Dilworth's theorem revisited, an algorithmic proof

Wim Pijls Rob Potharst*

Econometric Institute Report EI 2011-13

Abstract

Dilworth's theorem establishes a link between a minimal path cover and a maximal antichain in a digraph. A new proof for Dilworth's theorem is given. Moreover an algorithm to find both the path cover and the antichain, as considered in the theorem, is presented.

1 Introduction

Dilworth's theorem establishes a link between a minimal path cover and a maximal antichain in a digraph. There are multiple proofs of Dilworth's theorem[3, 4, 8, 11]. Those proofs do not show how an optimal path cover and optimal antichain are obtained for a given graph.
In the current paper we give a new proof, based upon an algorithm that constructs a minimal path cover along with a maximal antichain. The key of the proof is the following observation. Any path cover is a flow in an associated network and vice versa, so finding a minimal path cover can be reduced to finding a minimal flow. The algorithm for a minimal flow generates a cut (analogously to the famous maxflow/mincut connection), which is closely related to the maximal antichain. So an algorithm is presented generating a path cover and an antichain which are certificates of the correctness of Dilworth's theorem.
For a variant of Dilworth's theorem a similar proof was given in [5, section 4.9]. In [6] a direct proof was presented for the proposition that any maximal cut corresponds to a maximal antichain. The minimal flow construction was not involved in that proof.
One application of the maximal antichain problem is found in the field of data mining, see [9].

2 Path covers and flow networks

Path cover and antichain. A *path cover* in a digraph $D(V, A)$ is a collection of paths such that every node of V is included in at least one path. The number of paths is called the *size* of the path cover.
In a digraph a starting node is a node without incoming arcs. Likewise, an end node is a node without outgoing arcs. We assume that all paths of a path cover run from a starting node to an end node. If this property is not satisfied in a given path cover, it can be introduced without changing the size of the path cover. An *antichain* in $D(V, A)$ is a set of nodes, no two of which are included in any path of $D(V, A)$. The core of this paper is the proof of Dilworth's theorem:

*Econometric Institute, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands, e-mail: {pijls,potharst}ese.eur.nl

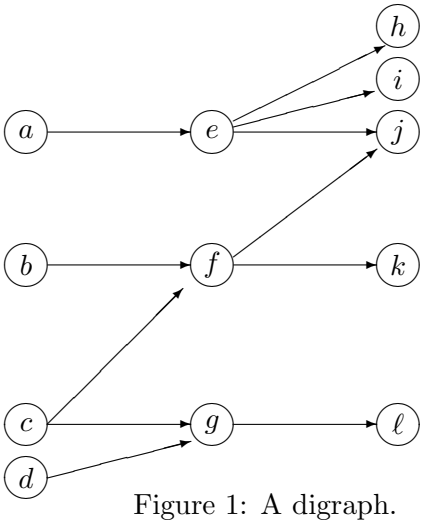

Figure 1: A digraph.

Dilworth's theorem: The minimal size of a path cover in a digraph D equals the maximal size of an antichain in D.

Flow network. A *flow network* is a digraph $N(V, A)$ containing two special nodes s and t being the single starting node and the single end node, and two functions ℓ and u, the lower and the upper capacity respectively, mapping the arc set A into $\mathbb{R}$. We call s and t the source and the sink respectively. A *flow* f in a given network is a function f from A into $\mathbb{R}$ which obeys the capacity constraint given by: $\ell(a) \leq f(a) \leq u(a)$ for all $a \in A$. Moreover, a flow has to meet the balance constraint defined by the equality $\sum_{(j,i)\in A} f(j,i) = \sum_{(i,k)\in A} f(i,k)$ for any node i, $i \neq s, t$. The balance constraint actually says that the incoming flow (inflow) must be equal to the outgoing flow (outflow) in each node, except in the source and the sink. If the balance constraint is met in every node, the outflow of s equals the inflow of t. This value is called the *value* of the flow.
Next to a flow, the *preflow* is a known notion. For a preflow, the balance constraint is relaxed: the inflow is larger than or equal to the outflow in each node (except the source).

Given a path cover C in a network, we define the function $f(i, j)$ for an arc (i, j) as the number of paths in C that go through arc (i, j). If the capacity constraint holds, f is a flow and its value equals the size of C. Conversely, an integer-valued flow f in an acyclic network is easily decomposed into a collection of paths (not necessarily constituting a path cover).

An associated flow network. From the theory of network flows it is known that any algorithm generating a maxflow ends with a cut, inducing a set of arcs. Since Dilworth's theorem deals with nodes, we make an extra step. Each node $v \in V$ is split up into two nodes v_1 and v_2 with an arc in between. To be more specific, we give the following definition.

Definition 1 *Given a digraph $D(V, A)$, an associated flow network $N(V', A')$ is constructed as follows:*

a) for each node $v \in V$, there are two nodes v_1 and v_2 in V' and an arc (v_1, v_2) in A';

b) for each arc $(v, w) \in A$, there is an arc (v_2, w_1) in A';

c) in addition to the node pairs (v_1, v_2) defined in a) the set V' contains a source node s and a sink node t; there is an arc (s, v_1) in V' for each starting node $v \in V$ and an arc (v_2, t)

for each end node $v \in V$;

d) the lower capacity $\ell(a')$, $a' \in A'$ equals 1 for the arcs a' defined in a) and equals 0 for the arcs a' defined in b) or c); the upper capacity of each arc a' equals $+\infty$.

See Figures 1 and 2 for an example of a digraph and its associated flow network (ignore the numbers of Figure 2 as yet). Since a (minimal) path cover corresponds to a (minimal) flow, the minimal flow problem is studied in section 3.

3 Finding a minimal network flow

Minflow reduced to maxflow. The problem of finding a flow of minimal value (*minflow* for short) does not receive much attention in the mathematical literature. Only a few references can be given, e.g. [2] or [9, section 2.2]. On the other hand there is a vast literature on maxflow algorithms. Almost any textbook on graph theory or combinatorial optimization treats this problem, e.g. [1, Chapter 6-7], [7, Chapter 6] and [10, Chapter 10]. For finding a minflow, we apply a method different from the above references. A given flow network $N(V, A)$ can be transformed into a network $\tilde{N}(V, A)$ by establishing new capacities $\tilde{u}$ and $\tilde{\ell}$ as follows: $\tilde{u}(i, j) = -\ell(i, j)$ and $\tilde{\ell}(i, j) = -u(i, j)$ for every arc $(i, j) \in A$. At any time the flow $\tilde{f}$ in the transformed network $\tilde{N}(V, A)$ corresponds to a flow f in the original network $N(V, A)$ according to the equation $\tilde{f} = -f$. Similarly, a maxflow in $\tilde{N}(V, A)$ corresponds to a minflow in $N(V, A)$. It is also possible to design a minflow algorithm in its own right. The operations of a maxflow algorithm can easily be translated into operations working on the original minflow instance. The augmenting paths of a maxflow algorithm can be converted into decreasing paths.

An initial flow. Two types of maxflow algorithms are distinguished: path augmenting algorithms and preflow-push algorithms. Either type requires an initial flow or an initial preflow respectively. Most of the literature assumes that all lower capacities are equal to 0 and in that case establishing an initial (pre)flow is trivial.
However, the above transformation converts the capacities of the associated flow graph $N(V', A')$ as defined in section 2 into negative upper and lower bounds. In that case, a different approach for finding an initial flow is necessary. Constructing a path cover in an acyclic graph is straightforward. This path cover generates a flow f and the related flow $\tilde{f} = -f$ can be used as the initial flow for the maxflow algorithm.

Note. In the context of Dilworth's theorem we only consider networks that do not include cycles. Suppose we have a network which does include cycles and some arcs $(i, j$ have $\ell(i, j) > 0$ or $u(i, j) < 0$. In that case, the construction of an initial flow is not straightforward. See [1, section 6.7] for this problem.

A cut in a network. Starting from the initial flow, the flow is repeatedly changed during the running time of any maxflow algorithm. All algorithms work with an auxiliary graph, the so-called residual graph. For a flow network $N(V, A)$ with flow function f, the *residual graph* $R_f(V, \bar{A})$ is defined as follows. The node set remains the same. The arc set $\bar{A}$ consists of arcs (i, j) (forward arcs) such that $(i, j) \in A$ and $f(i, j) < u(i, j)$, and arcs (j, i) (backwards arcs) such that $(i, j) \in A$ and $f(i, j) > \ell(i, j)$. Any maxflow algorithm ends when the residual graph R_f contains no path from s to t. As long as the residual graph contains such a path, the value of the flow can be increased.
A *cut* (S, T) in a flow network is a pair of sets $S \subset V$ and $T = V \backslash S$, such that $s \in S$ and

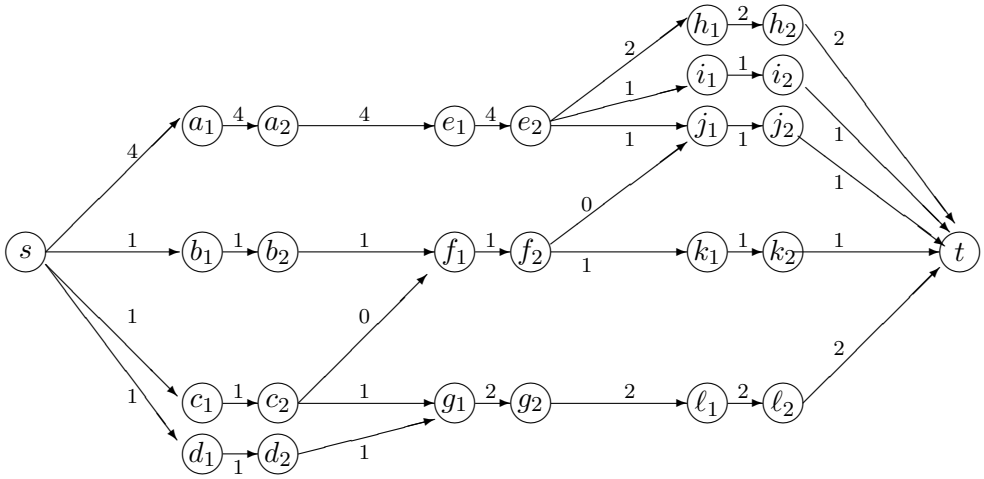

Figure 2: The flow network associated to the graph of Figure 1. The numbers denote an arbitrary flow function f.

$t \in T$. When the algorithm stops, one constructs a cut (S, T) in the flow network using the final residual graph. S is the set of nodes $v \in V$ such that the final residual graph has a path from s to v. Since there is no path from s to t, S does not include t and hence, the collection (S, T) with $T = V \backslash S$ is a cut.
An arc (i, j) is called a *crossing arc* if $i \in S$ and $j \in T$; it is called an anti-crossing arc if $i \in T$ and $j \in S$. Given the definition of the residual graph and the construction of the cut, it is easily derived that a maxflow f has the property:

$$f(i, j) = u(i, j) \tag{1}$$

for each crossing arc (i, j) and $f(i, j) = \ell(i, j)$ for each anti-crossing arc (i, j).

No anti-crossing arcs. The network $N(V', A')$ defined in section 2 has $u(i, j) = +\infty$ for every arc (i, j). So the corresponding maxflow instance has $\ell(i, j) = -\infty$. Since each maxflow algorithm terminates after a finite number of iterations (apart from some pathological exceptions), $f(i, j) = \ell(i, j) = -\infty$ cannot happen. Consequently there are no anti-crossings arcs in the final flow.
This property implies an important result for the final path cover: *every path starting in S and ending in T has exactly one crossing arc.* Indeed, if a path contained two crossing arcs, an anti-crossing arc would be somewhere in between.

Example. The flow in the network of Figure 2 has value 7, or, equivalently, the corresponding path cover comprises seven paths from s to t. Notice that the path $s, a_1, a_2, e_1, e_2, h_1, h_2, t$ is counted twice. The final minflow with value 5 is shown in Figure 3. The above mentioned path is counted once in this minflow. Furthermore, there is a new path in the induced path cover: $s, c_1, c_2, f_1, f_2, j_1, j_2, t$. Due to this new path the number of paths covering the pieces $s, a_1, a_2, e_1, e_2, j_1$ and $c_2, g_1, g_2, \ell_1, \ell_2$ has been reduced. The piece c_2, f_1, f_2, j_1 is contained in an extra path. The reduction of the flow value from 7 to 5 is achieved by two steps in the related max-flow algorithm using augmenting paths.
The resulting cut is given by $S = \{s, a_1, a_2, e_1, e_2, h_1, i_1, b_1, c_1, d_1\}$ and $T = V \backslash S$.

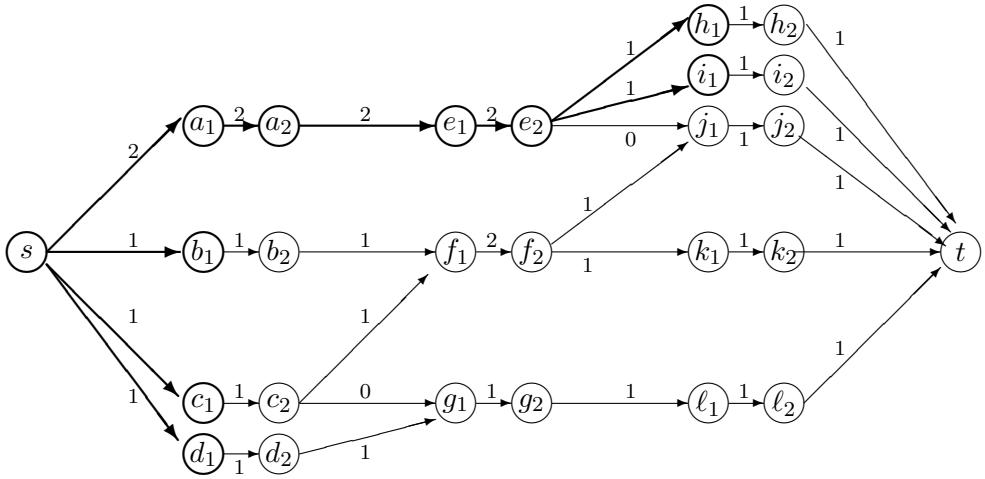

Figure 3: The minimal flow or minimal path cover in the flow network of Figure 2. (The nodes and arcs inside the S set are drawn in bold.)

4 Proof of Dilworth's theorem

In this section Dilworth's theorem is proved. The essential part of this proof is covered by Lemma 1.

Lemma 1 *The minimal flow in $N(V', A')$ induces a path cover and an antichain of equal size.*

Proof The minimal flow f in $N(V', A')$ is integer-valued, since all capacities are integers. This flow can be decomposed into a path cover C and the number of paths through arc (i, j) equals $f(i, j)$.
We mentioned in section 3 that any maxflow of minflow algorithm ends with a cut (S, T). As argued in section 3, every path starting in S and ending in T has exactly one crossing arc. So the size of C, say n, equals the number of paths in the crossing arcs, or in formula:

$$n = \sum_{i \in S, j \in T} f(i, j) \tag{2}$$

The counterpart for minflow in $N(V', A')$ of (1) says that $f(i, j) = \ell(i, j)$. Define Q' as the set of crossing arcs of the form (v_1, v_2). These arcs have $\ell(v_1, v_2) = 1$ and they are the only crossing arcs with ℓ-value $\neq 0$. Thus, (2) may be extended to

$$n = \sum_{i \in S, j \in T} f(i, j) = \sum_{i \in S, j \in T} \ell(i, j) = \sum_{(v_1, v_2) \in Q'} 1. \tag{3}$$

Define $Q = \{v_1 \mid (v_1, v_2) \in Q'\}$. The number of elements in Q' and hence also in Q is equal to n, due to (3). So the size of Q equals the size of a path cover.
We show by contradiction that Q is an antichain. Any path through a node $v_1 \in Q$ also contains the crossing arc (v_1, v_2). If a path contained two nodes from Q, it would also contain two crossing arcs, which is impossible as shown in section 3. □

Proof of Dilworth's theorem. According to the definition of an antichain, a path in a path cover cannot include two nodes of an antichain. So the size of any path cover is larger than or equal to the size of any antichain and hence:

$$\textit{minimum size of a path cover} \geq \textit{maximum size of a antichain.} \tag{4}$$

Equality in this relation is obtained thanks to Lemma 1. Now we have proved Dilworth's theorem in $N(V', A')$. Due to the lower capacity $\ell(v_1, v_2) = 1$ in $N(V', A')$ for each arc associated to a node $v \in V$, any path cover in $N(V', A')$ induces a path cover $D(V, A)$. It is clear that an antichain in $N(V', A')$ corresponds to an antichain in $D(V, A)$. Hence, Dilworth's theorem also holds in $D(V, A)$. □

Example *(continued).* There are six crossing arcs: (b_1, b_2), (c_1, c_2), (d_1, d_2), (h_1, h_2), (i_1, i_2) and (e_2, j_1), five of which make up the set Q'. So $Q = \{b_1, c_1, d_1, h_1, i_1\}$ and $\{b, c, d, h, i\}$ is the obtained maximal antichain in Figure 1.

5 Summary

Constructing a path cover and an antichain of equal size is the crux in the proof of Dilworth's theorem. The construction of such a path cover and antichain in a digraph $D(V, A)$ is summarized in the following steps.

1. Construct the associated network $N(V', A')$.
2. Construct in $N(V', A')$ an arbitrary path cover, which induces an initial flow f by superposition.
3. Transform $N(V', A')$ into network $\tilde{N}(V', A')$ having new upper and the lower capacities (see section 3). The initial flow in $\tilde{N}(V', A')$ is $\tilde{f} = -f$.
4. Execute any maxflow algorithm in $\tilde{N}(V', A')$; a derived product is a cut along with a set of crossing arcs. Convert the maxflow into a minflow.
5. Decompose the minflow into a path cover for $N(V', A')$ and next for $D(V, A)$.
6. Select the arcs of the type (v_1, v_2) from the crossing arcs obtained in step 4. The set of corresponding nodes v is an antichain in $D(V, A)$.

As mentioned in section 3, we can design a minflow algorithm in its own right. Hence, the above steps 3 and 4 can be replaced with the statement: execute a minflow algorithm in $N(V', A')$. Like a maxflow algorithm, a minflow algorithm starts with an initial flow. Notice that the initial flow f of step 2 obeys the capacity constraints of $N(V', A')$.

References

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

[2] E. Ciurea and L. Ciupala, Sequential and parallel algorithms for minimum flows, *J. Appl. Math. and Computing*, Vol. 15(2004), No. 1 - 2, pp. 53 - 75.

[3] R.P. Dilworth, A decomposition theorem for partially ordered sets, *Annals of Mathematics*, Vol. 51 (1951), pp. 161-166.

[4] F. Galvin, A Proof of Dilworth's Chain Decomposition Theorem, *American Mathematical Monthly*, Vol. 101, No. 4 (1994), pp. 352-353.

[5] E. Lawler, *Combinatorial Optimization*, 1976, reprint from Dover Publications, 2001.

[6] R.H. Moehring, Algorithmic aspects of comparability graphs and interval graphs. In: *Graphs and Order*, 1985, Reidel, Dordrecht, pp. 41-101.

[7] Chr. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, 1982, reprint from Dover Publications, 1998.

[8] M. Perles, A proof of Dilworth's decomposition theorem for partially ordered sets, *Israel J. Math*, 1963, pp. 105-107.

[9] M. Rademaker, Optimal resolution of reversed preference in multi-criteria data-sets, PhD thesis Ghent University Belgium, 2009.

[10] A. Schrijver, *Combinatorial Optimization, Polyhedra and Efficiency*, Springer-Verlag, 2003.

[11] H. Tverberg, On Dilworth's decomposition theorem for partially ordered sets, *J. Combin. theory*, 1967, pp. 305-306.