Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090102

Fundamental Products and Autonomous Sets: An Algorithmic Approach

Eugenio M. Fedriani and Ángel F. Tenorio[*]

Departamento de Economía, Métodos Cuantitativos e Historia Económica, Universidad Pablo de Olavide, Ctra, Utrera, km 1, 41013-Sevilla (Spain)

Received: 4 Mar. 2014, Revised: 4 Jun. 2014, Accepted: 5 Jun. 2014
Published online: 1 Jan. 2015

Abstract: The main goal of this paper is to analyze an economy through its symmetric input-output table by using Graph Theory. After providing a definition of the characteristic digraph for the economy, the authors give some properties and algorithms to characterize and compute the fundamental products and, later, the autonomous sets of the studied economy. Six of these algorithms are explained in detail and formulated; the best choices are also implemented with the computational package Mathematica, compared in terms of efficiency, and applied to real input-output matrices from Greece.

Keywords: Input-Output Analysis, fundamental product, autonomous set, indecomposable matrix, Graph Theory

1 Introduction

Graphical representations have helped scientists to better understand reality and its possible connection with the corresponding models. In Economics, Graph Theory is mostly used to complement Input-Output Analysis and Structural Analysis (see [1,8], for instance). Specifically, the technical coefficients matrix of an economy can be translated into the adjacency matrix of a digraph, which can therefore be univocally associated with the economy.

In this paper, we collect some preliminary ideas about fundamental products and autonomous sets sketched in a previous contribution [3]. In that extended abstract, by using topological graphs, two characterizations and three computational approaches were suggested (but not developed or applied). Here we give some additional results (with their correspondent proofs) as well as significant improvements in the algorithms and explicit implementations for the most efficient ones.

The paper is structured in five sections in addition to this short introduction. First, we recall some preliminary definitions and results which will be used later. In the next section, theoretical results and characterizations of fundamental products and autonomous sets in terms of Graph Theory are introduced and proved. Right afterwards, some algorithms are explained, implementing four of them in the computation system Mathematica (©1988–2013, trademark of Wolfram Research, Inc.). Then, as an example, we apply the algorithms to analyze the last, Greek input-output matrices. Finally, some ideas for future research are presented.

2 Preliminaries

For a general overview on Graph Theory and Input-Output Analysis, the reader can consult the classical works [5] and [6,7], respectively. We will restrict our attention to the concepts more closely related to our final goal.

2.1 Graph concepts

A *digraph* $D = (V, E)$ consists of a non-empty finite set V of elements and a finite set E of ordered pairs of these elements. The set V is called *vertex-set* of D and its elements are the vertices of D. Besides, the set E is called *arc-set* of D and its elements are the arcs of D. If v and w are two vertices of D, the arc (v, w) is said to be the *arc from v to w*.

In a digraph D, a *directed walk* (of length n) from v_1 to v_{n+1} is a sequence of arcs $v_1v_2, v_2v_3, \ldots, v_nv_{n+1}$ in D.

[*] Corresponding author e-mail: aftenorio@upo.es

Moreover, a directed walk is called (n-order) *directed path* from v_1 to v_{n+1} when $v_i \neq v_j$ for $i \neq j$; and a *directed cycle* starts and ends in the same vertex ("directed path" where $v_1 = v_{n+1}$).

A *directed-in-tree rooted in the vertex* v is a tree in which there exists a (unique) directed path in the tree from the vertex v to any other vertex. A *directed-out-tree rooted in the vertex* v is a tree in which there exits a (unique) directed path in the tree from any vertex to the vertex v. In both cases, the vertex v is called the *root vertex*. A rooted directed tree is said to be *spanning* when it includes every vertex of the digraph D.

2.2 Input-output analysis

Given an economy $\mathscr{E}$ with n productive sectors in the hypothesis of [6] (each sector produces only one good, not allowing for secondary production), these sectors can simultaneously act as both producers and consumers, because an arbitrary sector i sells its product (outputs of i) to other sectors and buys the products of other sectors (inputs of i) to generate its own product. In this way, the *technical coefficient* a_{ij} shows the value of the input purchased by sector j to sector i per monetary unit of output in sector j ($i,j \in N = \{1,\dots,n\}$). If we arrange these coefficients in a square $n \times n$ matrix, we obtain the so-called *structural matrix* or *technical coefficients matrix* of the economy $\mathscr{E}$. From now on, this matrix is denoted by $A = (a_{ij})$ and provides a quantitatively determined outlook of the internal structure of the economy $\mathscr{E}$. Indeed, these matrices allow to compare two economies between themselves, or even the same economy in two different time periods, because the i^{th} row of the technical coefficients matrix indicates the distribution of the total sales of sector i between the remaining sectors in $\mathscr{E}$. Analogously, the j^{th} column represents the purchases done by sector j to each sector in order to produce its good.

In Economics, it is often useful to know which products or sectors take part in the production of a selected set or all the products in the studied economy [10,9]. This can be studied through the technical coefficients matrix (or, analogously, the intersectorial flow table), by considering the concepts of fundamental product and autonomous set. We now recall these concepts.

Let $N = \{1,\dots,n\}$ be the index-set formed by all the sectors (or, equivalently, goods) in the economy $\mathscr{E}$. The product $i \in N$ is a *fundamental product* when it (directly or indirectly) takes part in the production of all the products (including itself). Mathematically, a set $B \subseteq N$ is said to be *autonomous* if $a_{ji}=0$, $\forall i \in B$, $\forall j \in N \setminus B$. The economical interpretation of an autonomous set B is that no sector out of B sells its product to any sector in B. Here we mean that a sector i *sells* its product to another sector j when the technical coefficient a_{ij} is nonzero.

Starting from the autonomous sets of the technical coefficients matrix A, the fundamental products can be obtained in two steps. First, the autonomous sets are computed and the elements of the minimal autonomous set are the candidates to be the fundamental products. Then, one of these elements is individually studied to determine whether it is a fundamental product. If that is the case, the fundamental products are all the ones in the minimal autonomous set. In the opposite, there is no fundamental product.

3 Connection between graphs and economies

The fundamental products in an economy are usually determined by using the autonomous sets of the economy. In [3], we suggested another approach to this concept by considering topological properties which can be associated with the technical coefficients matrix. In fact, we can determine the fundamental products, directly from the technical coefficients matrix A of the economy $\mathscr{E}$ (or the transaction flow table), by using Graph Theory. First, a digraph can be defined as follows: the vertex-set V is the proper set N and the arc-set E is formed by the pairs $(i,j) \in N \times N$ such that $a_{ij} \neq 0$. Hence, there exists an arc from sector i to sector j if and only if sector i sells its product to sector j. The digraph $D(\mathscr{E}) = (V,E)$ is said to be *associated with the economy* $\mathscr{E}$. The following result translates the condition of fundamental product into a property in the digraph $D(\mathscr{E})$ associated with the economy $\mathscr{E}$, what will provide both a new viewpoint and an efficient way of computing.

Proposition 3.1. Let i be a productive sector (and its good) of the economy $\mathscr{E}$. Then i is a fundamental product if and only if there exists a directed path in $D(\mathscr{E})$ from i to j, for all $j \in N$.

Proof. If i is a productive sector, then it takes part directly or indirectly in the production of each sector j. In the first case, $a_{ij} \neq 0$; i.e., there exists the arc (i,j) in the digraph $D(\mathscr{E})$. In the other case, the sector i sells its product to a sector i_1 (so $a_{ii_1} \neq 0$), i_1 sells to another sector i_2 ($a_{i_1i_2} \neq 0$) and so on, reaching a sector i_k such that $a_{i_kj} \neq 0$; this finite sequence of nonzero values is equivalent to the existence of the directed path $(i,i_1),(i_1,i_2),\dots,(i_{k-1},i_k),(i_k,j)$ in $D(\mathscr{E})$. $\square$

As an immediate consequence of Proposition 3.1, other fundamental products can be obtained starting from a fundamental product found in the digraph $D(\mathscr{E})$, in virtue of the following:

Corollary 3.2. If $i \in N$ is a fundamental product of $\mathscr{E}$, the product $j \in N$ is fundamental if and only if there exists a directed path from j to i in $D(\mathscr{E})$.

Proof. If j is a fundamental product, Proposition 3.1 implies the existence of a directed path from j to i. The converse is also true, because such a directed path reflects that sector j takes part in the production of sector i, which in turn takes part in the production of all the sectors (including j). □

Note that, in Corollary 3.2, since i is a fundamental product, there is always a directed cycle containing vertex j. Next, another characterization is given for fundamental products, by using spanning directed-in-trees:

Proposition 3.3. Let $i \in N$ be a sector (and its good). Then i is a fundamental product if and only if there exists a spanning directed-in-tree rooted in i and a directed cycle containing the vertex i.

Proof. If $i \in N$ is a fundamental product, then the digraph $D(\mathscr{E})$ is connected and contains a spanning directed-in-tree rooted in i. Firstly, let us check that it is connected: given two sectors i_1 and i_2, due to Proposition 3.1, we can design a walk between i_1 and i_2, by joining the path between i and i_1 and that between i and i_2; if this walk is not a path, we can obtain a path between i and i_2 by removing all the cycles inside the walk. Secondly, let us find a spanning directed-in-tree in $D(\mathscr{E})$ by using a depth-first search, starting from the fundamental product i, which will be its root. Additionally, a directed cycle containing the vertex i is found due to Proposition 3.1.

Conversely, the spanning directed-in-tree rooted in i implies the existence of a directed path from i to any $j \in N$ with $j \neq i$. Besides, the directed cycle containing the vertex i is a directed path from i to i. Hence, i is a fundamental product due to Proposition 3.1. □

The interest of the previous results is that there exist well-known algorithms which search for paths, cycles, directed-in-trees, etc. On the other hand, this viewpoint allows the appearance of useful results connecting Graph Theory and input-output models. Moreover, we are about to give some examples of how the intuitiveness of graphs eases the the appearance of results involving autonomous sets and decomposable matrices (which have at least one autonomous set different from the index set).

Lemma 3.4. The set $S \subsetneq N$ is autonomous if and only if there are not any arcs from a vertex of $N \setminus S$ to a vertex of S.

Proof. It is a direct consequence of the definition of autonomous set. □

Corollary 3.5. The union of autonomous sets is an autonomous set. □

Lemma 3.4 is also useful joint with Proposition 3.6 and the concept of decomposable economy (i.e., decomposable technical coefficients matrix).

Proposition 3.6. If there exists a non-fundamental product, then $\mathscr{E}$ is decomposable.

Proof. As there is a non-fundamental product s_1 in $\mathscr{E}$, there does not exist a directed path from sector s_1 to another sector s_2. Now we consider a maximal directed-out-tree rooted in s_2, whose vertex-set is denoted by V_T. The set $N \setminus V_T$ is non-empty because it contains s_1. Besides, for every vertex in $N \setminus V_T$, there does not exist an arc from it to any vertex in V_T (otherwise, such a vertex would belong to V_T). Therefore, V_T is an autonomous set different from N. □

The following characterizations allow us to improve the algorithms introduced in [3], as we will see later.

Theorem 3.7. The following statements are equivalent in the economy $\mathscr{E}$:

1. The technical coefficients matrix is decomposable.
2. The index-set can be split into two disjoint, non-empty sets such that there are not any arcs from the first set to the second.
3. There exist, at least, two strongly connected components in $D(\mathscr{E})$.

Proof. We prove the following chain of conditions: 1⇒2⇒3⇒1.

$1 \Rightarrow 2$: It is obvious due to Lemma 3.4.
$2 \Rightarrow 3$: We can find a non-empty subset $S \subsetneq N$ with no arc from $N \setminus S$ to S. Every digraph has, at least, one strongly connected component; by *reductio ad absurdum*, let us suppose that it is unique. Then there exists a directed path from a given vertex $i \in N \setminus S$ to any other vertex $j \in S$. This directed path must contain an arc from a vertex in $N \setminus S$ to another in S, which contradicts our hypothesis.
$3 \Rightarrow 1$: Let S be the vertex-set of a strongly connected component of $D(\mathscr{E})$. The remaining vertices of $D(\mathscr{E})$ have to belong to other components different from S. So, there are no arcs from $N \setminus S$ to S. In virtue of Lemma 3.4, $S \subsetneq N$ is autonomous and, hence, A is decomposable. □

Theorem 3.8. The following statements are equivalent in the economy $\mathscr{E}$:

1. The technical coefficients matrix is totally decomposable.
2. The index-set can be split into disjoint, non-empty subsets such that there are not arcs between vertices belonging to different subsets.
3. There exist, at least, two connected components in $D(\mathscr{E})$.

Proof. We prove the following chain of implications: 1⇒2⇒3⇒1.

$1 \Rightarrow 2$: The matrix is totally decomposable if and only if there exist two non-empty, autonomous sets, $S_1, S_2 \subset N$, such that $S_1 \cup S_2 = N$, $S_1 \cap S_2 = \varnothing$, and $a_{ij} = a_{ji} = 0$ for all $i \in S_1$ and for all $j \in S_2$. This last condition means that the digraph $D(\mathscr{E})$ does not contain either the arc (i, j) or (j, i) for all $i \in S_1$ and for all $j \in S_2$.

$2 \Rightarrow 3$: We can find a non-empty subset $S \subsetneq N$ such that there does not exist any arc between S and $N \setminus S$. If we suppose (*reductio ad absurdum*) the existence of only one connected component in $D(\mathscr{E})$, then there exists a path between two given vertices $i \in S$ and $j \in N \setminus S$. This path must contain an arc between a vertex in S and another one in $N \setminus S$. But this is impossible under our hypothesis.

$3 \Rightarrow 1$: Let S be the vertex-set of a connected component of $D(\mathscr{E})$. Then $N \setminus S$ consists of the union of the connected components different from S. Therefore, there are not arcs between $N \setminus S$ and S. Due to Lemma 3.4, both S and $N \setminus S$ are autonomous sets and, hence, the technical coefficients matrix A is totally decomposable. $\square$

Proposition 3.9. Let C be a connected component of the digraph $D(\mathscr{E})$. The vertices belonging to C form an autonomous set of $\mathscr{E}$.

Proof. Let V_C be the vertex-set of C. If j belongs to $N \setminus V_C$ and i to V_C, then there are not directed paths from i to j. Particularly, there are not arcs from i to j; that is, $a_{ji} = 0$. Therefore, V_C is an autonomous set. $\square$

Note that there may exist autonomous sets which are neither connected components nor the union of connected components. This fact is due to the possible existence of a chain of inclusions for some autonomous sets.

Proposition 3.10. If the digraph $D(\mathscr{E})$ is disconnected, then there is not any fundamental product in $\mathscr{E}$.

Proof. Since $D(\mathscr{E})$ is disconnected, there exist, at least, two connected components, C_1 and C_2, in $D(\mathscr{E})$. Let $i \in C_1$ and $j \in C_2$ be two sectors in $\mathscr{E}$. Then, there do not exist any paths between i and j, and i cannot be a fundamental product of $\mathscr{E}$. This reasoning is valid for every sector i. $\square$

4 Specific algorithms

4.1 Computing the fundamental products

In this subsection, we indicate two different algorithms to obtain the fundamental products of a given economy $\mathscr{E}$ (without knowing its autonomous sets). Both algorithms need as input data the technical coefficients matrix of $\mathscr{E}$ (or its intersectorial flow table), and the output data are all the fundamental products in $\mathscr{E}$.

The first algorithm (ALG1) consists of detecting one fundamental product from the digraph $D(\mathscr{E})$, and the remaining fundamental products are obtained starting from the fundamental product already known:

Step 0**.** Obtain a fundamental product by using an algorithm which obtains a rooted spanning directed-in-tree; the root i is a fundamental product of $\mathscr{E}$ when there also exists a directed cycle containing the root, in virtue of Proposition 3.3.

Step k **(**$k = 1, \dots, n$**).** Fixed another sector j in $\mathscr{E}$, check if there exists a directed path from j to i in the digraph $D(\mathscr{E})$. j is a fundamental product if there exists such a path, in virtue of Corollary 3.2.

Next, we show the implementation of algorithm ALG1 in Mathematica code (at least, it works from version 5 to version 9). To run the following implementations over Mathematica 5.2, `DiscreteMath`Combinatorica`` package has to be activated; later versions of the program also admit the command `<<Combinatorica``, instead.

```
(*"a" is the technical coefficients matrix*)
numVert:=Dimensions[a][[1]]
b=Table[0,{i,numVert},{j,numVert}];
For[i=1,i<=numVert,i++,
 For[j=1,j<=numVert,j++,
  If[a[[i,j]]!=0,b[[i,j]]=1,b[[i,j]]=0]]]
<<Combinatorica‘;
fundpro:={}
g:=FromAdjacencyMatrix[b,Type->"Directed"]
cycleList=DeleteDuplicates[Flatten[ExtractCycles[g]]];
For[i=1,i<=numVert,i++, dijkstra=Dijkstra[g,i];
 If[Norm[dijkstra[[2]],1]<Infinity,
  If[MemberQ[cycleList,i]==True, AppendTo[fundpro,i];
   Print[i," is a fundamental product."]]];
 If[fundpro!={},Break[]]]
If[fundpro=={},
 Print["There is no fundamental product."],
 min=fundpro[[1]];
 For[i=min+1,i<=numVert,i++,
  If[Dimensions[ShortestPath[g,i,min]][[1]]>1,
   AppendTo[fundpro,i];
   Print[i," is a fundamental product."]]]]
(*fundpro contains all the fundamental products*)
```

Although this first algorithm is directly based on Graph Theory and potentially more intuitive and insightful than others, the direct implementation of notions about graphs is usually harder than the implementation of essentially equivalent algorithms which use Matrix Algebra. In this sense, a simplification can be achieved by taking into consideration that the technical coefficients matrix A of the economy $\mathscr{E}$ with n productive sectors works as the adjacency matrix of the digraph $D(\mathscr{E})$ associated with this economy. Hence, the second algorithm proposed here is based on the well-known fact that each nonzero term a'_{ij} in the matrix power $A^k = (a'_{ij})$ indicates the existence of k-order directed walks from i to j, where $1 \leq k \leq n$. As each fundamental product of the economy $\mathscr{E}$ has been characterized with the existence of directed paths from its corresponding vertex in the digraph $D(\mathscr{E})$, in virtue of Proposition 3.1(and the fact that each walk between two vertices contains a path between them), the rows of nonzero terms in the matrix sum $M = \sum_{k=1}^{n} A^k$ are in one-to-one correspondence with the fundamental products of $\mathscr{E}$. Hence, the algorithm (ALG2) can be described as follows:

Step 1**.** Compute A^2 and add it to A.

Step $k-1$ **(**$k = 3, \dots, n$**).** Fixed k, compute A^k and add it to the sum obtained in the previous step.

Step *n*. Let M be the matrix defined as $M = \sum_{k=1}^{n} A^k$, and the fundamental products of $\mathscr{E}$ are determined by the subscripts of rows whose terms are all nonzero.

This algorithm can be binarized in order to reduce the size of computations. In such a case, the nonzero terms in each matrix are replaced by “1” in each step, and the algorithm (ALG2’) may stop when two consecutive power matrices are equal (see [3] if more details about this process are needed).

To conclude this section, we show our easy implementation of algorithm ALG2’ in Mathematica code. Let us recall that the input data are given by the technical coefficients matrix.

```
techmatrix=Input["Insert the technical coefficients
     matrix:"]
rank:=Dimensions[techmatrix]
n:=rank[[1]] (*n is the number of sectors*)
b=Table[0,{i,n},{j,n}] (*b is the binary matrix*)
For[i=1,i<=n,i++,
 For[j=1,j<=n,j++,If[techmatrix[[i,j]]!=0,b[[i,j]]=1]]]
(*We start step 1; c, d, and e are auxiliary matrices*)
c:=b
d:=Table[0,{i,n},{j,n}] (*We will add all the power
    matrices in d*)
e:=Table[0,{i,n},{j,n}]
For[k=1,k<=n,k++,e=c;c=c.b;
 For[i=1,i<=n,i++,
  For[j=1,j<=n,j++,If[c[[i,j]]!=0,c[[i,j]]=1]]];
 If[e==c,d=e;Break[],d=e+c]]
(*The fundamental products are obtained from d:*)
fundpro:={} For[i=1,i<=n,i++,r:=0;
 For[j=1,j<=n,j++,If[d[[i,j]]==0,r=r+1]];
 If[r==0, AppendTo[fundpro,i];
  Print[i," is a fundamental product of the
      economy."]]]
Print[fundpro," is the set formed by all the
  fundamental products."]
```

4.2 Computing the autonomous sets

In this subsection we suggest some algorithms to obtain the autonomous sets in a given economy $\mathscr{E}$ with n sectors. In these algorithms, the input data are given by the technical coefficients matrix A, and the outputs are all the autonomous sets in the economy. The first algorithm (ALG3) is based on the definition of autonomous set itself. This algorithm directly computes all the autonomous sets in an economy. We sum up the steps of this coarse algorithm for a given economy:

Step *k*. Fixed $k \in N = \{1,\ldots,n\}$, consider the subsets of N with cardinal k and determine which of them are autonomous by using the definition.

However, the total number of candidate sets is 2^n, too high in most cases. An alternative of the previous algorithm can be considered if the fundamental products in the economy have been computed before (and there exists at least one). As every fundamental product must belong to all the autonomous sets, then we can skip some subsets of N if we know the fundamental products (and they are inputs for this algorithm).

Now we show how to implement (in Mathematica) the alternative algorithm which has just been commented (ALG3’). The source code shown in the previous section has to be previously run to compute all the fundamental products. Note that this algorithm also works properly when the set of fundamental product is empty.

```
<<Combinatorica‘
(*In the newest versions, one can call the Combinatorica
Package*)
dimfundpro:=Part[Dimensions[fundpro],1]
(*dimfundpro is the number of fundamental products*)
autset:={fundpro}
(*The set constituted by all the fundamental products
is an autonomous set:*)
Print[fundpro," is an autonomous set."]
totalset:=Table[i,{i,n}]
(*totalset is the index set*)
For[k=dimfundpro+1,k<=n,k++,l=KSubsets[totalset,k];
 numsub=n!/((n-k)!*k!);
 (*l will be each candidate set*)
 For[j=1,j<=numsub,j++,
  If[Table[MemberQ[Part[l,j],Part[fundpro,i]],{i,
   dimfundpro}]==Table[True,{i,dimfundpro}],
    minusset=Complement[totalset,l[[j]]];
    z=0;
    For[v=1,v<=n-k,v++,
     For[u=1,u<=k,u++,
      If[techmatrix[[minusset[[v]],Part[l[[j]],u]]]!=0,
       z=z+1]]];
    If[z==0,AppendTo[autset,l[[j]]];
      Print[l[[j]]," is an autonomous set."]]]]]
```

If the number of sectors is too large, some improvements in the efficiency of the algorithm may be welcome, although the reading will turn slightly more complicated (ALG3”):

```
If[fundpro=={},
 For[k=dimfundpro+1,k<=n,k++,l=KSubsets[totalset,k];
  Print["Subsets of size ",k,":"];numsub=n!/((n-k)!*k!);
  For[j=1,j<=numsub,j++,
   If[Table[MemberQ[Part[l,j],Part[fundpro,i]],
      {i,dimfundpro}]==Table[True,{i,dimfundpro}],
    minusset=Complement[totalset,l[[j]]];z=0;
    For[i=1,i<=dimPosition,i++,
     If[{MemberQ[l[[j]],nZposition[[i,2]]],
        MemberQ[minusset,nZposition[[i,1]]]}
        =={True,True},
      z=z+1;Goto[escape]]]]];
   Label[escape];
   If[z==0,AppendTo[autset,l[[j]]];
    Print[l[[j]]," is an autonomous set."]]]],
 addIndices:=Complement[totalset,fundpro];
 dimIndices:=Dimensions[addIndices][[1]];
 For[k=1,k<=dimIndices,k++,
  l=KSubsets[addIndices,k];
  numsub=dimIndices!/((dimIndices-k)!*k!);
  (*Dimensions[l][[1]]*)
  Print["Subsets of size ",k+dimfundpro,":"];
  For[j=1,j<=numsub,j++,
   autCandidate=Join[fundpro,l[[j]]];
   minusset=Complement[totalset,autCandidate];z=0;
   For[i=1,i<=dimPosition,i++,
    If[{MemberQ[autCandidate,nZposition[[i,2]]],
       MemberQ[minusset,nZposition[[i,1]]]}==
       {True,True},
     z=z+1;Goto[escape]]];
   Label[escape];
   If[z==0,AppendTo[autset,autCandidate];
    Print[autCandidate," is an autonomous set."]]]]]
```

This algorithm works fine and quickly when there exist fundamental products in the economy $\mathscr{E}$, but the computing time increase meaningfully when there are not fundamental products in the economy (the worst case possible and quite rare). In such a case, other algorithms (even graphical) can be given by using the digraph associated with the economy and previous results (from Lemma 3.4 to Proposition 3.9).

Let us try to summarize two new algorithms supported by Graph Theory. In both cases, the first step consists of computing all the strongly connected components in the digraph associated with the economy. As a consequence of Lemma 3.4, if one vertex lies in an autonomous set, then all the vertices in its same strongly connected component belong to such an autonomous set. After this preliminary step (renaming the sectors which collapse into a vertex), the strategies are different. Nevertheless, both algorithms have two more aspects in common: they are based on the search of root vertices, and their last step is computing all the possible unions of sets obtained through the algorithms (according to Corollary 3.5).

On the one hand, ALG4 pays special attention to the vertices (or strongly connected components) such that there does not exist an arc from any other vertex to them; each of these vertices is usually called a *source*. Obviously, the sources are autonomous sets themselves. Moreover, when incorporating (in all the different ways possible) the vertices adjacent to (or successor of) the sources, more autonomous sets can be obtained. Finally, the algorithm would compute the unions of any number of autonomous sets already detected. This algorithm is quite intuitive, what is a pedagogical advantage, but it presents the difficulty caused by the excessive number of different ways to expand the autonomous sets from the initial sources.

On the other hand, there are usually vertices (or strongly connected components) such that there are not arcs from them (to other vertices). These vertices are the so-called *sinks* and they provide us a recurrent and more efficient way of computing the autonomous sets (ALG5). Firstly, the total set is autonomous. Secondly, each connected component is an autonomous set (see Proposition 3.9). Thirdly, when removing the sink vertices (or strongly connected components), some new connected components may arise, and they also correspond to autonomous sets in the economy. Finally, the union of autonomous sets is autonomous, too. An implementation of ALG5 would be as follows:

```
(*"a" is the technical coefficients matrix, again*)
numVert=Dimensions[a][[1]];
b=Table[0,{i,numVert},{j,numVert}];
For[i=1,i<=numVert,i++,
 For[j=1,j<=numVert,j++,
  If[a[[i,j]]!=0,b[[i,j]]=1,b[[i,j]]=0]]]
<<Combinatorica`;
g=FromAdjacencyMatrix[b,Type->"Directed"];
g=RemoveSelfLoops[g];
cFC=StronglyConnectedComponents[g];
nList=Dimensions[cFC][[1]];
listVertCFC=Table[i,{i,1,nList}];
posVertCFC=
  Table[{{1/2*Cos[i*2Pi/nList],1/2*Sin[i*2Pi/nList]}},
  {i,1,nList}];
edges=Edges[g];
edgesAux={};
For[i=1,i<=nList-1,i++,
  For[j=i+1,j<=nList,j++,
   res1=CartesianProduct[cFC[[i]],cFC[[j]]];
   res2=Intersection[res1,edges];
   If[res2!={},edgesAux=Join[edgesAux,{{{i,j}}}]]]];
For[i=1,i<=nList-1,i++,
  For[j=i+1,j<=nList,j++,
   res1=CartesianProduct[cFC[[j]],cFC[[i]]];
   res2=Intersection[res1,edges];
   If[res2!={},edgesAux=Join[edgesAux,{{{j,i}}}]]]];
gAux=Graph[edgesAux,posVertCFC,EdgeDirection->True];
autSet=WeaklyConnectedComponents[gAux];
nConComp=Dimensions[autSet][[1]];
cCAux={};
For[i=1,i<=nConComp,i++,
  cCAux=AppendTo[cCAux,{autSet[[i]],gAux}]];
While[cCAux!={},
  vODeg0={};
  vDeg0={};
  gAux=cCAux[[1]][[2]];
  For[j=1,j<=nList,j++,
   If[OutDegree[gAux,listVertCFC[[j]]]==0,
    AppendTo[vODeg0,listVertCFC[[j]]];
    If[InDegree[gAux,listVertCFC[[j]]]==0,
     AppendTo[vDeg0,{listVertCFC[[j]]}]]]];
  vODeg0=Complement[vODeg0,Flatten[vDeg0]];
  nODeg0=Dimensions[vODeg0][[1]];
  For[k=1,k<=nODeg0,k++,
   vec=Complement[
     Neighborhood[MakeUndirected[gAux],vODeg0[[k]],
      1],{vODeg0[[k]]}];
   n=Dimensions[vec][[1]];
   gAux2=gAux;
   For[h=1,h<=n,h++,
    gAux2=DeleteEdges[gAux2,{{vec[[h]],vODeg0[[k]]}}]];
   c=Complement[WeaklyConnectedComponents[gAux2],
     autSet,{{vODeg0[[k]]}},vDeg0];
   autSet=Join[autSet,c];
   nc=Dimensions[c][[1]];
   For[r=1,r<=nc,r++,
    AppendTo[cCAux,{c[[r]],gAux2}]];];
  cCAux=Complement[cCAux,{cCAux[[1]]}]];
nAutSet=Dimensions[autSet][[1]];
For[i=1,i<=nAutSet,i++,
  For[j=i+1,j<=nAutSet,j++,
   int=Intersection[autSet[[i]],autSet[[j]]];
   If[int!=autSet[[i]],If[int!=autSet[[j]],
     aux=Union[autSet[[i]],autSet[[j]]];
     If[MemberQ[autSet,aux]==False,
      AppendTo[autSet,aux]]]]]];
autSet=autSet/.Table[i->Flatten[cFC[[i]]],{i,1,nList}];
numAutSet=Dimensions[autSet][[1]];
For[i=1,i<=numAutSet,i++,
 autSet=ReplacePart[autSet,Flatten[autSet[[i]]],i]]
(*autSet contains all the autonomous sets*)
```

Let us propose the last algorithm (ALG6) to determine the autonomous sets. This one uses the existence of at least one permutation matrix (permutations of rows and columns of the identity matrix) π such that $U = \pi^{-1} \cdot A \cdot \pi$ is a block upper-triangular matrix, where A is the technical coefficients matrix of $\mathscr{E}$. So, as we are about to see, the autonomous sets (and even the fundamental products) are obtained by considering the sectors associated with the nonzero blocks in these block upper-triangular matrices. In [4], the authors proposed a similar method but searching for matrices of the type: $\left(\begin{array}{c|c} U_{11} & U_{12} \\ \hline \Theta & U_{22} \end{array}\right)$. Each matrix of this kind provides at least one autonomous set. The main problem of that algorithm was the number of permutation matrices which had to be checked ($n!$, whatever the case). This new proposal avoids the exhaustive searching, since one matrix found can solve the problem completely.

Let us suppose that U (the matrix obtained in the first part of this algorithm) is block upper-triangular, with the maximal number (t) of square blocks in the main diagonal:

$$U = \left(\begin{array}{c|c|c|c} U_{11} & U_{12} & \cdots & U_{1t} \\ \hline \Theta & U_{22} & \cdots & U_{2t} \\ \hline \vdots & \Theta & \ddots & \vdots \\ \hline \Theta & \cdots & \Theta & U_{tt} \end{array}\right)$$

Note that U_{ij} can be non-square if $i \neq j$, and note also that the new indexes (in U, with respect to the ones in A) are affected by the "variable change" given by π. Bearing this in mind, the algorithm ALG6 finishes as follows:

–If $t = 1$, then the only autonomous set is N.
–If $U_{ij} = \Theta, \forall i < j$, then the indexes corresponding to U_{jj} constitute an autonomous set.
–For each $i = 1, \ldots, t$, the indexes corresponding to the first i^{th} blocks constitute an autonomous set.
–Finally, each union of two of the determined autonomous sets is also an autonomous set (according to Corollary 3.5).

Therefore, this algorithm ALG6 can be reduced to the achievement of the permutation matrices from the technical coefficients matrix A. In order to find such a matrix (or the corresponding matrix U), some steps can be followed, obtaining the permutation matrix as the product of a sequence of matrices. We schematize the process depending on the number (j, with $0 \leq j \leq n$) of nonzero elements in each specific column:

–Fist, for $j = 1$, we are looking for a column in A (the i^{th}) with at most one nonzero element: a_{ii}. These columns (if they exist) have to be placed in the first positions (with the convenient permutation matrix). Each column of this kind correspond to an autonomous set (constituted by one element, each); their combinations may give more autonomous sets (according to Corollary 3.5).
–For any j, we proceed as follows. The remaining rows and columns constitute a submatrix. From such a submatrix, we are looking for j columns $(i_1, \ldots, i_j)$ with $n - j$ zeros and, at most, nonzero elements in the rows $i_1, \ldots, i_j$. In this case, we use the appropriate permutation matrix to place these j columns in the first positions of the submatrix. Each of these j-uples, provides us an autonomous sets; their combinations are also autonomous (Corollary 3.5). In the worst case possible, j could be n, obtaining only one autonomous set (N).

Although this last algorithm (ALG6) is more efficient than the one suggested in [4], nowadays it still does not allow an implementation that competes with the previous algorithms using Graph Theory (like ALG5), since the number of permutation matrices can be high in the worst case possible.

To conclude this subsection, we want to indicate that all these algorithms (in any of their implementations) are obviously useful to determine whether the economy is decomposable. Moreover, according to Proposition 3.6, only the case with n fundamental products (and, hence, 1 autonomous set) is not decomposable.

5 Example: Greek input-output matrices

In this section we apply the presented algorithms to analyze a real economy. Specifically, we deal with the input-output matrices of Greece from 1998 (because this technique was partially proposed for the first time in the International Conference of Computational Methods in Sciences and Engineering hold in Greece in 2006 [3], although these algorithms had neither been implemented nor applied to real-world examples). From an economic viewpoint, this example is particularly relevant since it reflects the situation of an economy through a crisis. The technical coefficients matrices were computed from the symmetric input-output tables published in the Eurostat European database [2].

The main difficulty in carrying out the construction is that, for all the technical coefficients matrices, almost every $a_{ij} \neq 0$, in spite that some of the coefficients are too close to zero to be considered signs of *irreplaceable goods*. So, firstly, for 1998 and 57 sectors, we will make the following assumption: all the coefficients smaller than 0.01% of the maximum technical coefficient (in this case, < 0.00006639) will be treated as zero. Obviously, that condition can be weakened and the study can be repeated considering other *frontier percentages*. In fact, a challenging task would be the analysis of the different simplified matrices (later, we will display the computations for only three different frontiers) as well as the determination of the most characteristic threshold. Note that this frontier can also be considered on the intersectorial flow table, slightly changing the final results.

We give the simplified matrix for the Greek economy of 1998 in Table 1; the figures may be too small to study them, but they allow the understanding the size of the problem to be faced (the productive sectors are labelled according to Table 2; there are two non-productive sectors that were removed for our analysis: *uranium and thorium ores* and *private households with employed persons*). Moreover, the drawing of the corresponding digraph is usually a good idea (see [4] to consult some examples), but this present case involves to many sectors to allow a reasonable picture.

Removing two non-productive sectors and from the corresponding algorithms, we obtain the only two non-fundamental products: *tobacco products (sector 9)* and *public administration and defense services; compulsory social security services (sector 51)*. We also get the four existing autonomous sets: the proper set $N = \{1, 2, \ldots, 57\}$, $N \setminus \{9\}$, $N \setminus \{51\}$, and $N \setminus \{9, 51\}$.

Table 1: Simplified Greek technical coefficients matrix for 1998, rounded to the ten-thousandths place (for sector description, seeTable 2).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57
1	.1659	.0521	.0051	0	0	0	0	.3763	.6316	.1682	.0148	.0001	.0005	.0019	.0001	0	.0029	.0018	0	.0023	.0003	.0011	0	0	0	0	0	0	.0027	.0166	0	0	0	0	0	.0049	.0264	0	0	0	.0003	0	0	0	0	0	0	0	0	0	.0024	0	.0022	0	0	0	.0064
2	0	.0001	0	0	0	0	0	0	0	0	0	.0001	.0421	0	.0001	0	.0007	0	0	.0005	0	0	0	0	0	0	0	0	.0004	.0002	0	0	0	0	0	.0001	.0003	0	0	0	.0004	0	0	0	0	0	0	0	0	.0001	.0034	0	0	0	0	0	0
3	0	0	.0028	0	0	0	0	.0006	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0126	0	0	0	0	0	0	0	0	0	0	0	0	0	.0004	0	.0001	0	0	0	0
4	0	0	0	0	0	0	0	0	0	.0002	0	0	.0002	.0002	.0003	.0004	.0022	.0002	.0246	.0033	.0005	0	0	.0001	0	0	0	.0003	.0001	.0006	.1021	0	0	0	0	.0001	.0004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.5727	.0001	0	0	0	0	0	0	0	0	0	0	0	0	.0007	.0129	0	0	0	0	.0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	.0039	0	0	0	0	0	0	0	0	0	0	0	.0001	.0014	.0133	.0009	0	0	0	0	0	0	0	0	.0004	.0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	.0003	.0001	0	0	0	0	.0086	.0006	0	0	0	0	0	.0002	0	.0017	.0044	.0005	.0586	.0050	.0008	.0001	0	0	0	0	0	0	0	.0009	0	0	.0098	0	0	.0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	.0288	.0092	.0138	.0076	0	.0013	.0011	.1157	.0003	.0001	0	.0199	0	.0018	.0021	.0002	.0024	.0002	0	0	0	.0003	0	0	0	0	0	0	.0003	.0043	0	0	.0020	0	0	.0014	.1404	0	.0217	0	.0011	0	0	0	0	0	.0017	0	0	.0016	.0083	.0002	.0091	0	.0004	.0009	.0016
9	0	0	0	0	0	0	0	0	.0066	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	.0006	.0003	.0528	.0052	0	.0010	.0008	0	.0002	.0972	.1600	.0087	.0002	.0040	.0031	0	.0005	.0050	.0001	.0013	.0015	.0002	0	.0001	0	0	0	.0001	.0136	.0056	0	.0001	.0001	.0001	.0026	.0044	.0017	0	.0002	0	.0006	0	0	0	0	0	0	0	0	.0001	.0045	0	.0029	0	0	0	0
11	0	0	0	.0001	0	.0001	.0001	.0002	.0001	.0005	.0474	0	.0009	.0004	0	.0003	.0002	.0002	.0005	.0002	.0009	.0004	0	.0001	0	.0001	0	.0006	.0004	.0004	.0002	.0001	0	.0004	.0006	.0004	.0001	0	.0001	0	.0002	0	.0016	0	.0001	0	0	0	0	0	.0013	0	.0001	.0014	0	.0032	.0001
12	.0001	0	0	0	0	0	0	0	0	.0002	.0011	.1502	.0001	0	0	.0001	0	.0001	.0001	0	.0001	0	0	0	0	0	0	.0001	.0004	.0001	0	0	0	.0001	0	.0001	0	0	0	0	.0002	0	0	0	0	0	0	0	0	0	.0012	0	.0001	0	0	.0005	0
13	0	0	.0060	0	0	.0025	.0020	.0014	0	.0001	0	.0005	.1229	.0001	0	0	.0001	.0005	.0028	.0008	.0020	.0004	0	.0036	.0005	.0005	.0002	.0018	.0684	.0009	0	0	.0383	0	0	.0003	0	0	0	0	0	0	0	.0075	0	0	0	0	0	0	0	0	0	0	0	0	.0121
14	.0001	0	.0041	0	0	.0001	.0005	.0092	.0079	.0023	.0046	.0020	.0134	.1339	.1684	.0002	.0077	.0039	.0049	.0004	.0012	.0004	0	.0018	.0001	.0005	0	0	.0034	.0958	0	.0002	.0004	.0004	.0163	.0133	.0081	.0002	.0001	.0009	.0024	0	0	.0281	.0047	.0023	.0011	.0068	.0009	.0019	.0017	.0013	.0002	0	.0063	.0026	.0041
15	0	0	0	0	0	.0001	0	.0069	.0009	.0050	.0100	.0021	.0012	.0015	.0067	.0001	.0066	.0044	.0018	.0004	.0012	.0005	0	.0004	.0011	.0006	.0001	.0003	.0009	.0044	0	.0007	0	.0008	.0032	.0015	.0003	.0004	0	.0012	.0041	.0014	.0056	.0079	.00047	.0006	.0167	.0057	.1299	.0017	.0005	.0007	.0014	0	.0093	.0033	.0014
16	.0147	.0051	.0375	.0104	0	.0150	.0416	.0046	.0002	.0040	.0050	.0014	.0073	.0084	.0061	.0199	.0053	.0035	.0227	.0103	.0042	.0024	0	.0029	.0007	.0006	.0005	.0011	.0028	.0048	.0458	.0130	.0082	.0041	.0065	.0053	.0042	.0666	.0803	.0234	.0100	.0068	.0025	0	.0253	0	.0045	0	.0018	.0030	.0052	.0023	.0042	.0252	.0015	.0013	.0072
17	.0406	.0131	.0019	.0057	0	.0104	.0108	.0078	.0053	.0262	.0135	.0480	.0194	.0143	.0444	.0047	.1121	.1892	.0068	.0040	.0105	.0028	0	.0189	.0006	.0089	.0005	.0040	.0109	.0312	.0002	.0073	.0104	.0006	.0578	.0059	.0027	0	.0078	0	.0011	.0004	.0012	.0045	.0001	.0023	0	0	.0039	.0025	.0035	.0040	.0682	.0177	.0003	.0069	.0103
18	0	0	0	.0031	0	.0047	.0143	.0010	.0006	.0111	.0070	.0312	.0222	.0153	.0145	.0010	.0036	.0279	.0048	.0010	.0083	.0015	.0031	.0159	.0026	.0085	.0012	.0022	.0126	.0098	.0001	.0041	.0192	.0056	.0135	.0099	.0003	.0276	0	0	.0017	0	0	0	0	0	.0032	0	0	.0004	.0004	0	0	.0027	.0004	0	0
19	0	0	0	.0005	0	.0008	.0007	.0001	0	0	.0001	.0004	.0002	.0001	0	.0003	.0006	.0007	.0875	.0066	.0028	.0005	0	.0025	0	.0030	.0008	.0001	.0004	.0012	0	.0006	.0863	0	0	.0002	.0013	0	0	0	0	0	0	0	0	.0015	0	0	0	.0005	0	0	.0013	0	0	.0002	0
20	.0003	.0003	0	.0038	0	.0065	.0055	.0020	0	.0024	0	.0002	.0012	.0014	.0054	.0008	.0007	.0071	.0039	.1791	.1748	.0192	0	.1103	.0030	.0077	.0050	.0125	.0279	.0164	.0002	.0016	.0606	.0019	0	.0012	0	0	0	0	.0013	0	0	0	0	.0001	0	0	0	.0002	.0103	0	0	0	0	0	0
21	.0001	.0394	0	.0174	0	.0066	.0040	.0132	.0024	.0012	.0010	.0021	.0111	.0067	.0121	.0002	.0070	.0048	.0020	.0646	.0795	.0036	.0001	.0077	.0007	.0010	.0002	.0054	.0059	.0123	.0005	.0118	.0439	0	0	.0008	0	0	0	0	.0003	0	0	0	.0025	.0029	0	0	0	0	.0022	0	.0002	0	0	0	.0020
22	.0030	.0029	0	.0188	0	.0201	.0293	.0003	0	.0016	.0014	.0014	.0025	.0018	.0027	.0006	.0005	.0019	.0032	.0067	.0061	.0276	.0001	.0021	.0001	.0004	.0023	.0029	.0016	.0014	.0085	.0195	.0241	.0035	0	.0005	.0006	0	.0005	0	.0120	0	0	0	0	.0007	.0001	0	0	.0021	.0940	.0003	.0022	.0025	.0016	0	.0026
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0002	0	.0003	0	0	0	0	0	0	0	0	0	0	0	0	0	.0001	0	.0004	0	0	0	0	0	0	0	0	0	.0001	0	0	0	0	0	0
24	0	0	0	.0303	0	.0055	.0159	0	0	.0006	.0019	.0001	.0019	.0006	0	.0016	.0009	.0022	.0045	.0012	.0092	.0293	.0057	.0594	.0687	.0015	.0004	.0087	.0008	.0024	.0061	0	.0158	0	0	.0013	.0003	.0100	0	0	0	0	0	0	0	0	0	.0147	0	0	.0005	0	.0001	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0001	.0001	.0005	.0009	.0423	.0012	0	0	0	.0001	0	0	.0046	0	0	.0080	0	0	0	0	0	.0060	0	0	0	0	0	0	0	.0050	.0004	0	0	0	0	0	0
26	0	.00021	0	0	0	0	0	0	0	0	0	0	0	0	.0003	0	0	.0002	.0001	0	0	0	0	.0008	.0010	.0351	0	0	0	.0002	0	.0008	.0001	0	0	0	0	0	0	0	.0012	.0015	0	0	0	.0003	0	0	.0202	.0011	.0097	.0017	.0175	0	0	.0001	0
27	0	.0003	0	.0049	0	.0049	.0109	0	0	0	0	0	.0004	.0003	.0001	0	0	0	.0008	0	.0091	.0016	0	0	0	0	.0321	0	0	.0003	.0001	.0009	0	.1043	0	.0003	0	.0215	0	0	.0004	.0011	0	0	0	0	0	0	0	.0001	.0028	0	0	.0139	0	0	0
28	0	.0003	.0212	0	0	0	0	0	0	0	0	0	0	0	.0004	0	0	0	0	0	.0005	0	0	0	.0003	0	0	.0275	0	.0001	0	0	.0002	.0168	0	0	0	.0065	.0228	.0293	.0036	0	0	0	0	.0001	0	0	0	.0004	.0155	0	.0016	0	0	0	0
29	0	.0001	0	0	0	0	0	.0007	.0001	.0002	.0037	.0001	.0012	.0007	.0018	.0004	.0002	.0010	.0007	.0001	.0015	.0003	0	.0003	.0015	.0018	.0002	.0009	.0345	.0012	.0001	.0003	.0013	.0002	.0004	.0004	.0009	.0001	0	0	.0005	0	.0023	.0010	.0002	0	.0272	.0024	.0127	.0085	.0014	.0029	.0019	.0023	.0045	.0041	.0020
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0070	.0001	0	.0003	.0253	.0232	.0021	.0219	0	.0001	0	0	0	0	.0001	.0054	0	0	0	0	0	.0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	.0076	.0026	.0004	.0785	0	.0194	.0320	.0063	.0010	.0114	.0204	.0061	.0132	.0159	.0131	.0064	.0115	.0233	.0442	.0559	.0138	.0021	.0001	.0073	.0012	.0039	.0009	.0045	.0095	.0112	.0779	.0565	.0022	.0056	.0064	.0051	.0168	.0017	.0020	.0018	.0172	.0261	.0029	.0058	.0065	.0012	.0034	.0066	.0138	.0023	.0107	.0056	.0079	.0064	.0010	.0097	.0199
32	.0045	.0016	0	0	0	0	.0001	.0004	0	.0008	.0020	.0002	.0002	.0003	.0019	.0008	.0004	.0005	.0006	.0004	.0001	.0002	0	.0002	0	.0001	0	.0004	.0005	.0011	0	.0009	.0001	.0004	.0007	.0006	.0036	.0004	.0006	.0004	.0074	.0006	.0032	.0003	.0003	0	0	.0005	.0009	.0003	.0015	.0015	.0008	.0001	.0003	.0015	.0042
33	.0030	.0044	0	.0056	0	.0023	.0025	.0001	0	.0003	0	0	.0013	.0002	.0006	.0004	.0001	.0011	.0006	.0001	.0004	0	0	.0001	.0002	0	.0001	.0002	.0005	.0001	.0012	.0379	.0046	.0002	.0069	.0051	.0044	0	.0009	.0028	.0092	.0064	.0044	.0089	.0067	.0565	.0006	0	.0116	.0023	.0188	.0043	.0054	.0102	.0059	.0044	.0021
34	.0032	.0013	.0084	.0055	0	.0070	.0164	.0024	.0001	.0034	.0044	.0020	.0046	.0037	.0031	.0053	.0022	.0074	.0110	.0020	.0041	.0012	.0011	.0016	.0005	.0011	.0014	.0008	.0042	.0025	.0100	.0089	.0034	.0693	.0262	.0106	.0011	.0808	.0227	.0052	.0032	.0161	.0005	0	.0058	0	.1533	.0025	.0004	.0040	.0034	.0012	.0013	.0381	.0009	.0005	.0026
35	.0088	.0059	.0066	.0066	0	.0057	.0109	.0194	.0060	.0100	.0105	.0160	.0165	.0195	.0308	.0031	.0076	.0108	.0164	.0133	.0138	.0053	.0003	.0099	.0040	.0034	.0006	.0021	.0094	.0199	.0136	.0045	.0192	.0013	.0069	.0043	.0164	.0091	.0096	.0028	.0022	.0011	.0016	.0055	.0037	.0007	.00483	.0011	.0056	.0024	.0043	.0012	.0057	.0049	.0017	.0023	.0043
36	.0163	.0110	.0122	.0122	0	.0107	.0202	.0361	.0112	.0186	.0196	.0297	.0307	.0362	.0572	.0058	.0140	.0200	.0305	.0248	.0255	.0098	.0005	.0184	.0074	.0062	.0011	.0038	.0175	.0369	.0252	.0084	.0356	.0023	.0134	.0085	.0319	.0170	.0178	.0053	.0040	.0021	.0029	.0102	.0068	.00130	.0090	.0021	.0105	.0044	.0081	.0021	.0106	.0092	.0032	.0042	.0080
37	0	.0002	0	0	0	.0004	.0001	0	0	0	0	0	.0008	.0003	.0003	.0006	.0001	0	0	0	.0013	0	0	0	0	0	0	0	0	0	.0001	.0017	.0010	0	.0006	.0004	0	0	0	.0637	.0008	.0264	.0026	.0011	.0028	.0001	.0004	0	.0003	.0082	.0058	.0007	.0016	0	.0167	.0244	.0002
38	.0021	.0010	.0056	.0180	0	.0046	.0048	0	0	0	0	0	.0001	0	.0002	0	0	.0001	.0001	0	0	0	0	0	0	0	0	0	0	0	.0009	0	.0003	.0028	.0575	.0407	.0025	.0017	0	.0011	.0010	.0037	.0037	0	.0284	0	.0009	0	0	.0021	.0068	.0022	.0009	0	.0011	.0053	.0309
39	0	0	.0083	0	0	0	0	.0006	.0003	.0008	.0031	.0006	.0008	.0010	.0023	.0001	.0010	.0015	.0011	.0004	.0012	.0005	0	.0010	.0007	.0004	.0001	.0011	.0014	.0015	0	0	.0001	.0001	.0056	.0033	.0025	0	.0014	0	.0001	.0028	0	0	0	0	0	0	0	0	.0010	.0001	.0001	0	0	0	0
40	0	0	0	.0024	0	.0019	.0011	.0005	.0004	.0003	.0015	.0002	.0009	.0004	.0013	.0005	.0006	.0011	.0007	.0003	.0007	.0002	0	.0004	.0007	.0001	0	.0004	.0005	.0008	.0003	.0001	.0007	0	.0051	.0031	.0002	.0003	0	.0007	.0005	.0109	.0013	.0039	.0001	0	.0008	.0009	0	.0007	.0022	.0020	.0002	0	0	.0079	0
41	0	0	.0301	0	0	.0003	.0002	.0015	.0002	.0005	.0004	0	.0002	.0004	.0002	0	.0003	0	0	.0002	0	0	0	0	0	0	0	0	.0001	.0002	0	0	.0005	0	.0152	.0114	.0024	0	0	.2428	0	.0003	.0004	0	0	0	.0025	0	0	0	.0003	.0002	0	0	0	0	0
42	0	.0005	.0033	.0007	0	.0036	.0031	.0020	.0003	.0024	.0093	.0019	.0039	.0026	.0087	.0006	.0014	.0049	.0043	.0010	.0042	.0011	.0001	.0021	.0015	.0030	.0003	.0014	.0050	.0051	.0016	.0071	.0044	.0029	.0485	.0375	.0108	.0077	.0002	.0044	.0359	.1060	.0127	.0235	.0777	.0003	.0234	.0107	.0011	.0144	.0183	.0039	.0050	0	.0067	.0120	.0136
43	.0008	.0025	.0138	.0224	0	.0039	.0033	.0072	.0031	.0172	.0139	.0068	.0162	.0111	.0123	.0072	.0052	.0177	.0265	.0133	.0195	.0052	.0002	.0201	.0086	.0010	.0009	.0060	.0108	.0098	.0001	.0461	.0012	.0002	.0332	.0165	.0001	.0005	.0006	0	.0019	.0043	.6639	.0136	.0564	.0004	.0004	0	0	.0164	.0078	.0020	.0024	0	0	.0021	0
44	.0004	.0001	.0023	.0001	0	.0014	.0015	.0012	.0009	.0018	.0029	.0008	.0025	.0021	.0020	.0012	.0011	.0019	.0010	.0007	.0012	.0003	0	.0012	.0004	.0004	.0002	.0012	.0016	.0014	0	0	.0001	.0008	.0017	.0014	.0004	.0051	.0079	.0035	.0052	0	.0028	.0337	.0002	0	.0047	0	0	.0003	0	0	.0001	0	.0004	.0001	0
45	0	0	0	0	0	0	0	.0051	.0011	.0015	.0036	.0003	.0006	.0006	.0008	.0019	.0004	.0024	.0016	.0032	.0008	.0002	0	.0005	.0002	0	0	0	.0004	.0008	0	0	.0049	.0016	.0029	.0022	.0025	.0005	.0005	0	0	.0001	.0139	.2753	.0788	.0008	0	0	0	.0015	0	0	0	0	0	0	0
46	0	.0011	.0007	.0047	0	.0317	.0232	.0024	.0009	.0013	.0033	.0011	.0011	.0018	.0029	.0011	.0018	.0021	.0018	.0010	.0012	.0003	0	.0007	.0006	.0003	.0001	.0004	.0011	.0014	.0029	.0332	.0040	.0566	.0008	.1217	.0393	.0131	.0005	.0061	.0538	.0144	.0099	.0247	.0008	.0019	.0556	.0141	.0199	.0480	.0279	.0308	.0211	.0147	.0277	.0500	.1122
47	0	0	0	.0049	.0001	.0059	.0252	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0001	.0002	.0306	.0025	0	0	0	.0002	.0069	0	.0358	.0389	.0127	.0108	.0018	.0009	.0001	.0010	0	.0098	.0015	.0007	.0009	.0003	.0106	.0010	.0039	0
48	0	0	0	.0055	0	.0097	.0081	0	0	0	0	0	0	0	.0001	0	0	.0002	0	0	.0001	0	0	.0002	0	0	0	0	.0001	0	.0002	.0001	0	0	0	0	0	0	0	.0040	.0003	.0012	.0045	.0048	.0004	0	0	0	0	.0001	.0022	.0014	.0056	0	0	0	0
49	0	.0014	.0010	0	0	.0061	.0097	0	0	0	0	0	0	0	.0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0002	0	0	0	0	0	0	0	0	0	.0005	0	0	0	0	0	0	0	.0037	.0001	.0040	.0005	0	0	0	0	0
50	.0004	.0005	.0018	.0008	0	.0432	.0143	.0262	.0181	.0225	.0676	.0166	.0125	.0286	.0384	.0053	.0308	.0311	.0162	.0139	.0137	.0065	.0003	.0101	.0080	.0052	.0009	.0044	.0220	.0258	.0016	.0005	.0145	.0028	.0381	.0253	.0096	.0030	.0005	.0035	.0291	.0035	.0496	.0760	.0042	.0119	.0165	.0107	.0018	.0262	.0150	.0074	.0051	.0011	.0121	.0045	.0032
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	.0008	.0007	.0001	.0001	.0002	.0014	.0001	.0001	.0003	.0001	.0001	.0003	.0003	.0003	0	.0002	0	0	.0002	.0011	0	.0001	.0005	.0004	.0001	0	0	0	0	0	.0001	0	0	0	.0005	.0003	0	.0040	0	.0006	0	0	0	.0001	.0045	.0009	.0013	0	0	0	0	0
53	.0004	.0005	0	0	0	0	0	0	0	0	0	0	0	0	.0005	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0001	0	.0035	0	.0026	0	.0002	.0001	0	0	0	.0005	.0190	.0002	.0032	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	.0005	.0002	0	.0003	.0003	0	.0002	0	.0002	0	0	0	0	0	0	0	0	0	.0001	.0008	0	0	.0099	.0071	0	.0357	.0034	0	.0206	0	.0051	0	.0004	.0001	0	.0032	.0025	.0016	.0002	.0030	.0029	0	.0147	0	0
55	0	0	0	0	0	.0028	.0009	0	0	0	0	0	.0008	.0002	0	.0001	.0002	.0005	.0005	0	.0003	0	0	0	0	0	0	.0003	0	0	.0002	.0022	.0001	0	.0053	.0036	.0022	0	0	.0001	.0008	.0026	0	0	0	0	.0006	0	.0023	.0016	0	0	.0008	0	.0145	.0006	.0004
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0100	0	0	0	.0001	0	0	0	0	0	0	0	0	0	0	.0050	.0006	0	0	0	0	0	.0066	0	0	0	.0002	0	0	0	0	.0001	0	0	0	.0386	.0018	.0014	0	0	0	.1985	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.0110	.0006	0	0	.0001	.0010	0	0	0	0	0	0	0	0	.0005	0	.0006	0	0	0	0

Table 2: Official sector description, according to Eurostat [2].

SECTORS / GOODS	
1. Products of agriculture, hunting and related services	29. Furniture; other manufactured goods n.e.c.
2. Products of forestry, logging and related services	30. Secondary raw materials
3. Fish and other fishing products; services incidental of fishing	31. Electrical energy, gas, steam and hot water
4. Coal and lignite; peat	32. Collected and purified water, distribution services of water
5. Crude petroleum and natural gas; services incidental to extraction	33. Construction work
6. Metal ores vehicles; retail sale of automotive fuel	34. Trade and repair services of motor
7. Other mining and quarrying products	35. Wholesale trade and commission trade services (no motor vehicles)
8. Food products and beverages	36. Retail trade services (no motor vehicles); repair services (household...)
9. Tobacco products	37. Hotel and restaurant services
10. Textiles	38. Land transport; transport via pipeline services
11. Wearing apparel; furs	39. Water transport services
12. Leather and leather products	40. Air transport services
13. Wood and products of wood and cork (no furniture); straw and plaiting...	41. Supporting and auxiliary transport services; travel agency services
14. Pulp, paper and paper products	42. Post and telecommunication services
15. Printed matter and recorded media	43. Financial intermediation services (no insurance and pension...)
16. Coke, refined petroleum products and nuclear fuels	44. Insurance and pension funding services (no compulsory services)
17. Chemicals, chemical products and man-made fibres	45. Services auxiliary to financial intermediation
18. Rubber and plastic products	46. Real estate services
19. Other non-metallic mineral products	47. Renting services of machinery and equipment without operator...
20. Basic metals	48. Computer and related services
21. Fabricated metal products, except machinery and equipment	49. Research and development services
22. Machinery and equipment n.e.c.	50. Other business services
23. Office machinery and computers	51. Public administration and defence...; compulsory social security services
24. Electrical machinery and apparatus n.e.c.	52. Education services
25. Radio, television and communication equipment and apparatus	53. Health and social work services
26. Medical, precision and optical instruments, watches and clocks	54. Sewage and refuse disposal services, sanitation and similar services
27. Motor vehicles, trailers and semi-trailers	55. Membership organisation services n.e.c.
28. Other transport equipment	56. Recreational, cultural and sporting services
	57. Other services

Table 3: Computing time when applying different algorithms to simplified Greek technical coefficients matrices for 1998, 2000, and 2005, depending on the threshold considered.

Year & threshold	ALG1	ALG2’	ALG3’	ALG5
Greece 1998 (0.01%)	17.253 sec.	0.998 sec.	1.698 sec.	2.57 sec.
Greece 1998 (0.05%)	11.449 sec.	1.061 sec.	>1 hour	2.013 sec.
Greece 1998 (0.1%)	11.981 sec.	1.202 sec.	>1 hour	1.965 sec.
Greece 2000 (0.01%)	23.821 sec.	1.17 sec.	<0.001 sec.	2.621 sec.
Greece 2000 (0.05%)	21.903 sec.	1.185 sec.	2.075 sec.	2.184 sec.
Greece 2000 (0.1%)	18.783 sec.	1.232 sec.	>1 hour	2.433 sec.
Greece 2005 (0.01%)	28.174 sec.	1.248 sec.	1.155 sec.	2.325 sec.
Greece 2005 (0.05%)	19.452 sec.	1.388 sec.	1.498 sec.	2.075 sec.
Greece 2005 (0.1%)	17.831 sec.	1.186 sec.	1.248 sec.	2.385 sec.

When repeating the computations for years 2000 and 2005 (the only two with an official data set after 1998), the results are very similar. In 2005 the fundamental products and autonomous sets remain the same, while in 2000 we gain the fundamental product from sector 51 and we lose two autonomous sets: only $N = \{1,2,\dots,57\}$ and $N \setminus \{9\}$ are left.

We can also change the threshold and consider that all the coefficients smaller than 0.1% of the maximum technical coefficient (for instance, for 2000) are zero, then there is no fundamental product in the economy. Finally, when considering a threshold of 0.05% for 2005, the results are the same than in 2000 with 0.01%: all the products are fundamental except for number 9, and there are only two autonomous sets. The interested researcher can economically interpret these results; however, this task exceeds the scope of this paper.

This example can be useful to make a simple comparison between the different algorithms described along the paper. After applying the best algorithms to the data set, the differences between them became evident in terms of computing time (see Table 3). The computer used for these calculations was an Intel$^{(R)}$ Core$^{(TM)}$ i3, 2.13GHz RAM 4Gb.

With respect to the complexity order, we can indicate some superficial characteristics. In general, they show a polynomial complexity, being more evident in the case of fundamental products. For ALG2’, the order is n^3, being robust since it can be applied to any binary matrix encoding the information about the transactional flows in an economy. In the case of ALG1, the complexity is conditioned by Dijkstra algorithm (whose complexity order is n^2), which involves a total complexity order n^3 for our implementation. In this sense, the theoretical complexity is equivalent for ALG1 and ALG2’. With respect to the robustness, ALG1 is also robust for binary matrices. Finally, in relation to ALG6 (regarding the autonomous sets), the algorithm has complexity order n^3 and it is robust (same reason than in the first case analyzed).

6 Conclusion

In this paper we explain some algorithms which allow to determine the fundamental products and the autonomous sets of a given economy. These algorithms provide a computational treatment of these concepts by using common packages as Mathematica. Their most valuable applications lie in both research on and teaching of Economics, since it is possible to compute and visualize the structure of real economies (see, for instance, [4]).

According to Table 3, we propose the use of ALG2' to compute the fundamental products of an economy, using graphs via their matrix representation. However, we think it is a better option to use ALG5 to obtain the list of autonomous sets, through the computation of the strongly connected components of the graph associated with the economy. Combining both routines (which can be run independently), an efficient algorithm can be proposed to compute the fundamental products as well as the autonomous sets.

Looking towards the future, the algorithm ALG6 shows promising characteristics, but the search of permutation matrices should be improved to guarantee a good performance in the worst cases possible. We propose the use of LU decomposition to find a way of simplifying the search of such matrices.

Here we can certainly provide other two ideas for future research. On the one hand, the analyst can use the more significant changes in the obtained results from year to year in order to detect mistakes or even forged information included in the data set. On the other hand, by changing the sensitivity for the input data set (something quite accessible from the implementation of our algorithms), it would be possible to estimate the dependence of an economy on its exports. Of course, these two tasks complement others, classically performed, like the comparison of different economies or the analysis of the evolution of a fixed economy.

References

[1] R. Bott, J.P. Mayberry, Matrices and trees, In: Morgenstern, O. (Ed.), Economic Activity Analysis. New York, Wiley, pp. 391–400, 1954.
[2] Eurostat, ESA 95 Supply Use and Input-Output tables (Symmetric Input-Output Tables), http://epp.eurostat.ec.europa.eu. Accessed 13 September 2012.
[3] E.M. Fedriani, A.F. Tenorio, Lect. Ser. Computer Computat. Sci. **7**, 145–148 (2006).
[4] E.M. Fedriani, A.F. Tenorio, Econ. Model. **29**, 1931–1937 (2012).
[5] F. Harary, Graph Theory, Massachusetts, Addison-Wesley, 1969.
[6] W.W. Leontief, Rev. Econ. Statistics **18**, 105–125 (1936).
[7] W.W. Leontief, Input-Output Economics, New York, Oxford University Press, 1966.
[8] L. de Mesnard, J. Regional Sci. **44**, 125–141 (2004).
[9] P. Michel, Cours de Mathématiques pour economistes, Paris, Economica, 1989.
[10] P. Sraffa, Production de marchandises par des marchandises. Prélude à une critique de la Théorie Economique, Paris, Dunod, (1970).

Eugenio M. Fedriani is an associate professor in the Department of Economics, Quantitative Methods, and Economic History at Pablo de Olavide University. He got his Ph.D. in Mathematics in 2001, and his main research interests are Topological Graph Theory, Poverty Measurement, Lie Theory, and Mathematical Education. He has published more than one hundred of scientific papers and books.

Ángel F. Tenorio is B.Sc., M.Sc. and Ph.D. in Mathematics from the University of Seville. At present, he is an Associate Professor of Applied Mathematics in the Department of Economics, Quantitative Methods and Economic History at Pablo de Olavide University, His contributions are related to Lie Theory, Computer Algebra, Graph Theory, Applications to Economics, History, Popularization and Didactics of Mathematics, with more than one hundred papers about these topics.