Suppose that we have a connected, edge-weighted undirected graph G.

A *minimum spanning tree (MST)* of the connected, weighted graph G is a subgraph of the graph G which satisfies the properties

- connects all the vertices of G;
- has no cycles;
- has minimum total weight.

If the graph G is not connected then the similar notion is *a minimum spanning forest*.

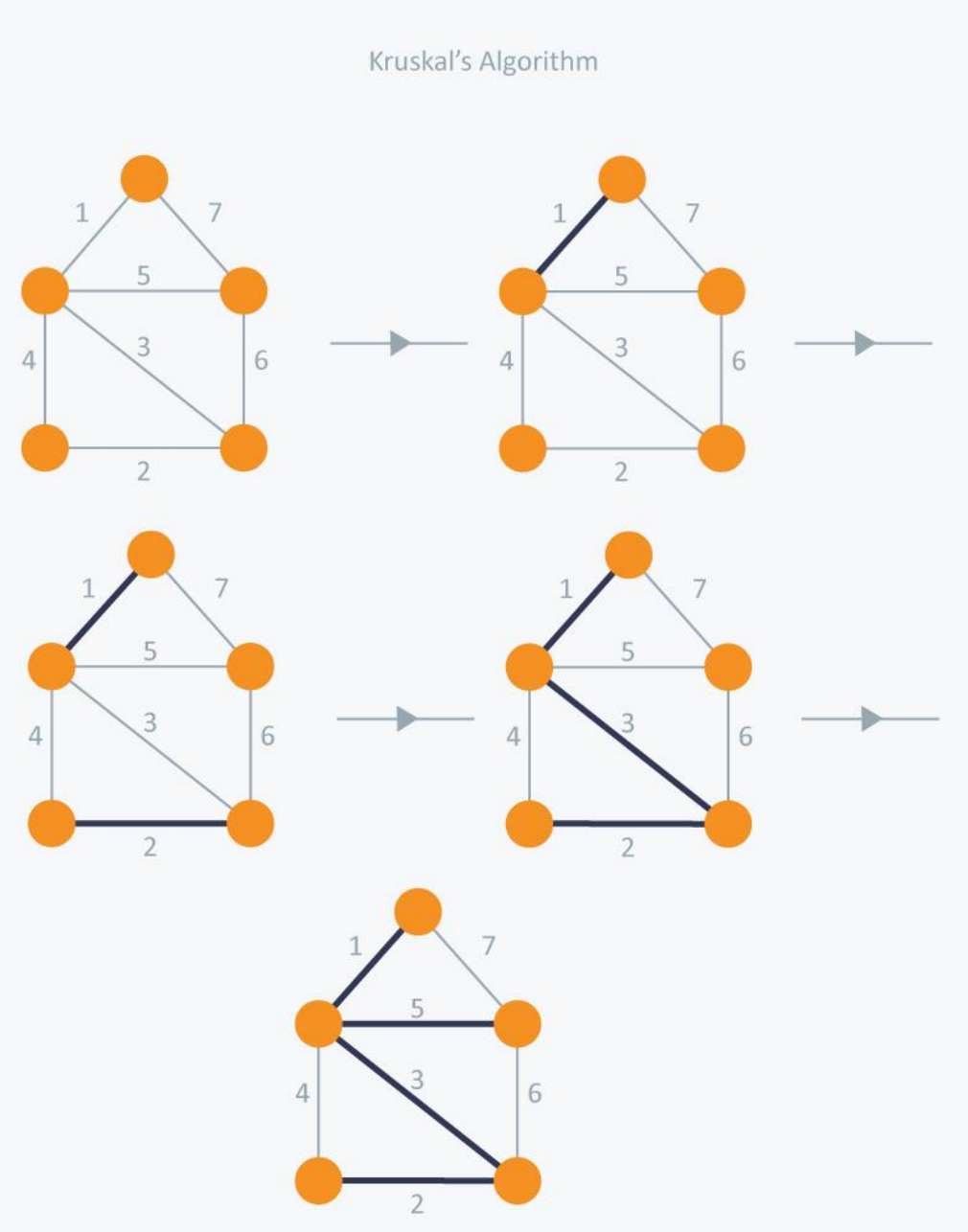


II. Kruskal's Algorithm

The algorithm works as follows

1. Create a forest F (a set of trees), where each vertex in the graph is a separate tree;
2. Create a sorted set S containing all the edges in the graph;
3. If S is empty or F is spanning tree then stop;
4. Remove an edge with minimum weight from S;
5. If the removed edge connects two different trees then add it to the forest F, combining two trees into a single tree;
6. Go to the step 3.

At the termination of the algorithm, the forest F forms a minimum spanning tree for the connected graph.



HUAWEI

II. Kruskal's Algorithm

The proof of correctness consists of two parts. First, it is proved that the algorithm produces a spanning tree. Second, it is proved that the constructed spanning tree is of minimal weight.

Let G be a connected, weighted graph and let T be the subgraph produced by the Kruskal's algorithm. T cannot have a cycle, as by definition an edge is not added if it results in a cycle. T cannot be disconnected, since the first encountered edge that joins two components of T would have been added by the algorithm. Thus, T is a spanning tree of G.

II. Kruskal's Algorithm

Next, we show that the following proposition $\boldsymbol{P}$ is true by induction: If F is the set of edges chosen at any stage of the algorithm, then there is some minimum spanning tree that contains F and none of the edges rejected by the algorithm.

- Clearly $\boldsymbol{P}$ is true at the beginning, when F is empty: any minimum spanning tree will do, and there exists one because a weighted connected graph always has a minimum spanning tree.
- Now assume $\boldsymbol{P}$ is true for some non-final edge set F and let T be a minimum spanning tree that contains F.
  - If the next chosen edge e is also in T, then $\boldsymbol{P}$ is true for $F + e$.
  - Otherwise, if e is not in T then $T + e$ has a cycle C. This cycle contains edges which do not belong to F, since e does not form a cycle when added to F but does in T. Let f be an edge which is in C but not in $F + e$. Note that f also belongs to T, and by $\boldsymbol{P}$, it has not been considered by the algorithm. f must therefore have a weight at least as large as e. Then $T - f + e$ is a tree, and it has the same or less weight as T. So $T - f + e$ is a minimum spanning tree containing $F + e$ and again $\boldsymbol{P}$ holds.
- Therefore, by the principle of induction, $\boldsymbol{P}$ holds when F has become a spanning tree, which is only possible if F is a minimum spanning tree itself.

II. Kruskal's Algorithm

For a graph with m edges and n vertices, Kruskal's algorithm can be shown to run in $O(m \log m)$ time, or equivalently, $O(m \log m)$ time, all with simple data structures. These running times are equivalent because:

- m is at most n^2 and $\log n^2 = 2 \log n \in O(\log n)$.
- if we ignore isolated vertices we obtain $n \leq 2m$, so $\log n$ is $O(\log m)$.

We can achieve this bound as follows:

- First, sort the edges by weight using a comparison sort in $O(m \log m)$ time; this allows the step "remove an edge with minimum weight from S" to operate in constant time.
- Next, we use a disjoint-set data structure to keep track of which vertices are in which components. We place each vertex into its own disjoint set, which takes $O(n)$ operations.
- Finally, in worst case, we need to iterate through all edges, and for each edge we need to do two 'find' operations and possibly one union. Even a simple disjoint-set data structure such as disjoint-set forests with union by rank can perform $O(m)$ operations in $O(m \log n)$ time. Thus the total time is $O(m \log m) = O(m \log n)$.

III. IP/LP Formulation

Lemma. A connected graph T is a tree if and only if $|E(T)| = |V(T)| - 1$.

Using this lemma the minimum spanning tree problem can be modelled by integer programming as follows.

- Variables:
  - Let x_{ij} be 1 if the edge ij in the tree T;
- Constraints:
  - $\sum_{ij \in E(G), i \in S, j \in S} x_{ij} = |S| - 1$ for all $S \subset V(G)$.
  - $x_{ij} \in \{0, 1\}$ for all $ij \in E(G)$,
    or, equivalently, $x_{ij} \geq 0$ for all $ij \in E(G)$.
- Objective function: $\sum_{ij \in E(G)} c_{ij} x_{ij}$

Note that

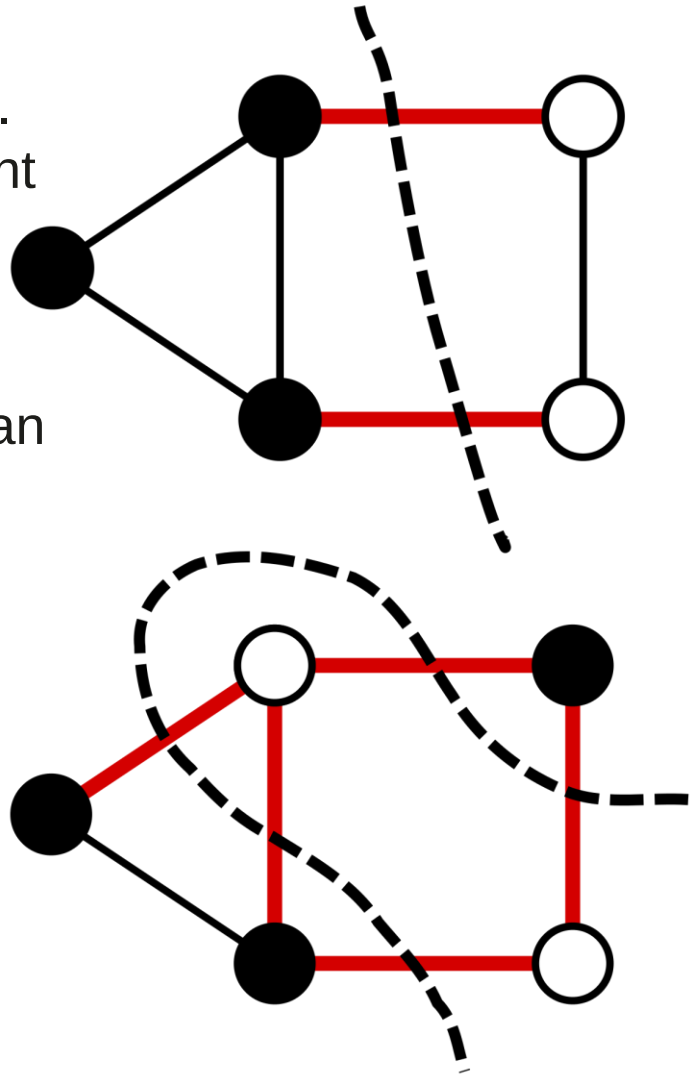
- One can show that the corresponding polyhedron is integral. So the solution of the relaxed LP problem is the solution of the original problem.
- The number of constraints depends exponentially on the size of the graph. So any solver will work too long.

IV. Graph Cut

Some definitions:

- *A cut* is a partition of the vertices of a graph into two disjoint subsets.
- Any cut determines *a cut-set*, the set of edges that have one endpoint in each subset of the partition.
- These edges are said to *cross* the cut.

- A cut is *a minimum cut* if the size or weight of the cut is not larger than the size of any other cut.
- A cut is *a maximum cut* if the size of the cut is not smaller than the size of any other cut.



V. Cut Formulation

Lemma. Assume that we have a spanning tree T in connected graph G. Then any cut-set contains at least one edge of the tree T.

Using this lemma we can model the minimum spanning tree problem in the following way

- Variables:
  - Let x_{ij} be 1 if the edge ij in the tree T;
- Constraints:
  - $\sum_{ij \in E(G)} x_{ij} = |E(G)| - 1$,
  - $\sum_{ij \in E(G): ij \in \delta(S)} x_{ij} \geq 1$ for all $S \subset V(G), \emptyset \neq S \neq V(G)$,
  - $x_{ij} \in \{0, 1\}$ for all $ij \in E(G)$.
- Objective function: $\sum_{ij \in E(G)} c_{ij} x_{ij}$

Note that

- The corresponding polyhedron is not integral in general, it can contain the fractional points. So this model cannot be reduced the LP model.
- The number of constraints also depends exponentially on the size of the graph. So any solver will work too long.

VI. Martin's Formulation

It is very surprising that there exists the LP formulation of the MST problem such that the corresponding polyhedron is integral and the number of constraint is polynomial. This model was proposed in the following paper as the particular case of more general construction:

R. Kipp Martin, Using separation algorithms to generate mixed integer model reformulations, Operations Research Letters, Vol. 10, Issue 3, 1991, pp. 119-128.

In this approach we consider some auxiliary variables z_{kij} where $k, i, j \in V(G)$ along with the usual variables x_{ij}. Constraints are the following:

- $\sum_{ij \in E(G)} x_{ij} = |E(G)| - 1$,
- $z_{kij} + z_{kji} = x_{ij}$ for all $k \in V(G), ij \in E(G)$,
- $\sum_{j \in V(G): j \neq i} z_{kij} \leq 1$ for all $k, i \in V(G), i \neq k$,
- $\sum_{j \in V(G): j \neq k} z_{kkj} \leq 0$ for all $k \in V(G)$.
- $x_{ij} \geq 0$ for all $ij \in E(G)$,
- z_{kij} for all $k, i, j \in V(G)$.

VI. Martin's Formulation

Now let us show that the set of edges in G which correspond to $\overline{x_{ij}} = 1$ in any binary solution $(\bar{x}, \bar{z})$ do not contain a cycle.

Suppose that we have a cycle of undirected edges containing the vertex $k \in V(G)$. Then there exists a corresponding cycle of directed edges defined by $\overline{z_{kij}} = 1$ which also contains vertex k due to the following constraint

- $z_{kij} + z_{kji} = x_{ij}$ for all $k \in V(G), ij \in E(G)$.

We will call the directed edges of this set as k-edges. One can show that the set of k-edges cannot contain a cycle which contains vertex k by the following reasons

- If such cycle exists and it is directed, then this cycle has at least one k-edge directed out of vertex. But it is impossible due to the constraint
  - $\sum_{j \in V(G): j \neq k} z_{kkj} \leq 0$ for all $k \in V(G)$.
- If such cycle exists and it is not directed, then there is at least one vertex i with two k-edges directed out of it. But this is impossible due to the other constraint
  - $\sum_{j \in V(G): j \neq i} z_{kij} \leq 1$ for all $k, i \in V(G), i \neq k$.

Thus, there are no cycles in the solution. Hence, we have a spanning tree.

Thank you.

Bring digital to every person, home, and organization for a fully connected, intelligent world.