Research Article

Another Note on Dilworth's Decomposition Theorem

Wim Pijls and Rob Potharst

Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

Correspondence should be addressed to Wim Pijls; pijls@ese.eur.nl

Received 8 June 2012; Accepted 9 November 2012

Academic Editor: Stefan Richter

Copyright © 2013 W. Pijls and R. Potharst. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a new proof of Dilworth's theorem. The proof is based upon the minflow/maxcut property in flow networks. In relation to this proof, a new method to find both a Dilworth decomposition and a maximal antichain is presented.

1. Introduction

Several proofs are known for Dilworth's theorem. This theorem says that, in a poset P , a maximal antichain and a minimal path cover have equal size. Shortly after Dilworth's seminal paper [1] a "Note" [2] was published containing an algorithmic proof, that is, a proof which also gives a method to find a combination of a maximal antichain and a minimal path cover. The other proofs [1, 3–5] are nonalgorithmic. The key issue in [2] is the relation between a minimal path cover and a maximal antichain in P on the one hand and a maximal matching and a minimal vertex cover (in this order) in an associated bipartite graph B on the other hand. Dilworth's theorem is proved in [2] using König's theorem stating that, in a bipartite graph, a maximal matching and a minimal vertex cover have equal size.

The combination of a maximal matching and a minimal vertex cover in B corresponds to a maxflow/mincut combination in a flow network B' akin to B . So the obvious algorithm for a minimal path cover along with a maximal antichain is executing a maxflow/mincut algorithm in B' associated indirectly to P .

In the current paper a shortcut is proposed between maxflow/mincut and an optimal path cover jointly with an antichain. To a given poset P we associate a flow network N which is much simpler than graph B' constructed via a matching/vertex cover instance. A similar idea for finding a maximal antichain is found in [6]. However, the discussion in that paper was not connected with Dilworth's theorem.

Other more complex algorithms in this domain can be found in [7–10]. For an application of the maximal antichain we refer to [11, 12].

2. Some Preliminaries

A poset $P(V, <)$ is a partially ordered set V , where $<$ denotes a transitive order relation. An ordered pair $a, b \in V$ with $a < b$ is called *transitive* if there exists an element $c \in V$ such that $a < c < b$. A poset $P(V, <)$ can be transformed into a directed acyclic graph $D(V, A)$ in a straightforward manner. Each nontransitive pair a, b with $a < b$ generates an arc $(a, b) \in A$. An example of a graph derived from a poset is given in Figure 1.

An *antichain* in a directed acyclic graph $D(V, A)$ is a set of nodes, no two of which are included in any path of $D(V, A)$. A *path cover* in $D(V, A)$ is a collection of paths, not necessarily disjoint, such that every node is included in at least one path. The number of paths is called the *size* of the path cover. Now we can formulate Dilworth's decomposition theorem.

Dilworth's Theorem. In a poset the size of a maximal antichain equals the size of a minimal path cover.

This theorem can also be formulated for a directed acyclic graph (DAG for short). In Section 3, the essential part of this paper, we deal with DAG's.

A *flow network* is a directed acyclic graph $N(V, A)$ containing two special nodes s and t being the source and the sink, and two constraint functions ℓ and u mapping the

FIGURE 1: A directed acyclic graph (DAG).

FIGURE 2: The flow network associated to the graph of Figure 1; the numbers at the arcs represent the minimal flow. (Nodes and arcs inside S are in bold).

arc set A into $\mathbb{R}$. A *flow* f in a network is a function f from A into $\mathbb{R}$ which obeys the flow 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 $i \neq s, t$. Due to this balance constraint the outflow of s equals the inflow of t . This value is called the *value* of the flow.

An integer-valued flow f in an acyclic network can be regarded as a path collection and vice versa.

3. A Maxcut/Minflow in a Network

An Associated Flow Network. To find a combination of a maximal antichain and a minimal path cover in a DAG we construct an acyclic flow network associated to the DAG under consideration.

Definition. Given a directed acyclic graph $D(V, A)$ the associated flow network $N(V', A')$ is constructed as follows:

- 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' ;
- for each arc $(v, w) \in A$, there is an arc (v_2, w_1) in A' ;
- 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) for each node $v \in V$ without incoming arcs in $D(V, A)$ and an arc (v_2, t) for each node $v \in V$ without outgoing arcs in $D(V, A)$;
- the constraint function $\ell(a), a \in A'$ equals 1 for the arcs a defined in (a) and equals 0 for the arcs a defined in (b) or (c); $u \equiv +\infty$ holds for the other constraint.

Figure 2 shows the flow network associated to the graph of Figure 1 (ignore the numbers for the moment). A feasible flow in $N(V', A')$ has $f(v_1, v_2) \geq \ell(v_1, v_2) = 1$ for every $v \in V$ and can be decomposed into a path cover for N . Dilworth's theorem will be proved using the minflow/maxcut equality in network N .

Minflow/Maxcut Reduces to Maxflow/Mincut. The problem of finding a flow of minimal value (*minflow* for short) is not given much attention in the literature. Only a few references can be found, for example, [13] and [12, Section

3.3]. For achieving a minflow in $N(V', A')$ we apply a method different from those references. Maxflow is a classical problem discussed in many textbooks, for example, [14, 15]. We can reduce minflow to maxflow by establishing new capacities $\tilde{u}(i, j) = -\ell(i, j)$ and $\tilde{\ell}(i, j) = -u(i, j)$ for every arc $(i, j) \in A$. A maxflow $\tilde{f}$ related to the new capacities is converted to a minflow $f = -\tilde{f}$.

In any maxflow algorithm an initial feasible flow is required. Most of the literature assumes $\ell \equiv 0$ and in that special case an initial flow is trivial. However, in the generalized case a different approach is necessary. Construct an arbitrary path cover in $N(V', A')$. This path cover may be regarded as a flow f obeying the constraints ℓ and u in the above definition. The related flow $\tilde{f} = -f$ is an initial feasible flow for the maxflow algorithm.

The well-known maxflow/mincut relationship has a counterpart in the minflow theory. A *cut* (S, T) in a flow network $N(V', A')$ is a pair of sets $S \subset V'$ and $T = V' \setminus S$, such that $s \in S$ and $t \in T$. In the generalized maxflow theory, the *capacity* of a cut is defined as $\sum_{i \in S, j \in T} u(i, j) - \sum_{i \in T, j \in S} \ell(i, j)$. (The generalized case does not assume $\ell \equiv 0$). We have reduced the minflow problem to maxflow by inverting the constraint functions. Analogously, we invert the constraints in the definition of the capacity m of a cut (S, T) :

$$m = \sum_{i \in S, j \in T} \ell(i, j) - \sum_{i \in T, j \in S} u(i, j). \quad (1)$$

Now the following property applies: the minimal value of a flow equals the maximal capacity of a cut. So using a modified capacity definition we have obtained the counterpart of the generalized maxflow/mincut theorem in networks with arbitrary constraints, see [14, Section 6.7].

An arc (i, j) is called a *forward arc* if $i \in S$ and $j \in T$; a *backward arc* has $i \in T$ and $j \in S$. Given a minflow f and a cut (S, T) of maximal capacity (a maxcut), the forward arcs (i, j) satisfy $f(i, j) = \ell(i, j)$ and the backward arcs (i, j) satisfy $f(i, j) = u(i, j)$, again analogously to the maxflow/mincut theory. To derive Dilworth's theorem, we first prove the following crucial lemma.

Lemma. In network $N(V', A')$, a cut of maximal capacity m generates an antichain of size m .

Proof. Since each flow algorithm terminates after a finite number of iterations (apart from some pathological exceptions), $f(i, j) = u(i, j) = +\infty$ cannot happen. Hence, there are no backward arcs in $\text{maxcut}(S, T)$. The only forward arcs with ℓ -value $\neq 0$ have the form (v_1, v_2) . Define Q as the set of nodes $v_1 \in V'$ such that (v_1, v_2) is a forward arc. Relation (1) transforms to

$$m = \sum_{i \in S, j \in T} \ell(i, j) = \sum_{v_1 \in S, v_2 \in T} 1 = \sum_{v_1 \in Q} 1 \quad (2)$$

and thus $|Q| = m$. We show by contradiction that Q is an antichain. Suppose a path includes two nodes v_1 and w_1 from Q . This path must also contain arc (v_1, v_2) . Since $v_1, w_1 \in S$ and $v_2 \in T$, a backward arc is contained in the path. Hence, it is a contradiction.

The minflow in $N(V', A')$ has also value m and this flow can be decomposed into a path cover of size m . An antichain and a path cover in $N(V', A')$ clearly correspond to an antichain and path cover in the original DAG.

According to the definition of an antichain, a path cannot include two nodes of an antichain. This implies that the size of any path cover is larger than or equal to the size of any antichain. Since we have found a path cover and an antichain of the same size m , we conclude that the minimal size of a path cover and the maximal size of an antichain are equal.

Now, an algorithmic proof of Dilworth's theorem has been obtained. The result in this section is twofold: a new proof for Dilworth's theorem and a construction method for a maximal antichain along with a minimal path cover.

Example. The minflow in the network of Figure 2 has value 5. The maxcut is given by $S = \{s, a_1, a_2, e_1, e_2, h_1, i_1, b_1, c_1, d_1\}$ and $T = V \setminus S$. There are six forward 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) . The set Q in the Lemma consists of five nodes and is equal to $\{b_1, c_1, d_1, h_1, i_1\}$. Hence, the maximal antichain of the DAG in Figure 1 is $\{b, c, d, h, i\}$.

4. Discussion and Comparison

All antichain constructing methods proposed so far are based upon the relationship between the theorems of Dilworth and König, as proposed in [2]. From a poset P a bipartite graph $B(V', A'')$ is derived, where V' is the same set as in $N(V', A')$ defined in Section 3. So B has two nodes for every element in the poset. By contrast, the set of arcs A'' strongly differs from our set A' . Every ordered pair in P , regardless whether it is a transitive or a nontransitive one, becomes an arc in A'' , the mapping being one-to-one. Hence, A'' is a huge set in general. To the huge bipartite graph $B(V', A'')$ a maximum matching algorithm is applied, which is essentially a maxflow algorithm. In each iteration one looks for an augmenting path. The number of iterations equals the size of the maximum matching. At the end a maximum matching and a minimum vertex cover of size m' are obtained, corresponding to, respectively, a minimal path cover and a maximal antichain in P of size $m = |V| - m'$.

Because of the transitivity of the arcs, a cover of *disjoint* paths is generated, but this property does not matter when only a maximal antichain is asked for.

Using the method proposed in this paper, a maxflow algorithm is applied to $N(V', A')$ with inverted constraints. The difference between A' and the above set A'' is significant. The number of iterations of the maxflow algorithm running on $N(V', A')$ depends on the shape of the initial flow. Due to the small size of A' each iteration deals with a small number of arcs.

Suppose that the size m for the maximal antichain and the minimal path cover is relatively small compared to $|V|$. Then the number of arcs in $B(V', A'')$ is large and the number of iterations in the maxflow algorithm running on this graph B equals $m' = |V| - m$, a relatively large number. In that case the new method is superior to the old one.

References

- [1] R. P. Dilworth, "A decomposition theorem for partially ordered sets," *Annals of Mathematics*, vol. 51, pp. 161–166, 1951.
- [2] D. R. Fulkerson, "Note on Dilworth's decomposition theorem for partially ordered sets," *Proceedings of the American Mathematical Society*, vol. 7, no. 4, pp. 701–702, 1956.
- [3] F. Galvin, "A proof of Dilworth's chain decomposition theorem," *American Mathematical Monthly*, vol. 101, no. 4, pp. 352–353, 1994.
- [4] M. A. Perles, "A proof of Dilworth's decomposition theorem for partially ordered sets," *Israel Journal of Mathematics*, vol. 1, no. 2, pp. 105–107, 1963.
- [5] H. Tverberg, "On Dilworth's decomposition theorem for partially ordered sets," *Journal of Combinatorial Theory*, vol. 3, no. 3, pp. 305–306, 1967.
- [6] R. H. Möhring, "Algorithmic aspects of comparability graphs and interval graphs," in *Graphs and Order*, pp. 41–101, Reidel, Dordrecht, The Netherlands, 1985.
- [7] A. A. Benczur, "Pushdown-reduce: an algorithm for connectivity augmentation and poset covering problems," *Discrete Applied Mathematics*, vol. 129, no. 2–3, pp. 233–262, 2003.
- [8] S. Felsner, V. Raghavan, and J. Spinrad, "Recognition algorithms for orders of small width and graphs of small Dilworth number," *Order*, vol. 20, no. 4, pp. 351–364, 2003.
- [9] S. Ikiz and K. Vijay Garg, "Online algorithms for Dilworth's chain partition," Tech. Rep., Parallel and Distributed Systems Laboratory, Department of Electrical and Computer Engineering, University of Texas at Austin.
- [10] G. V. Jourdan, J. X. Rampon, and C. Jard, "Computing online the lattice of maximal antichains of posets," *Rapport de Recherche 2271*, Inria, Paris, France, 1994.
- [11] H. A. M. Daniels and M. Velikova, "Derivation of monotone decision models from noisy data," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 36, no. 5, pp. 705–710, 2006.
- [12] M. Rademaker, B. de Baets, and H. de Meyer, "Optimal monotone relabelling of partially non-monotone ordinal data," *Optimization Methods and Software*, vol. 27, no. 1, pp. 17–31, 2012.
- [13] E. Ciurea and L. Ciupală, "Sequential and parallel algorithms for minimum flows," *Journal of Applied Mathematics and Computing*, vol. 15, no. 1–2, pp. 53–75, 2004.

- [14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, New York, NY, USA, 1993.
- [15] A. Schrijver, *Combinatorial Optimization, Polyhedra and Efficiency*, Springer, New York, NY, USA, 2003.