View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Erasmus University Digital Repository

The minimum spanning tree and duality in graphs

Wim Pijls
Econometric Institute Report EI 2013-14

April 19, 2013

Abstract

Several algorithms for the minimum spanning tree are known. The Blue-red algorithm is a generic algorithm in this field. A new proof for this algorithm is presented, based upon the duality of circuits and cuts in a graph.
The Blue-red algorithm is genetic, because the other algorithms can be regarded as special instances. This is shown using the same duality.

1 Introduction

The problem of the minimum spanning tree is a classical topic in graph theory. In computer science many textbooks "Algorithms and Data Structures" have been published, which almost always treat the minimal spanning tree. The two best-known algorithms for this problem are those which are named after Kruskal and after Prim. However, yet other algorithms can be mentioned. In this article, we put all algorithms into one framework.
We denote a graph by $G(V, E)$ where V is a set of nodes and E a set of edges. A *tree* is a subgraph without circuits and a *spanning tree* is a tree that is adjacent to all nodes. In a weighted graph, each edge has a weight. The weight of a tree is the sum of the weights in the tree. A minimum spanning tree, MST for short, is a spanning tree with minimal weight. In the eighties several algorithms for the MST were known (cf. section 6); Tarjan launched a new one, the Blue-red algorithm[12].

Blue-red algorithm

> *Apply in random order the blue and red rule until all edges have been colored.*
> *Blue rule: find a cut containing uncolored edges but without blue edges, select among the uncolored edges an edge with minimal weight, color it blue.*
> *Red rule: find a circuit containing uncolored edges but without red edges, select among the uncolored edges an edge with maximal weight, color it red.*

On termination the blue edges make up an MST. In section 6 the classical algorithms are derived as instances of the Blue-red algorithm. The algorithm gives rise to a few questions. First, there is of course the question of correctness. Second, how to explain the symmetric role of circuit and cut? The question of correctness divides into two sub-questions. Is it always possible to select an edge to be colored red or blue? Why does the set of blue edges make up a MST? In this article we answer these questions.

2 Families, anti-families and dual families

In this section we make a step beyond the area of graphs and discuss some properties of sets. The key notion is the notion of a family.

Definition 1 *Given a set Z, a family $\mathcal{F}$ of Z is a collection of subsets $Y \subseteq Z$ with the property: $Y \in \mathcal{F} \wedge X \subset Y \Rightarrow X \in \mathcal{F}$.*

So a family is closed under inclusion. A set $Y \in \mathcal{F}$ such that no superset $X'' \supset Y$, $X'' \in \mathcal{F}$ exists, is called *maximal*. For any family of Z a so-called anti-family can be defined.

Definition 2 *An anti-family $\hat{\mathcal{F}}$ for a family $\mathcal{F}$ of Z comprises the subsets of Z that do not belong to $\mathcal{F}$.*

An anti-family $\hat{\mathcal{F}}$ is closed in the reversed direction: any superset of a set in $\hat{\mathcal{F}}$ also belongs to $\hat{\mathcal{F}}$. Consequently, an anti-family has minimal sets. Next to a given family one has the *dual family*.

Definition 3 *For a given family $\mathcal{F}$ of a set Z the dual family $\mathcal{F}'$ is the collection of subsets $Y' \subseteq Z$ that are disjoint from at least one maximal set Y in $\mathcal{F}$.*

The family $\mathcal{F}'$ is closed as well under inclusion. The maximal sets in $\mathcal{F}'$ are exactly the complements of the maximal sets in $\mathcal{F}$. It is easily shown that $(\mathcal{F}')' = \mathcal{F}$. The dual family $\mathcal{F}'$ in turn has an anti-family, denoted by $\hat{\mathcal{F}}'$. So we have the following schema.

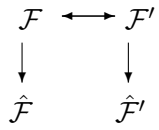

Figure 1: The schema of (anti-)families of a set Z.

A simple relation between $\hat{\mathcal{F}}$ and $\hat{\mathcal{F}}'$ cannot be established. The following theorem establishes a relation between $\hat{\mathcal{F}}$ and $\mathcal{F}'$.

Theorem 1 *The intersection of any set in $\hat{\mathcal{F}}$ and any maximal set in $\mathcal{F}'$ is non-empty.*

Proof If some set $\hat{Y}$ in $\hat{\mathcal{F}}$ has an empty intersection with a maximal set Y' in $\mathcal{F}'$, then $\hat{Y}$ would not belong to $\hat{\mathcal{F}}$ but to $\mathcal{F}$. □

Theorem 1 has a dual counterpart (left to the reader). The above theory is also relevant to the area of hypergraphs. In that area $\hat{Y}'$ in $\hat{\mathcal{F}}'$ is called a transversal for $\mathcal{F}$ [2]. The schema of Figure 1 is the basis for the theory of matroids[9].

3 Graphs and dual graphs

This section discusses some concepts related to graphs. As said before, a graph $G(V, E)$ comprises a set V of nodes and a set E of edges. In a weighted graph, each edge $e \in E$ has a real-valued weight $w(e)$. We assume *connected* graphs. *Connected* means that each pair of nodes can be connected through a path. This condition is not a restriction to our problem, because in a non-connected graph each component has its own spanning tree. In a connected graph a spanning tree has $|V| - 1$ edges. It is clear what is meant by a *circuit* (also known as a cycle). Next to it, we define a *cut*.

Definition 4 *A cut S in a graph $G(V, E)$ is a set of $S \subseteq E$ such that the $G(V, E \backslash S)$ is not connected.*

One can construct a cut as follows. Mark each node in the graph at random with a digit 0 or 1. The edges connecting a 0-node to a 1-node form a cut.
From now, we define a *tree* as a set of edges without any circuit. The collection of trees is the family $\mathcal{F}$ of circuit-free subsets of E. A spanning tree is a maximum set in $\mathcal{F}$. The complement of a maximal spanning tree is a cut-free, because after removal of that complement the graph remains connected. A cut-free set of edges is called a *filling* in this paper. Figure 2 is a specific implementation of Figure 1. Obviously, the minimal sets in the family of circuit containing sets are the

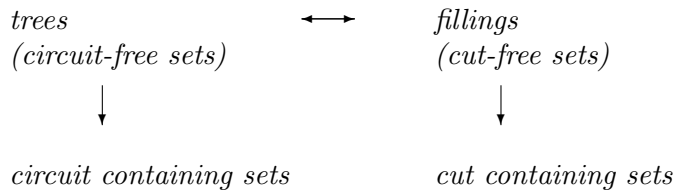

Figure 2: Sets of edges in a graph.

circuits. Likewise the cuts are minimal in the family of cut containing sets. Circuits and cuts are dual counterparts of each other.

For connected planar graphs the duality of circuits and cuts can be illustrated in a visual way. Each planar graph $G(V, E)$ has a dual graph $G'(V', E')$ which is constructed as follows. In each face of G a node v' is placed. Each edge $e \in E$ is crossed by an edge e' that connects the nodes belonging to the adjacent faces. The new nodes and the edges form the sets V' and E' respectively. We say that e and e' are *crossing edges* of each other. For explanation, see Figure 3, taken from [13]. In this figure: $V = \{A, B, C, D, E\}$ and $V' = \{X, Y, Z\}$. The dual of $G'(V', E')$ is $G(V, E)$.
The circuits in G correspond one-to-one with the cuts in G'. Since the dual of G' is G, there is also a one-to-one correspondence between the circuits in G' and the cuts in G. If a subset of E is circuit-free (a tree), then the corresponding subset in E' of crossing borders is cut-free (a filling). Again there is a dual statement.
If a graph $G(V, E)$ itself is a tree, G has no fillings and any set of edges is a cut. In the dual graph G', there is only one node and each edge is a circuit.

4 Proof of the algorithm

In this section, the Blue-red algorithm is proved. We start from a graph $G(V, E)$ where each edge $e \in E$ has a weight $w(e)$. The denotations $A + b$ and $A - b$ stand for respectively $A \cup \{b\}$ and $A \backslash \{b\}$. We need two properties of graphs. For the proof, see the literature on graph theory, for example [2, 3].

Properties of a graph

a) Each pair C and S, where C is a circuit and S a cut, has $|C \cap S| \neq 1$.
b) The set $B + e$, B is a spanning tree and $e \in E \backslash B$, has exactly one circuit C; after deleting randomly one edge from C a spanning tree is recovered.

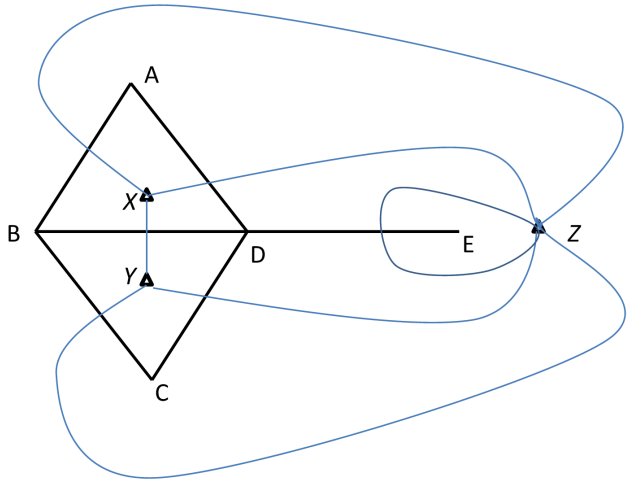

Figure 3: A graph $G(V, E)$ and its dual $G'(V', E')$.

Property b) has a dual counterpart, which we do not need here.

The correctness proof of the Blue-red algorithm comprises the theorems 2 through 5. In these theorems B represents the set of edges that have been colored blue, and R the set of edges that have been colored red. Due to the duality in Figure 2, only half of the proof is given for the theorems 2, 3 and 4. The other half is symmetrical.
The theorems 2 and 4 exhibit invariant properties of the algorithm, i.e. a property that holds after each *step*. By a *step* we mean the execution of the blue or red rule.

Theorem 2 *(invariant)*
a) Set B is a tree.
b) Set R is a filling.

Proof of a) by contradiction (part *b)* is dual).
Assume an edge e in a cut S is colored blue and $B + e$ includes a circuit C with $e \in C$. S has no other blue edges and hence $S \cap C = \{e\}$. This contradicts Property a). □

The next theorem shows that, as long as any edge is uncolored, it is always possible to apply the blue or red rule.

Theorem 3 *The algorithm terminates.*

Proof Suppose $e \in E$ is an uncolored edge and $E \backslash R$ does not contain a circuit, so the red rule cannot be applied. Then $E \backslash R$ is a tree. Since R is a filling and $E = R \cup (E \backslash R)$, R and $E \backslash R$ are maximal sets. Since R is a maximal cut-free set, $R + e$ contains a cut S including e. Moreover $R + e$ and hence S have no blue edges. The blue rule can be applied to S. □

Theorem 4 *(invariant)*
a) Every circuit C includes a non-blue edge e such that $w(e)$ is maximal in C.
b) Every cut S includes a non-red edge e such that $w(e)$ is minimal in S.

Proof Assume that *a)* en *b)* hold after some step; we prove that a) holds after the next step (part *b)* is dual).

Suppose an edge e in a cut S is colored blue according to the blue rule and suppose $w(e)$ is maximal in a circuit C. As a consequence of *b)* $w(e)$ is minimal not only among the weights of the uncolored edges in S, but among all weights in S. (There were no blue edges in S.) Since $|S \cap C| \neq 1$, C and S have another edge e' in common. Since S had no blue edges, e' is not blue. Because $w(e)$ is minimal in S, $w(e') \geq w(e)$. This implies, as $w(e)$ is maximal in C, that $w(e')$ is maximal in C as well. □.

Theorem 5 *On termination of the algorithm:*
a) B is a minimal spanning tree.
b) R is a maximal filling with maximal weight.

Proof of a)
According to Theorem 2, B is a tree and R is a filling. On termination $B \cup R$ is a partition of E. It follows that B and R are maximal sets in the families of respectively trees and fillings.
We prove by contradiction that the weight of B is minimal among the spanning trees. Assume that B' is a spanning tree with $w(B') < w(B)$, where B' is chosen so that $|B' \cap B|$ is as small as possible. (The total weight of B is denoted by $w(B)$.) Add an edge $b \in B \backslash B'$ to B'. According to Property b), $B' + b$ contains a unique circuit C. According to Theorem 4, C has a non-blue edge c such that $w(c)$ is maximal and hence, $w(c) \geq w(b)$. Remove c from $B' + b$ and a new spanning tree B'' is obtained. Because $w(c) \geq w(b)$, we have $w(B'') \leq w(B')$ and hence $w(B'') < w(B)$. It follows that B'' is also an MST. The inequality $|B'' \cap B| < |B' \cap B|$ contradicts the above assumption.
Proof of b). The relation $B \cup R = E$ implies that B and R are maximal sets. Since B has a minimal weight, R has a maximal weight. □

5 The dual graph

When an edge e in a circuit C has been colored red, C is no longer taken into consideration. However, a cut containing e remains relevant for the rest of the execution. Likewise, a colored edge e in cut S, makes this cut useless, but any circuit C including e remains useful. These observations lead us to an alternative formulation of the Blue-red algorithm.
The red rule corresponds to the so-called *deletion* operation. This operation is explained using Figure 4. An edge e is deleted from $G(V, E)$. Then the number of faces decreases. In Figure 4 edge AD is deleted. The faces adjacent to AD are unified. This means that in the dual the nodes X and Z are unified, i.e. contracted. So the crossing edge XZ of AD vanishes in the dual graph. The circuits in $G(V, E)$ including AD do not exist any longer. However, the circuits in $G'(V, E')$ still exist, as are the cuts in $G(V, E)$.
Dually we have *contraction*. This operation is deletion in the dual graph $G'(V', E')$. For instance, in Figure 5 edge YZ has been deleted. The crossing edge BC vanishes and the nodes B and C are contracted. So the faces associated to X and Z are preserved. The circuits in $G(V, E)$ are also preserved.

The new operations enable us to describe the blue and red rule in an alternative way.
Blue rule: select an edge e with minimal weight in a cut; apply contraction and add e to B.
Red rule: select an edge e with maximal weight in a circuit; apply deletion and add e to R.
Notice that the sets V and V' as well as E and E' reduce in each step of the algorithm, due to the deletion and contraction operations, The execution ends with V and V' being single node sets and E and E' being empty.

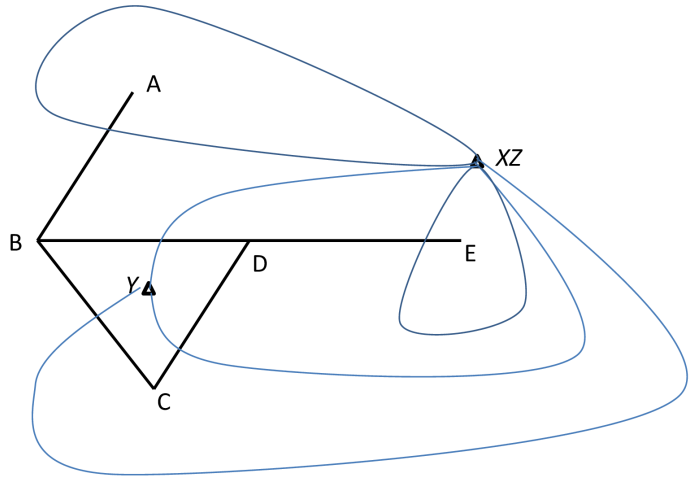

Figure 4: Applying a deletion operation.

6 Algorithms

In the literature on algorithms, mainly aimed at computer scientists, one always finds the algorithms of Kruskal and Prim-Dijkstra-Jarnik, see for example [3, 6]. These are the easiest to implement. In the proofs of those algorithms the Properties a) and b) in section 4 are exploited implicitly. The algorithm of Boruvka-Sollin is purely of historical interest. For a historical overview, see [5].
In the description below *lightest* stands for *having minimal weight.*

Algorithm of Kruskal. Repeat the following steps until $|B| = |V| - 1$. Select the lightest edge e that was not selected previously. If $B + e$ contains a circuit C, C has no red edges and e has a maximal weight in C. If not, then at least one node v is found such that the edges adjacent to v are not blue. This set of edges (including e) makes up a cut. So, add e respectively to R or B.

Algorithm of Jarnik-Prim-Dijkstra. Choose an arbitrary node v and select the lightest edge e adjacent to v. Then $B = \{e\}$. Repeat the following steps until $|B| = |V| - 1$. Select the lightest edge e, adjacent to B and not selected previously. If $B+e$ is not a circuit, add e to B, otherwise to R.

Algoritme of Boruvka-Sollin. Given a graph $G(V, E)$, we construct a tree graph $G(V, B)$. Initially we have $B = \emptyset$ and $|V|$ components each consisting of exactly one node. Select at each component C_i the lightest adjacent edge e_i, where e_i is not selected previously. Add the edges e_i to B. (A side e_i may be selected from two components.) The graph $G(V, B)$ with the new B has a smaller number of components. Repeat the action until $|B| = |V - 1|$.
Note that the algorithm does not work properly if the graph includes equal weights. It is possible that three edges e_i form a triangle. The algorithm is commonly named after Boruvka. In [1] it is explained why the name Sollin is associated to it.

We might add the *Dual Kruskal algorithm* to the above overview. Delete the heaviest edge e from the graph, such that the graph stays connected. Add e to R. Repeat this action until $|V - 1|$ edges remain. These make up the set B.
The simplest derivative of the Blue-red algorithm can be formulated as follows: open each circuit

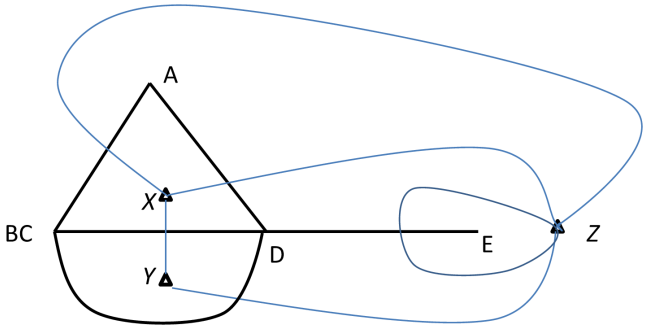

Figure 5: Applying a contraction operation.

by deleting its heaviest edge.

7 Concluding remarks

The theory in this paper is actually the theory of matroids applied to graphs. That theory is described in [11] and comprehensively in [9]. The schema in Figure 1 also applies to matroids. Kruskal's algorithm is a special case of the Greedy algorithm. By imposing additional requirements to a family $\mathcal{F}$ one obtains a matroid. For each matroid the Properties a) and b) in section 4 are valid. Property a) gives a relationship between the families $\hat{\mathcal{F}}$ and $\hat{\mathcal{F}}$ in Figure 1.
Most of the literature ignores the duality of circuits and cuts. In the original publication[12] of the Blue-red algorithm the blue rule and the red rule are proved separately. The papers of Turing award winner Dijkstra[4] contain a discussion of Kruskal's algorithm, where duality is recognized implicitly. Well-known textbooks on algorithms such as [3, 6] do not mention matroids and do not exploit the duality. In [7] the Blue-red algorithm is discussed on the basis of matroids. Before the Blue-red algorithm was published[12], a similar algorithm dealing with matroids was mentioned in [8] citing [10].

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network flows: Theory, Algorithms, and Applications, Prentice Hall 1993.

[2] C. Berge, Hypergraphs, Combinatorics of finite Sets, North-Holland Publishing Company, 1973.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, Third Edition. MIT Press, 2009.

[4] E.W.Dijkstra Archive, nr. 1273, available at:
`http://www.cs.utexas.edu/~EWD/ewd12xx/EWD1273.PDF`

[5] R.L.Graham and Pavol Hell, On the History of the Minimum Spanning Tree Problem, Annals of the History of Computing, Volume 7, Nr. 1, January 1985.

[6] M.T. Goodrich and R. Tamassia, Algorithm Design, Wiley, 2002.

[7] Dexter Kozen, The Design and Analysis of algorithms, Springer, 1991.

[8] E. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rhinehart and Winston, New York, 1976.

[9] J.G Oxley, Matroid theory, Oxford University Press, 1992.

[10] P. Rosenstiehl, L'Arbre Minimum d'un Graphe, in: Theory of Graphs, Rosenstiehl editor, Gordon and Breach, New York 1967.

[11] A. Schrijver, Combinatorial Optimization, volume 2, Springer, 2003.

[12] R.E Tarjan, Data Structures and Network Algorihms, CBMS-NSF Regional Conference Series in Applied Mathematics, 1983.

[13] R.J. Wilson, Introduction to Graph Theory, Longman, 1972.