DETERMINING ALL MAXIMAL DATA SUBSETS

CONSISTENT WITH REVEALED PREFERENCE

by

M. Houtman* and J.A.H. Maks*

ABSTRACT

A complete 'branch-and-bound' algorithm to determine all minimal feedback node sets of a directed graph is presented. In economics the minimal feedback node sets of a directed graph have an interesting application in the revealed preference theory, which will be pointed out in this paper. The described method determines all maximal subsets of a finite data set, that are compatible with the strong axiom of revealed preference.

* The authors are grateful to W.J. Keller and an anonymous referee for helpful comments and the Netherlands Organization for the Advancement of Pure Research for financial support.

Current address: University of Groningen
Department of Economics
Tel. 050-114968
P.O. Box 800
9700 AV Groningen
The Netherlands

INTRODUCTION

The problem of finding one minimal feedback node set is applied in logical circuit design. Solutions for this problem are described in Guardabassi (1971), and Smith and Wallford (1975). In economics the minimal feedback node sets of a directed graph have an interesting application in the revealed preference theory. This application is described in section 1. Section 2 relates the application in economics to the minimal feedback node set problem in graph theory. The problem of determining all feedback node sets is solved in section 4 with a 'branch-and-bound' algorithm. In section 3 we give a short general description of the 'branch-and-bound' method that we use. Section 5, finally, contains the source text of the algorithm in the computer language PASCAL. Although the definitions and concepts of the sections 1,2 and 3 are relatively well known, it may be convenient to have them at hand together.

1. REVEALED PREFERENCE THEORY

In this section we define the problem in economic terms. Assume a given dataset (p_i, q_i), for periods i=1..n, where $p_i \in R^m$ is a price vector and $q_i \in R^m$ the corresponding quantity vector for m commodities, bought by the consumer with a budget $p_i'.q_i$ at period i.

The weak axiom of revealed preference

We define a preference relation R on q_i, i=1..n, by:

$$q_i \; R \; q_j \quad \iff \quad p_i'.q_i \geq p_i'.q_j \text{ and } q_i \neq q_j$$

An economic interpretation of this relation is the following. The consumer was able to buy q_j at period i. Because he bought q_i at this period, we may assume that he prefers q_i to q_j.
We call R consistent with the weak axiom of revealed preference if there is no pair q_i, q_j that satisfies:

$q_i \; R \; q_j$ and $q_j \; R \; q_i$

The strong axiom of revealed preference

We define the transitive closure R* of R as:

$q_i \; R^* \; q_j \quad <=>$

a sequence indices $i_1, i_2, .., i_r$ exists, that satisfies:

$$q_i \; R \; q_{i_1} \text{ and } q_{i_1} \; R \; q_{i_2} \; \ldots \text{ and } q_{i_r} \; R \; q_j$$

We call R consistent with the strong axiom of revealed preference if there is no pair q_i, q_j that satisfies:

$q_i \; R^* \; q_j$ and $q_j \; R^* \; q_i$

As is described in Afriat (1967), consistency with the strong axiom of revealed preference implies the existence of a time invariant strict quasi-convex nonsatiated utility function, which is in agreement with the data and utility maximizing behaviour for given prices.

Results of testing the axioms of revealed preference

Koo (1963, 1971), and Koo and Hasenkamp (1972) use consumer food panel data of 250 households to test empirically the strong axiom of revealed preference. They conclude that nearly every family made a relatively small number of inconsistent choices. Mossin (1972) tests the axioms of revealed preference with data based on consumers´ reports about weekly purchases of everyday commodities. He compares the individual demand functions with the mean demand function and concludes that the mean demand function agrees better with the axioms of revealed preference. He obtains an occurrence of inconsistent choices comparable with the results of Koo´s investigation.
Maks (1978, 1980, 1982, 1984), Landsburg (1981) and Varian (1982) investigate aggregate consumption data for the Netherlands, Germany, the United Kingdom and the United States. They find hardly any violations of the axioms of revealed preference. Maks (1984) partitions the data into commodity

groups and assumes separability. Testing these groups did result again in only a few violations of the axioms of revealed preference for some commodity groups.

The problem

The results of Varian and Landsburg, mentioned above, investigate the consistency of consumer data with the axioms of revealed preference. Koo determines the maximal number of periods in the data set, which is consistent with the strong axiom of revealed preference. Here we describe the solution of the following problem:

What are all smallest subsets of periods, for which the removement of all corresponding data points (p_i, q_i) from the original dataset results in a dataset which is consistent with the strong revealed preference axiom.

This problem will be solved in the remaining sections with the use of graph theory. Application of the solution to this problem may be found in Maks (1978, 1980, 1982, 1984).

2. MINIMAL FEEDBACK NODE SETS

The property of one smallest subset, which is a subsolution of the problem outlined in the previous section, is now described in graph theoretical terms. We may depict a graph of a strong preference relation R as a set of nodes q_i with arrows from q_i to q_j if q_i R q_j. Finding one smallest possible subset of the dataset (p_i, q_i), $i=1..n$, for which the removement results in consistency with the strong axiom of revealed preference is now described by:

Remove a least possible number of nodes q_i, with its arrows to and from q_i, such that there are no cycles in the remaining subgraph.

This is called the minimal feedback node set problem. The aim of our program is to find all minimal feedback node sets. For a rigorous description of this problem and its solution we need the following definitions as given in Carre (1979).

Graph

A graph $G = (X,U)$ consists of:

- a finite set $X = \{x_1,x_2,\ldots,x_n\}$ of elements called nodes
- a subset U of the cartesian product $X\times X$, the elements of which are called arcs

Graph of a binary relation

A graph $G = (X,U)$ of a binary relation R on X is defined by:

$$U \triangleq \{(x_i,x_j) \in X\times X \mid x_i \; R \; x_j\}$$

Incident arc, successor and predecessor

An arc $(x_i,x_j) \in U$, in a graph $G = (X,U)$ is said to be incident to x_j and incident from x_i. We call x_j a successor of x_i and x_i a predecessor of x_j.

Path

A path from x_{i_1} to x_{i_r} is a finite set of arcs of the form:

$$(x_{i_1},x_{i_2}),(x_{i_2},x_{i_3}),\ldots,(x_{i_{r-1}},x_{i_r})$$

Connectivity

When there is a path from x_i to x_j we call x_i connected with x_j. We denote this by the binary relation C:

$$x_i \; C \; x_j \iff x_i \text{ is connected with } x_j$$

When G is the graph of the binary relation R we have R* = C.

Strong connectivity

When $x_i \; C \; x_j$ and $x_j \; C \; x_i$, we say that x_i is strongly connected. We denote this by the binary relation S:

$$x_i \; S \; x_j \iff x_i \text{ is strongly connected with } x_j$$

The set of all strongly connected nodes is called a the maximal strongly connected set. A set of strongly connected nodes $Y \subset X$ is called a strongly connected component when Y satisfies:

$$x_i \; S \; x_j, \text{ for every pair } x_i, x_j \in Y$$

A maximal strongly connected component (MSSC) is a strongly connected component that is not contained in a larger strongly connected component.

Essential node

When the arc $(x_k, x_k) \in U$, we call x_k an essential node. The set of all essential nodes of a graph is called the maximal essential set.

Cycle and acyclic graph

A cycle is a path from a node x_i to itself. An acyclic graph is a graph that does not contain any cycles.

Subgraph

If we remove from a graph G = (X,U) a subset of its nodes, together with all the arcs incident to or from these nodes, we are left with a subgraph of G of the form:

$$G_Y \overset{\Delta}{=} (Y, U \cap (Y \times Y)), \text{ for } Y \subset X$$

Removing a node

We define the subgraph G_k obtained from $G = (X,U)$ by the removal of x_k as:

$$G_k \overset{\Delta}{=} G_{X-\{x_k\}}$$

Absorbing a node

Let G' be the graph obtained of the graph $G = (X,U)$ by joining each predecessor x_i of x_k to each successor x_j of x_k by an arc (x_i,x_j).

$$G' = (X, U \cup \{(x_i,x_j) \in X \times X \mid (x_i,x_k) \in U, (x_k,x_j) \in U\})$$

We define G_k' as the graph obtained from G' by removal of the node x_k. We say that G_k' is obtained from G by absorbing the node x_k.

Feedback node set

We call $F \subset X$ a feedback node set of $G = (X,U)$, when G_{X-F} is acyclic. We denote the set of all feedback node sets of G by $F(G)$. When the cardinality of $F \in F(G)$ is minimal in $F(G)$, we call F a minimal feedback node set. We denote the set of all minimal feedback node sets by $F^*(G)$.

3. SOLVING A PROBLEM WITH BACKTRACK PROGRAMMING

To understand how the problem of determining all minimal feedback node sets is solved we must have insight in the attack of solving a problem with backtrack programming.

Specialization and simplification rules

The general concept underlying backtrack programming is the application of specialization rules to the problem. A specialization rule creates several smaller versions of the problem, whose solutions solve the original problem. We may repeat applying this rule on the smaller problems we have created, until the starting problem is reduced to a collection of sub-problems whose solutions are immediately obtainable. A specialization rule which creates only one smaller version of the problem is called a simplification rule. The following rule is essential for backtrack programming:

At every stage we apply the specialization rule to one of the most recently created sub-problems.

This rule called depth-first search, may be implemented in a natural way in computer languages as ALGOL and PASCAL, which allow recursive procedure calls.

The ´branch-and-bound´ principle

Suppose we have a problem whose solution consists of a set of sub-solutions and we have a backtracking method to find them. If we are only interested in sub-solutions which satisfy some optimality criterium in this set of sub-solutions, the backtracking method can be made more efficient by applying the ´branch-and-bound´ principle in the following way. In the course of the search for sub-solutions, we keep record of the optimal sub-solutions yet discovered. Further we apply a bounding rule to find out if all sub-solutions of a sub-problem are not optimal. If this is the case, the exploration of this sub-problem is terminated.

4. OUTLINE OF THE ALGORITHM

To solve the problem of finding all minimal feedback node sets we first construct a backtracking algorithm to find all feedback node sets. Then we apply the ´branch-and-bound´ principle to find all minimal feedback node sets. The backtracking algorithm is constructed by using simplification and specialization rules. The 'branch-and-bound' algorithm is then constructed by using a bounding rule and an additional simplification rule.

The problem

Let $G = (X,U)$ be a graph and Y a set of nodes for which $X \cap Y = \emptyset$. We define the problem P as:

$$P(G,Y) \overset{\Delta}{=} \{F \cup Y \mid F \in F(G)\}$$

The starting problem

The starting problem of finding all feedback node sets of G is now defined by:

$$P(G,\emptyset) = F(G)$$

The trivial problem

The problem of a graph $G = (X,U)$, where $X = \emptyset$ is immediately solved by:

$$P(G,Y) = \{Y\}$$

Simplification rule

Every feedback node set of G must contain its essential nodes. So we remove all essential nodes of G and record them:

$$P(G,Y) = P(G_{X-E}, Y \cup E),$$

where $E \subset X$ is the maximal essential set of G

Specialization rule

We have for $G = (X,U)$ and $x_k \notin F \subset X$:

$$G_{X-F} \text{ is acyclic } \Leftrightarrow$$

$$(G_{X-F})'_k \text{ is acyclic, and } x_k \text{ is not essential}$$

Further we have for $x_k \notin F$:

$$(G_{X-F})'_k = (G'_k)_{X-F}$$

This implies:

$$F \text{ is a feedback node set of } G \text{ and } x_k \notin F \quad \Leftrightarrow$$

$$F \text{ is a feedback node set of } G'_k\text{, and } x_k \text{ is not essential}$$

This results in the following specialization rule. Choose a node $x_k \in X$ that is not an essential node. Decompose the problem P in a sub-problem with the chosen node removed and recorded and a sub-problem with the chosen node absorbed.

$$P(G,Y) = \{F \cup Y \mid F \in F(G),\ x_k \in F\} \cup \{F \cup Y \mid F \in F(G),\ x_k \notin F\}$$

$$= P(G_k, Y \cup \{x_k\}) \cup P(G'_k, Y)$$

The backtrack algorithm

We define the algorithm that solves the problem P(G,Y) by:

- Remove all essential nodes and record them.
- If there are any nodes left then choose one of these nodes and:
  - solve the problem with the chosen node removed and recorded,
  - solve the problem with the chosen node absorbed.
- If there is no node left we have a trivial problem that may be solved by using all recorded nodes.

At every stage of this backtrack algorithm the specialization rule creates two new problems which treats two smaller graphs. This ensures the reduction of the original problem by the algorithm in a collection of trivial problems.

Bounding rule

When G has at least one strongly connected node we have:

$$\min \{\text{card}(F) \mid F \in P(G,Y)\} > \text{card}(Y)$$

An additional simplification rule

The nodes which are not envolved in a cycle do not play a role in the search for minimal feedback node sets. Hence we may simplify the search by removing these nodes:

$$F^*(G) = F^*(G_{X-S}),$$

where $S \subset X$ is the maximal strongly connected set of G.

The 'branch-and-bound' algorithm

We define the algorithm that solves the problem of finding all minimal feedback node sets of G by:

- Search for all feedback node sets with the backtracking algorithm.
- Remove all nodes not involved in a cycle before specializing each problem.
- Record only the sub-solutions of the smallest cardinality yet found.
- Use the bounding rule to decide if a sub-problem has no optimal sub-solutions and does not need to be solved.

A refinement

When there are several MSCC's in a graph G, it is possible to simplify the search for feedback node sets:

$$F^*(G) = \{\bigcup_{i=1}^{r} F_i \mid F_i \in F^*(G_{C_i})\},$$

where C_i, i=1..r, are all the MSCC's of G

This rule may be implemented as a specialization rule in the algorithm, but we can't use it when there is only one MSCC. Then we have to apply the original specialization rule. An efficient MSCC identification algorithm is described by Tarjan (1972).

5. PASCAL SOURCE TEXT OF THE ALGORITHM

Here we list the source text of the algorithm in PASCAL. Comments are inserted to clarify the meaning of the several procedures and variables. The procedure warshall we uses the algorithm given by Warshall (1962) to compute the transitive closure of a graph. The last given specialization rule, which treats all MSCC's of a graph, is not implemented.

```
TYPE
   node        = 1..maxnode;
   (* Node-index i for xi; xi is identified with its index i *)
   nodeset     = SET OF node;
   (* Set of nodes xi *)
   successors = ARRAY [node] OF nodeset;
   (* Used to store all the successors of each node in a nodeset *)
   graph       = RECORD
                     x   : nodeset;
                     suc : successors;
                  END;
   (* Used to store a graph *)
   solutions = FILE of nodeset;
   (* Used to store feedback node set solutions *)

FUNCTION firstnode (x : nodeset) : node;
(* This procedure returns the first node of the nodeset *)
VAR i : INTEGER;
BEGIN
   i := 1;
   WHILE NOT (i IN x) DO i := i + 1;
   firstnode := i
END;

PROCEDURE connect`(VAR g : graph; k : node);
(* This procedure connects each predecessor of xk with all its successors *)
VAR i : INTEGER;
BEGIN
   WITH g DO
     FOR i := 1 TO maxnode DO
       IF i IN x THEN
         IF k IN suc[i] THEN suc[i] := suc[i] + suc[k]
END;
```

```
PROCEDURE findmfb (Var g : graph; VAR sol : solutions);
(*
   This procedure writes all minimal feedback node sets of g on the file sol
*)
VAR bound : INTEGER;
    (* The minimal cardinality of the feedback node sets yet found *)

   PROCEDURE problem (g : graph; y : nodeset);
   (*
     This procedure solves P(G,Y). The specialization rule is executed by
     recursive procedure calls.
   *)
   LABEL 1;
         (* Exit label *)
   VAR k : node;
       (* The node used in the specialization rule *)
   BEGIN
     simplify (g, y);
     (* Simplification of the problem *)
     IF g.x <> [] THEN WITH g DO
       (* The trivial sub-problem with g.x = [] has solution {Y} *)
       BEGIN
         IF CARD (y) >= bound THEN GOTO 1;
         (* The bounding rule *)
         k := firstnode (x);
         (* Choose a node xk in X *)
         x := x - [k];
         problem (g, y + [k]);
         (* Solve the problem with node xk removed and recorded *)
         connect (g, k);
         problem (g, y);
         (* Solve the problem with node xk absorbed.                  *)
       END
       (* Specialization of the problem *)
     ELSE
       BEGIN
         IF CARD (y) > bound THEN GOTO 1;
         (* The 'branch-and-bound' principle *)
         IF CARD (y) < bound THEN
         (* Is this solution better then all other solutions? *)
           BEGIN
             bound := CARD (y);
             rewrite (sol);
           END;
           (* There is a new bound and forget the old solutions *)
         WRITE (sol, y);
         (* Write the solution y on the file sol *)
       END;
       (* record solution *)
   1:END;

BEGIN
   bound := maxnode;
   (* Bound is assigned to its maximum value *)
   problem (g, []);
   (* The starting problem *)
END;
```

```
PROCEDURE warshall (VAR g : graph);
(*
   This procedure computes the transitive closure of a graph. xj is a
   successor of xi in the transitive closure if there exist a path from
   xi to xj.
*)
VAR i : INTEGER;
BEGIN
   WITH g DO
     FOR i := 1 TO maxnode DO
        IF i IN x THEN connect (g, i)
END;

PROCEDURE findme (VAR g : graph; VAR me : nodeset);
(* This procedure identifies the maximal essential set me of g *)
VAR i : INTEGER;
BEGIN
   me := [];
   WITH g DO
     FOR i := 1 TO maxnode DO
        IF i IN x THEN
           IF i IN suc[i] THEN me := me + [i]
END;

PROCEDURE findmsc (g : graph; VAR msc : nodeset);
(* This procedure identifies the maximal strongly connected set msc of g *)
VAR i : INTEGER;
BEGIN
   warshall (g);
   (* Compute the transitive closure of g *)
   findme (g, me);
   (*
     The strongly connected nodes of a graph are the essential nodes of its
     transitive closure
   *)
END;

PROCEDURE simplify (VAR g : graph; VAR y : nodeset);
(* This procedure executes the simplification rules *)
VAR me,msc : nodeset;
   (* The maximal essential and strongly connected node sets *)
BEGIN
   WITH g DO
     BEGIN
        findme (g, me);
        x := x - me;
        y := y + me;
        (* Remove all essential nodes and record them *)
        findmsc (g, msc);
        x := msc;
        (* Remove all nodes not involved in a cyclepath *)
     END
END;
```

SUMMARY

The described non-parametric method determines all subsets with a maximum number of elements of a finite data set, that are compatible with the strong axiom of revealed preference. This goal is obtained by deleting data points to obtain subsets with consistent choices. The strong axiom of revealed preference can be used to depict a graph as a set of nodes q_i with arcs from q_i to q_j if $p_i'.q_i \geq p_i'.q_j$ and $q_i \neq q_j$. Stated in these terms the problem is:

Find all sets with a minimal number of nodes for which the removement of these nodes q_i, with its arcs from and to q_i, results in an acyclic subgraph.

The computer program to solve this problem uses backtrack and 'branch-and-bound' principles and Warshalls' algorithm.

REFERENCES

Afriat, S.N. (1967), The Construction of Utility Functions from Expenditure Data, International Economic Review **7**, 67-77

Carre, B. (1979), Graphs and Networks, Oxford

Guardabassi, G. (1971), A Note on Minimal Essential Sets, IEEE Transactions on Circuit theory **CT-18**, 557-560.

Koo, A.Y.C. (1963), An Empirical Text of Revealed Preference Theory, Econometrica **33**, 646-664

---------- (1971), Revealed Preference - A Structural Analysis, Econometrica 39, 89-97

----------, and G. Hasenkamp (1972), Structure of Revealed Prefrence - Some Preliminary Evidence, Journal of Political Economy **80**, 724-744

Landsburg, S.E. (1981), Taste Change in the United Kingdom, 1900-1955, Journal of Political Economy **89**, 92-104

Maks, J.A.H (1978), Consistency and Consumer Behaviour in the Netherlands, 1951-1977, European Economic Review **11**, 343-362

---------- (1980), Empirical Preference Orderings and Applied Demand Analysis, Dissertation, University of Groningen

---------- (1982), A Supplementary Method for Consumer Demand Analysis and Welfare Comparision Applied to United Kingdom and West German Data Sets, Kwantitatieve Methoden **5**, 56-77

---------- (1984), Consumer Behaviour in the Netherlands, 1951-1977, A Nonparametric Approach, Presented at the 1984 European Meeting of the Econometric Society in Madrid, forthcoming in the European Economic Review

Mossin, A. (1972), A Mean Demand Function and Individual Demand Functions Confronted with the Weak and Strong Axioma of Revealed Preference: An Empirical Test, Econometrica **40**, 177-192

Smith, G.W. and Walford, R.B. (1975), The Identification of a Minimal Vertex Set of a Directed Graph, IEEE Transactions on Circuits and Systems **CAS-22**, 9-14

Tarjan, N. (1972), Depth-first Search and Linear Graph Algorithms, SIAM Journal Computing **1**, 146-160

Varian, H.R. (1983), Nonparametric Tests of Consumer Behaviour, The Review of Economic Studies **50**, 99-110

Warshall, S. (1962), A Theorem on Boolean Matrices, Journal of the American Association for Computing Machinery **9**, 11-12