JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematical Sciences
Vol. 80B, No. 3, July-September 1976

Finding the Circuits of a Matroid*

Edward Minieka**

(June 14, 1976)

Given the bases of a matroid, this paper presents a primal algorithm and a dual algorithm for finding the circuits of the matroid.

Key words: Algorithm; graph; matroid; network; operations research.

1. Introduction

A matroid is a combinatorial structure that possesses important combinatorial properties of a wide variety of mathematical structures. Such varied structures as vector spaces, transversals, certain ployhedral corner points, cycles in a graph, spanning trees, and the source arcs used by a network flow are all special cases of matroids. Matroid theory provides a convenient way to summarize these scattered results and to extend them simultaneously [1, Ch. 21],[1] [2–5], [8], [11].

Let E be any finite set of elements. There are several equivalent ways to define a matroid on E [1, p. 476], [2], [9], [10]. The following definition is convenient for the purposes of this paper.
Independence Definition of a Matroid Let $\mathcal{I}$ be any set of subsets of E with the following properties:

(1) If $A' \subseteq A$ and $A \epsilon \mathcal{I}$, then $A' \epsilon \mathcal{I}$ and

(2) for any subset E' of E, all maximal [2] members of $\mathcal{I}$ that are contained in E' have the same cardinality.

The members of family $\mathcal{I}$ are called the *independent sets* of the matroid. The maximal members of $\mathcal{I}$ are called the *bases* of the matroid. It follows from (2) that all bases have the same cardinality. Let $\mathcal{B}$ denote the set of all bases. Set $\mathcal{B}$ equivalently defines a matroid on E.

For example, let E be the set of edges in a connected, undirected graph. The set of all forests of this graph form the independent sets of a matroid on E. The bases of this matroid are the spanning trees of the graph. Call this matroid a *tree matroid.*

A subset of E that is not independent is called *dependent.* A *circuit* is any minimal dependent set. The circuits of the tree matroid constitute the set of all simple cycles in the graph.

Let M be a matroid on set E that is specified by its base set $\mathcal{B}$. Form set $\mathcal{B}^*$ by taking the complements in E of each member of $\mathcal{B}$. The independent sets corresponding to $\mathcal{B}^*$ also satisfy properties (1) and (2) and thus $\mathcal{B}^*$ defines a matroid M^* on set E. Matroid M^* is called the *dual* of matroid M.

For example, the dual of a tree matroid is the matroid whose bases are the complements of spanning trees. Call this matroid the *co-tree matroid.* The set of all simple cuts forms the circuit set of the co-tree matroid.

What further relationships exist between the bases and circuits of a matroid?

AMS Subject Classification: 05B35, 05C25.
* An invited paper.
** Present address: Department of Quantative Methods, University of Illinois, Chicago, Illinois 60680.
[1] Figures in brackets indicate the literature references at the end of this paper.
[2] Set S is a *maximal* member of $\mathcal{S}$ if S is not contained in any other member T. of $\mathcal{S}$, $S \neq T$. Set S is a *minimal* member of $\mathcal{S}$ if S contains no other member T of $\mathcal{S}$, $S \neq T$.

LEMMA 1: *Let* I *be any independent set of matroid* M.

Let $x \in E$, $x \notin I$ *and* $I \cup \{x\}$ *be dependent.*
Then, $I \cup \{x\}$ contains exactly one circuit. For the proof, see [1, p. 479].

Let D be any subset of E. For matroid M, the *rank* of D, denoted by $r(D)$, is defined as the cardinality of the largest independent set of M that is contained in set D. Thus, $0 \leq r(D) \leq |D|$.

The *closure* of set D, denoted by $cl(D)$, is defined as the largest set D' such that $D \subseteq D'$ and $r(D) = r(D')$. It can be shown [2, p. 70] that $cl(D)$ is unique.

Set D is called *closed* if $D = cl(D)$. For example in a tree matroid, the rank of a subset D of edges is the cardinality of the largest forest that can be grown inside set D. The closure of set D is set D together with all other edges that have both endpoints in the same component of D. Consider the tree matroid of the graph in figure 1. Let $D = \{a, b, f\}$, then, $r(D) = 3$, $cl(D) = \{a, b, e, f, g\}$. Since $D \neq cl(D)$, set D is not closed. However, set $\{a, b, e, f, g\}$ is closed.

A *hyperplane* of matroid M is any closed set whose rank equals $r(E) - 1$. Let $\mathcal{H}$ denote the set of all hyperplanes of matroid M. Consider the tree matroid for the graph shown in figure 1. For this matroid, $r(E)$ equals 4 which is the cardinality of each spanning tree (base). The set $\{a, b, f, e, g\}$ is a hyperplane of this matroid since a forest of at most 3 edges can be formed from this set.

Let $\mathcal{C}^*$, $\mathcal{B}^*$, $\mathcal{I}^*$, $\mathcal{H}^*$, r^*, respectively denote the circuit set, base set, independent sets, hyperplane set and rank function of matroid M^*, the dual of matroid M.

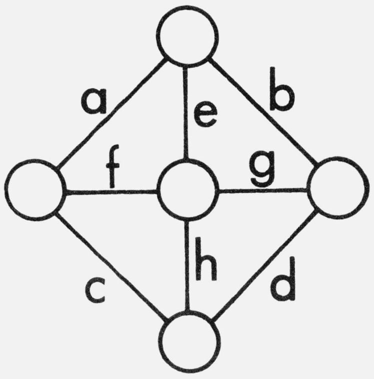

FIGURE 1

LEMMA 2: $\mathcal{C} = \{E - H^* : H \in \mathcal{H}^*\}$

PROOF. Let $C \in \mathcal{C}$. Since C is a minimal dependent set of M, it follows that $C - \{x\}$ is independent for any $x \in C$. Thus, $E - C \cup \{x\}$ contains a base of M^*, but $E - C$ does not contain a base of M^*.

Thus, $E - C$ is a hyperplane of M^*.
Consequently $\mathcal{C} \subseteq \{E - H^* : H^* \in \mathcal{H}^*\}$.

Let $H^* \in \mathcal{H}^*$. Then $E - H^*$ contains a base of M. Suppose $x \notin H^*$. Then $H^* \cup \{x\}$ is a base of M^*, and $E - H^* - \{x\}$ is a base of M. By Lemma 1, $E - H^*$ contains a unique circuit of M. Since $E - H^* - \{x\}$ is independent in M, $E - H^*$ is a circuit. Thus $\mathcal{C} \supseteq \{E - H^* : H^* \in \mathcal{H}^*\}$.

Q.E.D.

Hence to determine the circuits of M we need only determine the hyperplanes of M^*.

2. Algorithms to Construct the Circuits from the Bases

An algorithm to construct the base set using only knowledge of the circuit set was developed by Hull [6]. This section presents two complementary algorithms that construct the circuit set using knowledge of only the base set. These algorithms exploit the properties of circuits given in Lemmas 1 and 2.

Primal Algorithm

For each $B \in \mathcal{B}$ and for each $x \in E$, $x \in B$, form the set $B \cup \{x\}$. Let $B=\{x_1,x_2, \ldots, x_{r(E)}\}$. For $i=1, 2, \ldots, r(E)$, check if $B \cup \{x\}-\{x_i\}$ is a base. If so, color x_i orange; otherwise, color x_i blue. Element x together with all orange elements form a circuit.

Proof: By Lemma 1, $B \cup \{x\}$, $x \notin B$, contains exactly one circuit C. If deleting an element x_i, $i=1, 2, \ldots, r(E)$, creates an independent set $B \cup \{x\}-\{x_i\}$, then x_i must be present in circuit C. Otherwise, x_i is not present in circuit C. Consequently, circuit C must consist of x and all orange elements.

Since a cirucit is a minimal dependent set, the deletion of any member x of a circuit C results in an independent set I. Thus, there is some base B, $I \subseteq B$, such that $C \subseteq B \cup \{x\}$. Hence, all circuits are generated by the algorithm.

Since E is a finite set, the algorithm must terminate after a finite number of steps.

Q.E.D.

Improving the Algorithm

The essential operation of the algorithm is coloring. Coloring involves checking if the set $B \cup \{x\}-\{x_i\}$ is a base. To generate one circuit requires $r(E)$ such base list checks, i.e., for $x_1, x_2, \ldots, x_{r(E)}$. If the elements of E are treated as letters and the bases written in alphabetic order, each base list check is equivalent to checking if a word of $r(E)$ letters appears or does not appear in a dictionary consisting only of words of $r(E)$ letters. Of course, computational short cuts can be developed for bases with special structures, such as trees.

Needless to say, it is probable that some circuit C will be generated more than once. Repeated generation of the same circuit can be avoided by checking to see if each set $B \cup \{x\}$ under consideration contains a circuit previously generated. If so, there is no need to examine $B \cup \{x\}$ further since it will yield a previously discovered circuit.

However, as the number of discovered circuits increases, this becomes more involved. Also, this operation is not an identity check as above but a proper subset check which involves more computations. Consequently, it might be computationally advantageous to generate a circuit repeatedly rather than to perform many proper subset operations. Ultimately, this depends on the magnitude of $r(E)$ and the computational efficiency of the proper subset operation. See [7, p. 391].

Example: Let's generate the circuits of the tree matroid of the graph in figure 2. The bases are listed below:

Given Bases (Spanning Trees)	
1.	*abde*
2.	*abdf*
3.	*abef*
4.	*acde*
5.	*acdf*
6.	*acef*
7.	*bcde*
8.	*bcdf*
9.	*bcef*

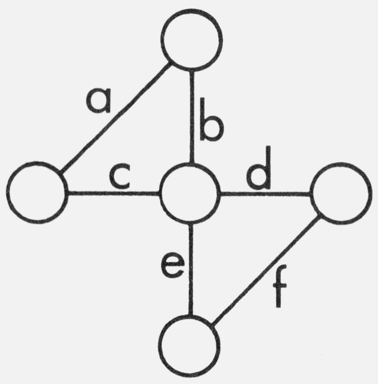

Figure 2

1. Examination of *abde*
   Consider *abde*+*c*

   a—color a orange since *bcde* is a base
   b—color b orange since *acde* is a base
   d—color d blue since *abce* is not a base
   e—color e blue since *abcd* is not a base

   Resulting circuit *abc*
   Consider *abde*+*f*

   a—color a blue since *bdef* is not a base
   b—color b blue since *adef* is not a base
   d—color d orange since *abef* is a base
   e—color e orange since *abdf* is a base

   Resulting circuit *def*.

All further sets contain either circuit *abc* or *def* and will yield no new circuits.

From section 1, we know that the circuits of the tree matroid are the simple cycles of the graph. Obviously, the circuits *abc* and *def* produced by the algorithm are the only simple cycles of this graph.

Dual Algorithm

Step 1. Determine $\mathcal{B}^*$ from $\mathcal{B}$ by taking complements.

Step 2. For each $B^* \in \mathcal{B}^*$ and for each $x \in B^*$, form the set $B^*-\{x\}$. For each $y \in E$, $y \notin B^*$, determine if $(B^*-\{x\}) \cup \{y\}$ has rank $r^*(E)$ by consulting $\mathcal{B}^*$. If so, ignore y. Otherwise, color y red. After examining all $y \notin B^*$, let $H_{B^*,x}$ be the set $B^*-\{x\}$ along with all red y. For all B^* and all $x \in B^*$, generate $H_{B^*,x}$.

Step 3. Find the complement of each $H_{B^*,x}$ generated in Step 2. These sets are the circuits of M.

PROOF: By Lemma 2, we need determine only the hyperplanes of M^*. From Step 1, we can determine the bases of M^*. Clearly, each base B^* of M^* less one of its members x has dual rank $r^*(E)-1$ and is contained in exactly one hyperplane of M^*. This hyperplane $H_{B^*,x}$ is constructed in Step 2.

Since E is assumed to be finite, the algorithm terminates in a finite number of steps.

Q.E.D.

Improving the Algorithm

Step 1 and Step 3 require only the operation of taking complements.

Step 2 requires that we test if a set $(B^*-\{x\}) \cup \{y\}$ is a base of M^*. As before, this requires checking to verify if this set is on the list of all bases.

Clearly, it is possible that there exist bases B^*_1 and B^*_2 with $x_1 \in B^*_1$ and $x_2 \in B^*_2$ such that

$$cl(B^*_1-\{x_1\})=cl(B^*_2-\{x_2\}).$$

In this case, the algorithm would generate this hyperplane twice, which is computationally inefficient. To avoid generating this hyperplane more than once, the algorithm could maintain a list $\mathcal{L}$ of all the hyperplanes already generated. Before the algorithm starts to generate the closure of a set $B^*-\{x\}$ in Step 2, the algorithm could check if set $B^*-\{x\}$ is a subset of any hyperplane H* in list $\mathcal{L}$. If so, then $B^*-\{x\} \subseteq H^*$, and

$$r^*(B^*-\{x\})=r^*(E)-1=r^*(H^*)$$

Since closures are unique, it follows that

$$H^*=H_{B^*,x}.$$

If $B^*-\{x\}$ is not contained in any member of list $\mathcal{L}$, then the algorithm should proceed to generate $H_{B^*,x}$ as outlined in Step 2.

Checking list $\mathcal{L}$ involves verifying if the set under consideration is a proper subset of any member of this list. If the list is long, this can involve a large number of operations. Moreover, the proper subset operation is computationally more difficult than an identity check. Consequently, it might be better to generate a hyperplane repeatedly. Ultimately, this depends on the magnitude of $r^*(E)$ and the computational efficiency of the proper subset operation.

EXAMPLE: Let's generate the circuits of the tree matroid of the graph in figure 2 using the dual algorithm. The bases and their complements are listed below:

Step 1.

	Given Bases (Spanning Trees)	Step 1 Complements
1.	*abde*	*cf*
2.	*abdf*	*ce*
3.	*abef*	*cd*
4.	*acde*	*bf*
5.	*acdf*	*be*
6.	*acef*	*bd*
7.	*bcde*	*af*
8.	*bcdf*	*ae*
9.	*bcef*	*ad*

Step 2. Examination of cf

1. Consider $cf-f=c$

   a—color red since $r(ac)=1$
   b—color red since $r(bc)=1$
   d—ignore d since $r(cd)=2$
   e—ignore e since $r(ce)=2$

   Resulting hyperplane *abc*

2. Consider $cf-c=f$

   a—ignore a since $r(af)=2$
   b—ignore b since $r(bf)=2$
   d—color red since $r(df)=1$
   e—color red since $r(ef)=1$

   Resulting hyperplane *def*

Examination of *ce*

1. Consider $ce-e=c$	c is already contained in *abc* generated above.
2. Consider $ce-c=e$	e is already contained in *def* generated above.

As seen, the examination of each remaining member of $\mathcal{B}^*$ will yield no new hyperplanes.

Step 3. Take the complements of the sets generated in Step 2.

Set	Circuit
abc	*def*
def	*abc*

3. Primal Versus Dual Algorithm

The primal algorithm and the dual algorithm are complementary algorithms.

To generate a circuit, the primal algorithm adds one element to a primal base and then deletes elements from the resulting set of $r(E)+1$ members. Each deletion decision requires checking the list of all primal bases. For each primal base, $r(E)(|E|-r(E))$ such list checks are required. Thus, at worst, $|\mathcal{B}|r(E)(|E|-r(E))$ list checks are required in total.

To generate a dual hyperplane, the dual algorithm deletes one element from a dual base and then adds elements to the resulting set of $r^*(E)-1$ elements. Each addition decision requires checking the list of all dual bases. For each dual base, $r^*(E)(|E|-r^*(E))$ list checks are required. Thus, at worst, $|\mathcal{B}^*|r^*(E)\ (|E|-r^*(E))$ list checks are required in total.

Since

$$|\mathcal{B}|=|\mathcal{B}^*|$$

and

$$r^*(E)=|E|-r(E),$$

both algorithms require the same number of list checks. Moreover, these lists have the same number of members.

Which algorithm should be used? Barring special computational procedures that exploit the special nature of the matroid, it is probably best to use the primal algorithm if $r(E)<r^*(E)$ and the dual algorithm if $r^*(E)<r(E)$. In this way, the length of the words on the base list will be minimized.

What if we elect not to generate repeatedly the same circuit or the same hyperplane? Which algorithm is best? If the primal algorithm is selected, then we must check if sets of $r(E)+1$ elements contain any circuit already generated. If the dual algorithm is selected, we must check if sets of $r^*(E)-1$ elements are contained in any hyperplane already generated.

If $r(E)<r^*(E)$, the former is preferable computationally. If $r^*(E)<r(E)$, the latter is preferable computationally. This is consistent with the above preferences.

Hull [6] gives an algorithm for constructing the circuits of M from the circuits of M^*. If $r^*(E)<r(E)$, the primal algorithm applied to M^* could generate the circuits of M^*, and Hull's algorithm could be used to generate the circuits of M from the circuits of M^*. If $r(E)<r^*(E)$, the dual algorithm applied to M^* could generate the circuits of M^*, and Hull's algorithm could be used to generate the circuits of M from the circuits of M^*. Unfortunately, Hull's algorithm is rather involved, and there seems to be no computational advantage in this circuitous approach.

4. References

[1] Berge, C., Graphs and Hypergraphs, translated by E. Minieka, (North-Holland, Amsterdam, 1973).

[2] Edmonds, J., Minimum partition of a matroid into independent subsets, J. Res. Nat. Bur. Stand. (U.S.), **69**B (Math. and Math. Phys.), Nos. 1 and 2, 67–72 (Jan.–June 1965).

[3] Edmonds, J., Submodular functions, matroids, and certain polyhedra, Proc. of the Calgary Combinatorial Conference, Aug. 1969 (Gordon and Breach, New York, 1970), pp. 69–87.

[4] Edmonds, J. and Fulkerson, D. R., Transversals and matroid partitions, J. Res. Nat. Bur. Stand. (U.S.), **69**B (Math. and Math. Phys.), No. 3, 147–153 (July-Sept. 1965).

[5] Gale, D., Optimal assignments in an ordered set: an application of matroid theory, J. of Combinatorial Theory, **4**, 176–180 (1968).

[6] Hull, B., Two algorithms for matroids, Discrete Math. **13**, 121–128 (1975).

[7] Knuth, D. E., The Art of Computer Programming, Vol. 3, Sorting and Searching (Addison-Wesley, Reading, Mass., 1973).

[8] Minieka, E., Maximal network flows, matroids and matchings, J. Res. Nat. Bur. Stand. (U.S.), **79**B (Math. Sci.), No. 1, 59–62 (Jan.-June (1975)).

[9] Tutte, W. T., Introduction to the Theory of Matroids, (Elsevier, New York, 1971).

[10] Tutte, W. T., Lectures on matroids, J. Res Nat. Bur. Stand. (U.S.), **69**B (Math. and Math. Phys.) Nos. 1 and 2, 1–47 (Jan.-June 1965).

[11] Young, H. P., Equicardinal Matroids and Matroid Designs, Thesis (University of Michigan (Mathematics), 1970).

(Paper 80B3–448)