Development and Aplication of Mathematical Programs for Contribution Analysis in Life Cycle Assessment

Monica Vlad

Master of Science in Energy and Environment
Submission date: July 2009
Supervisor: Anders Hammer Strømman, EPT
Co-supervisor: Richard Wood, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

Problem Description

Background and objective:

In Life Cycle Assessment today the main method for contribution analysis is based on simple matrix operations. These operations can, for example, tell us about the contribution to total impacts and stressors by the various processes in a production network. However, they not reveal the paths describing the process relationships linking the final delivered product with the emitting processes. For this purpose more advanced methods are required. The use of Taylor series expansion allows insight into the contributions of the individual layers of the production system. However, structural path analysis (SPA) is required for a complete insight into the chain of processes linking the most emitting processes with the final product.

The potential utility of mathematical programming in life cycle contribution analysis remains to a large extent unexplored. The prevailing methods for advanced contribution analysis, like SPA, are formulated as specialized algorithms. In this project we aim to explore the possibility of applying mathematical programs for contribution analysis. The advantage is larger flexibility in terms of functionality. The downside could be larger computational times.

The project involves formulating mathematical programs to perform various types of inventory analysis. Further, to implement these in Gams and Matlab and solve the problem using generic solvers. If possible, the performance of the mathematical program and generic solver should be benchmarked and compared to an existing dedicated algorithm performing the same function.

The following elements should be included in the project work:

1. Overview of present practices for contribution analysis;
2. Development of mathematical programs for contribution analysis;
3. Testing of these programs on an appropriate LCA database;
4. Application to analysis of selected case studies;
5. Discussion.

Assignment given: 02. February 2009
Supervisor: Anders Hammer Strømman, EPT

Abstract

The environmental impact of a final product can be regarded as the sum of the impacts of all processes needed to obtain it. The impacts of these processes in all individual layers of production can be quantified using contribution analysis methods. SPA is an advanced method used to identify the chain of production processes linking the most highly emitting process with the final product. This analysis was performed in Matlab, using a specialized algorithm developed by Peters and Hertwich in 1996.

In this thesis we test an interdisciplinary approach combining LCA and operational research methods for doing a SPA. A mixed integer program was developed and implemented in Gams. The performance of this generalized algorithm was benchmarked against the specialized algorithm for three test cases performed on three databases of increasing complexity.

The results suggest the advantage of this algorithm in performing analysis on sparse data systems compared with the classic method involving Matlab. However, Matlab's specialized algorithm performs better for dense data systems.

Many of the requirements and limitations imposed by the software involved in different steps have proved manageable.

This study proves that mathematical programming can be a very useful tool for contribution analysis in general and SPA in particular.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

1.1 Motivation

In our days there is an increased concern about the way we exhaust the resource endowments of the planet and destroy the ecosystem by pollution. Sustainability is still a far-fetched ideal in most locales, and climate is changing at an accelerated rate. These effects are the results of the way we produce and consume products and services, use toxic substances and treat the increasing amount of waste.

The concern about the worldwide environmental problems appeared in the 1960's and grew constantly, along with the joint effort of the scientists for finding a solution to these problems. Some of the most representative instances of common effort in mitigating the climate change are the four assessment reports (1990, 1992, 1995, 1997) of the Intergovernmental Panel on Climate Change (IPPC), where human impact on the environment is quantified and presented, along with recommendations to be used by politicians in elaborating mitigation polices. The high level of interest and involvement in finding solutions for dealing with global warming was first demonstrated through two conferences of unprecedented global participation: the Second World Climate Conference in Geneva in 1990 and the United Nations Conference on Environmental Development (UNCED) in Rio de Janeiro in 1992. A third conference of similar magnitude – the UN Convention on Climate Change – will be held in December 2009 in Copenhagen. This event will bring together government delegates, representatives from business and industry, environmental organizations and research institutions to negotiate solutions for "... stabilization of greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic interference with the climate system." (http://unfccc.int/ghg_data/items/3800.php)

Reducing the level of greenhouse gas emissions is recognized as having a higher priority than the other problems enumerated in the first paragraph. This is because existing models predict the direst consequences if no measures are taken. The extent of these effects is global, and efforts to reduce emissions must be also coordinated globally to avoid shifting of responsibilities through outsourcing of production. This issue is treated by Dunchin (1994), who builds on her previous work to show that if existing trends continue for the next decades, total emissions of principal global pollutants will increase considerably. This will happen even if the most optimistic assumptions about pollution reduction and introduction of new technologies come to pass, and even if only moderate economic development objectives are achieved in the developing countries in the next few decades. The root of the problem is the continuation of the historic shift of the geographic source of most emissions from the developed to the developing economies.

The reduction of the environmental impact can only be achieved through a joint effort of the producers, consumers and governments. They can influence each other by demand, supply and taxes.

Consumers have lots of goods and services to choose from, each one with a different environmental impact. In order to help reduce environmental impact, the consumer can choose a less polluting product instead of a more polluting one and adopt a lifestyle based on saving resources (reusing and recycling).

Producers have an even bigger leverage in using technology to reduce the environmental impact. At the most basic level it is technology which determines the environmental impact of a certain product. It is the producer's main responsibility to fulfill the consumers' needs and to reduce the environmental impact of its products. The consumers' needs are usually very well defined (i.e. if a consumer needs a chair, he knows well what shape and size the chair must have) and the company that tries to fulfill it needs to integrate in the product or service all the features connected with the functionality of the product. So, the company does not have much of a choice in terms of required features and the type and size of inputs required from the suppliers, especially when there is only one product available which can fulfill a requirement. However, in other cases, producers have several choices (i.e. the chair can be made of wood, plastic, metal or a combination of these). The reason for choosing a material or another, for example, must depend on both economical and environmental considerations. Changing the inputs, the company changes the technology and the requirements towards its suppliers. The suppliers will produce the requested good with the smaller environmental impact from now on. If most of them follow the same practice, the entire network will be "greener" than before the change.

Because only a relatively small number of processes and inputs can be researched, producers need to know which have the largest environmental impact. They also need to estimate economic and technological trends related to these processes and inputs in order to adapt the production accordingly. For a company trying to fulfill the demand, this is a strategic decision.

Governments are the other agents involved in changing the technologies with the declared aim of diminishing environmental impacts. While customers need to be educated in order to make the right decision for them, governments can use their share of final demand in a premeditated fashion. This includes buying more environmentally-friendly products for their use or even using taxation, subsidies and standards to favor introduction of lower-impact technologies in detriment of polluting ones.

Producers, consumers or governments trying to make a change have difficulties identifying which actions have a higher leverage in reducing the impacts. This is true partially because of the relatively high number of actions and relatively limited resources available. In addition, any measure affecting one sector will indirectly affect other sectors through the production-consumption link. The indirect effects should not be ignored by the government imposing the measure.

A policy measure or a consumer initiative to reduce impact can have implications beyond the first order impact (different inputs to the product). It is important to take into consideration the eventual rebound or spillover effect characterizing so deeply the human behavior and to adjust the initiative accordingly.

Many of the above issues regarding companies, consumers and governments have been identified by Hertwich (2005) in his critical review about modalities in which LCA, IO and their hybrid forms can be efficiently used to identify consumption patterns leading to environmental impact.

1.2 State of the field

For the companies involved in product development and production it is important to have a method for identifying the implications of every alternative strategy. This method is *input-output analysis* because, as Dunchin (1994) mentions, the input-output model can be a very powerful tool for identifying both the economical and environmental implications of every alternative strategy.

Input output analysis is a methodology developed to describe the production network and the connections between production and final consumption. As I mentioned already, it has the ability to incorporate environmental externalities. This recommends it as a very useful tool to determine the environmental impacts of the various economic activities in any production network. But standard input output analysis can only analyze the aggregated impacts of consumption and production separately; it does not identify the processes with the higher environmental load, as part of the production sectors. This statement will be detailed in the methodology chapter.

Early attempts to determine the endogenous interaction process between the production and consumption agents as households and institutions used a *social accounting model* (SAM) framework. SAMs are extended input-output tables measured in monetary units, obtained through calibration, in order to reproduce the flows from the past (Strømman et al. 2009). This process limits their applicability to the years when data was recorded and are not very suitable for scenarios analysis in the future. Defourny and Thorbecke (1984) introduced *structural path analysis* (SPA) by using the SAM framework to explain how the influence is transmitted through structural paths in an economic system. They used multipliers to measure the influence of the endogenous and exogenous accounts in the general equilibrium data system. The multipliers correspond to the amplification of the economic activity in a network determined by fulfilling a certain final demand, calculated through the general decomposition technique derived from equation 1.12 (see next chapter). The paths were identified by making a unitary change in one of the elements of the path, all other parameters remaining unchanged. These authors measured only activities and did not take into consideration the applicability of their framework to model environmental impacts. The transition between $(I-A)^{-1}$ to $F(I-A)^{-1}$ in the *calculation of the multiplier* was exemplified by Lenzen (2001). He calculated labor and energy multipliers for labor and energy for all 109 sectors of the Australian economy, taking imports into consideration. Capital investment and imports are internalized in the inter-coefficient matrix along with representation in both monetary and physical units for a more precise and clear result.

SPA was refined by Peters and Hertwich (2006) by taking trade into consideration. They performed their calculations for a *multi-regional input-output analysis* (MRIO) of the Norwegian economy. MRIO models have the advantage of measuring economic flows not only in monetary units, as SAM, but also in physical units (Dunchin 2004, Weisz and Dunchin 2006). This allows including in the general models factors such as energy use (i.e. MJ), labor (i.e. hours of work) and natural resources (i.e. oil, copper). MRIO tables have two additional advantages. One advantage is their increased level of detail. The second advantage is the fact that they can separate emissions associated with the exports from domestic emissions determined by domestic demand. However, including imports in the database used for calculation introduces uncertainties associated with the assumptions about quantities and technologies used to produce the imported goods. Also, the MRIO framework does not allow evaluating simultaneous changes in the economic process, and therefore cannot be used to identify an optimal state of the economy.

Peters and Hertwich used three complementary approaches to study the economy: (1) the consumption perspective, identifying final demand purchases producing environmental impact; (2) the production perspective, identifying the production processes generating pollution for a given final demand, and (3) structural path analysis to identify the linkages between production and consumption.

The methodology for the first two approaches used the power series expansion. This was done in order to scale the environmental impacts of household, government and export (as consumption agents) on one hand, and aggregated sectors (as producing agents) on the other.

The methodology for structural path analysis proceeds by representing the economy as a tree which can be search for all individual linkages between consumption and production. This representation is based on both the input-output representation of the economy and on the complete set of endogenous supply-demand relations. Peters and Hertwich (2006) identified the most important paths by enumerating all possible paths and stopping when the value of the emissions associated with the path became lower than a given threshold. The specialized algorithm was implemented in Matlab. This implementation is fast taking into consideration the fact that it is searching the paths one by one. Chapter 2.3.1 will present the algorithm in detail.

From a very different perspective*, operational research* (OR) is a field of study designed specifically to identify the optimal state of a system when parameters are given. The IO framework is traditionally used with OR models since 1958, when Dorfman et al. showed how to use a linear program to determine international flows based on lowest cost

allocation of resources among competing uses. They used this model to exemplify the connections between primal and dual versions of the linear programs.
Another classical application of linear programming using the IO framework was to determine the low cost choice among alternative technological options (Dunchin and Lange 1995).

Linear programs have been successfully used by Dunchin in 2005 to build a *world trade model* (WTM) which minimizes resource use at given global consumption. The model is based on an input-output description of different regions and availability of regional resources evaluated at local prices. This WTM was refined one year later by Strømman and Dunchin by introducing transportation and developing the *world trade model of bilateral trade* (WTMBT). In 2009 Strømman et al. refined the model even further by introducing a second objective: minimization of CO_2 in physical units. He also extended the database with sectors comprising different parts of the aluminum chain.

The disadvantages of linear programming models consist in extreme or unique solutions, which can prove practically unrealistic. Also, for a high number of parameters, it is hard to say how reliable the obtained solution is. These draw-backs can be partially compensated by using sensitivity analysis to check the reliability of the results.

An advantage of OR models consists of using only a limited number of measurable parameters which can change as subjected to different scenarios (Strømann et al. 2009). This feature compensates for the main disadvantages of the MRIO models mentioned earlier.

However, the number of parameters used in linear programs increased with the size and complexity of the underlying trade models (Dorfman et al. in 1958: 2 regions and 2 products, Dunchin WTM in 2005: 10 regions, 8 goods, 3 factors of production, Strømman et al. 2009: 11 regions, 15 production sectors, 7 factors of production). If linear programming is applied to larger systems, for different types of analysis, availability of memory and larger computation times may become an issue.

The capabilities of the linear programming in addressing trade models, especial for, are very promising, but the potential utility of the mathematical programming to address other industrial ecology issues was to a large extent unexplored until now.

1.3 Objective of the research

SPA methodology developed up to this point is based on a specialized algorithm developed by Peters and Hertwich (2006). This algorithm uses enumeration to check every path, then stops when the emissions are below a certain threshold.

This thesis explores the possibility of doing a SPA using a generalized algorithm which starts with cutting paths and after that enumerates all possible combinations.

The new method takes an interdisciplinary approach: it combines linear programming methodology with the input-output methodology used for the specialized algorithm. It builds a mathematical model of the problem which is implemented into a high-level modeling system (The General Algebraic Modeling System – Gams). Gams software has the generalized algorithm necessary to solve the linear programming model. Because the SPA is intended to use Gams, from now on it will be called Gams-SPA. The specialized algorithm is programmed exclusively for Matlab and will be called Matlab-SPA.

Gams-SPA results are benchmarked against the Matlab-SPA results by comparing computation times. Three different data systems will be used for testing: (1) a 10-sector economy simplified case, (2) a 481-sector economy derived from the US input-output table and (3) an LCA system with approximately 2500 processes. Three final demands will be chosen to compare the results, one for each data type. First goal is to find out which algorithm is faster for a specific type of data. Second goal is to find out the limits of the algorithms, respectively the biggest data system they can be used for. The testing results on LCA data are very important because this is the biggest data system available now. The third goal is to determine the weaknesses of the linear programming model. One of these possible weaknesses is the long computation time, and another is the complicated data manipulation required to be performed by an algorithm which was not specifically designed for this type of analysis.

The linear programming model is designed for only one stressor. This will make the model easier to understand, without affecting the algorithm.

The aim of this research is to explore the utility of using linear programming to address industrial ecology issues; it is not intended to come with a final program for SPA, but to explore different formulations and approaches. The goal is an intermediate working version for SPA using operational research which can be used as a starting point for future development.

1.4 Thesis outline

The rest of this thesis is structured as follows:

Chapter 2 discusses methodology to the extent necessary to facilitate understanding, but not beyond the uses for our model. It gives an overview of the concepts and equations used by IO analysis to represent the relations between economic sectors. The contribution analysis section shows how to calculate the environmental impacts associated with a final demand. The chapter also includes the particular case of one stressor. It explains the limits of the IO methodology for doing a SPA and justifies the necessity of the tree representation to go over these limits. An example is then used to exemplify these findings. Based on this problem formulation we then explain the specialized algorithm developed by Peters and Hertwich (2006). Then we introduce mathematical programming as an alternative approach, discussing the general structure of a mixed integer programming model, its applicability to our problem and some of its advantages and disadvantages in general.

Chapter 3 presents three mixed integer formulations of increased optimization of Gams SPA problem. Implementations adjustments will be discussed in detail.

The reasons for keeping the actual version of the model will became obvious in chapter 4. Three case studies, of increasing data volume, will be used to test the model. The technical performance of the model will be assessed by benchmarking the results of Gams-SPA against the results of Matlab-SPA for each of the three cases mentioned above. We introduce some alternative formulations which have been tried and discarded in the process of developing the actual model. The features included in the actual model will be justified by comparing the results of this version with older versions.

Chapter 5 will discuss the achievement obtained with this program and the relevance of the results. In the end we will establish the coordinates from which to start future improvements in a SPA using a generic algorithm.

Chapter 2

Methodology

2.1 Input-output analysis

In trying to understand and attempt to solve the problems at global and local level, we need a systematic model. The model is used to study production and consumption patterns and the connection between them at any level.

The economy can be visualized as a very large production network. The basis of this paradigm is the fact that the process of production of a good or service needs inputs as materials and energy from other sectors of the economy, which in turn require inputs from other sectors and so on, until the network includes the entire economy. Most of the goods or services produced in a sector as output are at the same time sold as an input to other sectors. Some of these goods, however, are sold to the households or to the government. The amount produced from these goods depends on factors which are exogenous to the producers' requirements as inputs. For example, the request for small cars from households depends on the price of the gasoline; the demand for military aircrafts depends on the national policy and the budget level and so on. The demand for this type of products, external to the production network requirements, is referred to as the *final demand.*

Starting from this representation of the economy, W. Leontief developed in the 1930's a framework called input-output analysis – IO – (Leontief 1941) for which he received the Nobel Prize in economics in 1973.

The input-output model is constructed from observed data from a region, usually a country. The data are the flow of products from each sector (as purchasing sector) to each sector (as selling sector). The flows of goods from sector i to sector j is usually represented by z_{ij} and traditionally it is in monetary units. Some goods are required by purchasers exogenous to the industrial sectors (households, governments, net export) and tend to be used as such and not as inputs to the industrial sectors. They form the final demand and are denoted by $\mathbf{y}$. The total output of sector i (as input for other industries and for final demand) is represented as x_i. Labor, taxes, interest, profit and so on which are paid by the purchasing sector in addition to the inputs from other industries make the value added, $\mathbf{v}$. All these elements can be combined in a table, called the *input-output table* and represented by the shadowed part of the Table 2.1, below.

		Purchasing sectors				Final demand	Total output
		sector 1	sector2	…	sector n		
Selling sectors	sector 1	z_{11}	z_{12}	…	z_{1n}	$\mathbf{y_1}$	x_1
	sector 2	z_{21}	z_{22}	…	z_{2n}	$\mathbf{y_2}$	x_2
	…	…	…	…	…	**…**	…
	sector n	z_{n1}	z_{n2}	…	z_{nn}	$\mathbf{y_n}$	x_n
Value added		$\mathbf{v_1}$	$\mathbf{v_2}$	**…**	$\mathbf{v_n}$	GDP	
Total input		x_1	x_2	…	x_n		**X**

Table 2.1: Very simplified representation of the input-output table

The relations between the elements of the table can be represented by a set of linear equations, each one describing the distribution of a producer's output throughout the economy.

$$\begin{aligned} x_1 &= z_{11} + z_{12} + \ldots + z_{1n} \quad + y_1 \\ x_2 &= z_{21} + z_{22} + \ldots + z_{2n} \quad + y_2 \\ \ldots \\ x_n &= z_{n1} + z_{n2} + \ldots + z_{nn} \quad + y_n \end{aligned} \tag{1.1}$$

The main assumption of input-output model is that the inter-industry flows from sector *i* to sector *j* during a specific period (say 1 year) depend entirely and exclusively on the total output of sector *j* for the same period of year. So, it is possible to calculate *inter-industry coefficients* a_{ij} by reporting the flows from sector *i* to sector *j* (in monetary or physical units) to the associated amount of production from sector *j*.

$$a_{ij} = \frac{z_{ij}}{x_j} \tag{1.2}$$

The inter-industry coefficients a_{ij} define the production technology of the sector *j* and can also be interpreted as the amount of input *i* required to produce 1 unit of output from sector *j*.

$$a_{ij} = \frac{amount\ of\ \ i\ required}{output\ of\ \ j} \tag{1.3}$$

The matrix containing the inter-industry coefficients is called the *inter-industry coefficient matrix* and is denoted by A.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{1.4}$$

Each column is associated with a certain sector and describes the technology of the sector. Implicit to the technology are the direct requirements from all sectors required to obtain 1 unit from its own output. For each sector, the associated line represents the distribution of the output to the rest of the sectors in the entire economy. Because of this interpretation, matrix *A* completely describes all the input and output flows among the sectors of an economy.

The output of an economy consists of two parts: *intermediate demand* and *external demand*. Intermediate demand is used to cover the demand between the various sectors in the economy. External demand is the requirement of products that the economy has to deliver to final consumers (households, government, investments, net exports). Together, they make the *total output of an economy*, represented as a vector *x,* required for a specific final demand. Elements x_i are the outputs of every sector *i*, as in equation 1.5:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{1.5}$$

The external demand can also be represented as a vector *y* with elements y_i for every sector as in equation 1.6:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{1.6}$$

The output of an industry is dependent by the output of the other sectors and from the external demand; making the substitution $z_{ij} = a_{ij}x_j$ from equation 1.2 in equations 1.1, an economy with 3 sectors can be described by the equations:

$$\begin{aligned} \overbrace{x_1}^{\text{output}} &= \overbrace{a_{11}x_1 + a_{12}x_2 + a_{13}x_3}^{\text{intermediate demand}} + \overbrace{y_1}^{\text{final demand}} \\ x_2 &= a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + y_2 \\ x_3 &= a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + y_3 \end{aligned} \tag{1.7}$$

Equations 1.7 can be expressed in a generalized form using the output vector *x*, inter-process coefficient matrix *A* and final demand vector *y* as:

$$x = Ax + y \tag{1.8}$$

Equation 1.8 is solved by finding the output vector x (A and y being given):

$$\begin{aligned} x - Ax &= y \\ (I - A)x &= y \\ x &= (I - A)^{-1} y \end{aligned} \quad \text{(where } I \text{ is the Identity matrix)} \tag{1.9}$$

Introducing the notation $L = (I - A)^{-1}$ equation 1.9 becomes

$$x = Ly \tag{1.10}$$

The term L is called the Leontief Inverse, after Wassily Leontief.

To the economy level, the Leontief inverse gives the relation between the final demand and total quantity produced. Breaking down the Leontief inverse to the sector level, for a 3 sectors industry, we get:

$$L = \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \tag{1.11}$$

where

$$l_{ij} = \frac{output\ of\ i\ produced}{final\ demand\ of\ j} \tag{1.12}$$

The coefficients of the L matrix, $l_{ij,}$ represent the amount of output of sector i per unit of final demand of sector j. The columns in the Leontief inverse contain the output from all sectors required per unit of external demand of each sector. The lines in the Leontief inverse contain the output from each sector required to satisfy the final demand from all sectors.

The Leontief inverse can be decomposed in a geometric series expansion as in equation 1.13:

$$\sum_{t=0}^{\infty} A^t = I + A + A^2 + \ldots + A^n = (I - A)^{-1} = L \ \ if \ \ \rho(A) < 1 \tag{1.13}$$

In equation 1.13 $\rho(A)$ is the spectral radius of the matrix A and $\rho(A) = \max|\lambda|$ as matrix eigenvalue.

Multiplying both parts of the equation 1.13 with y, we obtain:

$$y + Ay + A^2 y + A^3 y + ... + A^n y = (I - A)^{-1} y = Ly = x \qquad (1.14)$$

The terms of this series are denoted by *t*, from tiers. The first term is called the zeroth tier because $A^0 y = Iy = y$. The second term *Ay* corresponds to tier 1 and represents the output of the first round of activities (tier 0) initiated in the economy as a result of final demand. The third term $A^2 y = A(Ay)$ represents the output of the second round of activities (tier 1) initiated in the economy as a result of a demand of *Ay* imposed by the final demand on the first round of activities, and so on. The output decreases with every additional tier and the series converges $(\rho(A) < 1)$. This formula gives a way to calculate the outputs of the sectors in any tier generated by an external final demand imposed on the economy.

If the outputs are scaled with the associated environmental impact, we can determine the impact of every activity at any tier. The procedure for this is explained in detail in the next chapter.

2.2 Contribution analysis

Contribution analysis is a type of analysis used to calculate total emissions and environmental loads in general, for a given final demand. Using a more general term such as stressors makes it possible to include not only emissions as CO_2, CH_4, NO_x, but also land use, heat waste and others as contributors to the environmental impact.

As I mentioned in the previous chapter, the activity in a sector causes activity in other sectors of the economy through the chains of requirements, directly or via other processes. The activity in the instigating sector is called the *direct activity*. The activity in all other sectors as a result of the final demand in the instigating sector is called *indirect activity*. The *total activity* is the direct activity plus the indirect activity.

In the process of identifying the emissions from each sector per unit of final demand required from the entire network, we can multiply the *emission factors* with the activity and find the emissions generated in each node as a result of requirements of final demand. Similarly with the distinctions made in the previous paragraph, we distinguish between *direct emissions*, generated by the instigating sector, and *indirect emissions*, generated by indirect activity in the other sectors. *Total emissions* totalize the direct emissions and indirect emissions.

The emission factor is analog to the requirements coefficients from *A* matrix. The *stressor intensity matrix F* contains the amount of environmental stressors associated with the output of each process. For a 3 sector economy, the stressor intensity matrix looks as in equation 2.1:

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{2.1}$$

Every line in the above matrix corresponds to a specific stressor and every column corresponds to a specific sector. The number of stressors included varies depending on the study. Generalizing, f_{ij} represents the amount of stressor i generated per unit output of sector j, also known as *stressor intensity*, see equation 2.2.

$$f_{ij} = \frac{amount\ of\ i\ produced}{output\ of\ j} \tag{2.2}$$

The vector of stressors associated with a given final demand is obtained by multiplying the stressor intensity matrix per unit of output F, the matrix of outputs per unit of external final demand L and the final demand vector y. Using a set notation as *sec* for sectors and *str* for stressors, e, the *vector of stressors generated for a given external demand,* is calculated as:

$$e = \begin{bmatrix} e_1 \\ \vdots \\ e_{str} \end{bmatrix} = \begin{bmatrix} f_{11} & \cdots & f_{1,\sec} \\ \vdots & \ddots & \vdots \\ f_{str,1} & \cdots & f_{str,\sec} \end{bmatrix} \begin{bmatrix} l_{11} & \cdots & l_{1,\sec} \\ \vdots & \ddots & \vdots \\ l_{\sec,1} & \cdots & l_{\sec,\sec} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{\sec} \end{bmatrix} \tag{2.3}$$

In a more compact form, equation 2.3 becomes

$$e = FLy = F(I - A)^{-1} y = Fx \tag{2.4}$$

Vector e gives the total amount of stressors associated with the external demand given by the y vector.

To find out how much the various sectors contributes to the total stressor load, the total output vector x has to be diagonalized as follows:

$$E = F\hat{x} \text{ , where } x = Ly = L(I - A)^{-1} y \tag{2.5}$$

E is called the matrix *of stressors generated from each sector for a given final demand* and is defined as:

$$E = \begin{bmatrix} e_{11} & \cdots & e_{1,\sec} \\ \vdots & \ddots & \vdots \\ e_{str,1} & \cdots & e_{str,\sec} \end{bmatrix} \tag{2.6}$$

The elements e_{ij} of matrix E represent the amount of stressor from sector i generated in sector j as a result of the given final demand.

$$e_{ij} = \frac{amount\ of\ stressor\ from\ i}{output\ of\ j} \tag{2.7}$$

The lines corresponding to each stressor in E matrix show the distribution of total emission generated by a certain final demand between all the sectors taking part to the production of that final demand. Summing the rows from E gives the total stressor vector e:

$$\sum_{\text{sec}} E_{str,\text{sec}} = e \tag{2.8}$$

The E matrix calculated in equation 2.5 relates to the total output and calculates the total impact generated from each sector for every stressor taken into consideration; more precisely, the elements e_{ij} represent the amount of stressor originated in sector j, <u>in all tiers</u>, as a result of the final demand. It is possible to decompose the impact of every element e_{ij} in the impacts <u>to every tier</u>, by using the same technique: multiplying F with the diagonalized output at every tier calculated with the geometric series expansion in equation 1.14.

$$F(I-A)^{-1}y = Fy + FAy + FA^2y + FA^3y + FA^4y + \ldots \tag{2.9}$$

Table 2.2 gives an overview on how the total outputs and impacts decompose in outputs and impacts at every tier and how they relate to each other:

Tier	Output at Tier	Impact at Tier
0	$x_0 = A^0 y = y$	Fy
1	$x_1 = Ax_0 = Ay$	$F\hat{x}_1$
2	$x_2 = Ax_1 = A^2 y$	$F\hat{x}_2$
3	$x_3 = Ax_2 = A^3 y$	$F\hat{x}_3$
$\vdots$	$\vdots$	$\vdots$
n	$x_n = Ax_{n-1} = A^n y$	$F\hat{x}_n$

Table 2.2: Impact at tier generated by Geometric Series Expansion

<u>Particular case: contribution analysis for only one stressor</u>

In the particular case when only one stressor is considered, the stressor intensity matrix *F* is reduced to one line, with values for every sector of the economy. For an economy with m sectors, F becomes:

$$F = [f_1 \quad f_2 \quad \ldots \quad f_m] \tag{2.10}$$

The vector of stressors *e* is reduced to only one value (for the chosen stressor), see equation 2.11

$$e = [f_1 \quad f_2 \quad \ldots \quad f_m] \begin{bmatrix} l_{11} & \cdots & l_{1,m} \\ \vdots & \ddots & \vdots \\ l_{m,1} & \cdots & l_{m,m} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{2.11}$$

The total output at every tier and the associated impact are calculated according to the formulas from Tabel 2. A tier n, the output vector x_n will have the form from equation 2.12 *(*$x_1, x_2, \ldots x_m$ are the outputs from m sectors and are different from $x_1, x_2, \ldots x_n$ from table 2.2, where they represent the output per n tiers)

$$x_n = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{2.12}$$

The elements of x_n represent the output at tier n from every m sector. The associated environmental impact is obtained by multiplying the output from every sector with the stressor intensity from the same sector, as obvious from the result of equation 2.13.

$$F\hat{x}_n = [f_1 \quad f_2 \quad \ldots \quad f_m] \times \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & x_m \end{bmatrix} = [f_1x_1 \quad f_2x_2 \quad \ldots \quad f_mx_m] \tag{2.13}$$

It is very important to mention one thing: if the total output of a sector at a specific tier is used by more sectors in previous tier, it is not possible to differentiate the amount to be used in one sector by the amount to be used in a different sector. Exemplifying, at tier 3, electricity can be used for melting metals in different sectors but also for lighting offices and homes, but we do not know how much is used for melting metals and how much for lighting; all we know is the total amount produced and total environmental impact. It is not possible to calculate, using only the above methodology, how much from the total impact of a sector at a specific tier is generated as a result of different sectors requirements. The next chapter will show how to use structural path analysis to solve this problem.

2.3 Structural path analysis

2.3.1 Problem formulation

In any economy, the demand from a producing sector for a certain product requires direct inputs from other sectors of the economy like raw materials, electricity, material parts and others; even its own products can be used in the production process as in the case of an industry producing light bulbs. The production and supplying sectors form production networks.

Final demand instigates a production network which spans through the entire economy, at country and at world level. The high numbers of processes which potentially can be traced back have specific environmental impacts. For mitigating climate change, it is important to know the processes with the highest environmental impact connected with a specific final demand and to know where these processes occur in the production network.

To acquire this objective, we can start by representing the production network as a graph. Figure 1 is an illustration for a three-sector economy. The nodes of the graph constitute the three sectors, identified by numbers 1, 2 and 3. The final demand is y_1 and applies to sector 1. The arcs connecting the nodes represent flows between sectors, of magnitudes $a_{ij}\ y_i$, where a_{11}, a_{21} and a_{31} represent the inter-industry coefficients. These magnitudes are calculated as in equation 1.2.

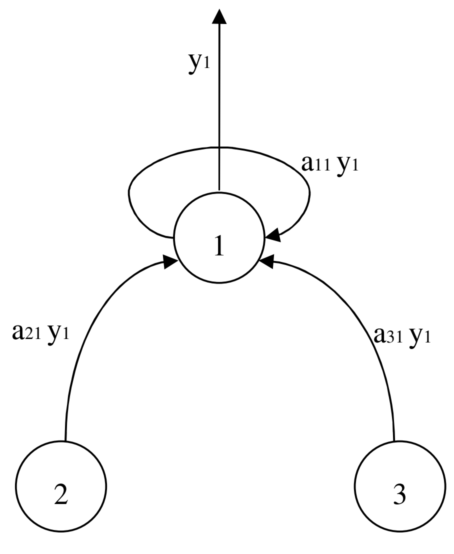


Figure 2.1: Graph representation of direct inputs for a demand y_1 of sector 1 in a three sector economy

Figure 2.1 gives a schematic representation of the relations between sectors 1, 2 and 3, where an external final demand y_1 of sector 1 generates requirements of $a_{21}y_1$ units from sector 2, $a_{31}y_1$ units from sector 3 and $a_{11}y_1$ units from it own production.

The graph representation of the economy can be used only for direct inputs. It must be noted that inputs can have inputs themselves. Representing all inputs to each of the direct inputs and so on by arrows will make the graph very hard to read and use.

The reason above is why it is much more useful to use a tree representation. In such a representation, all linkages in an economy can be visualized. These linkages are calculated with the geometric series expansion from equation 1.14.

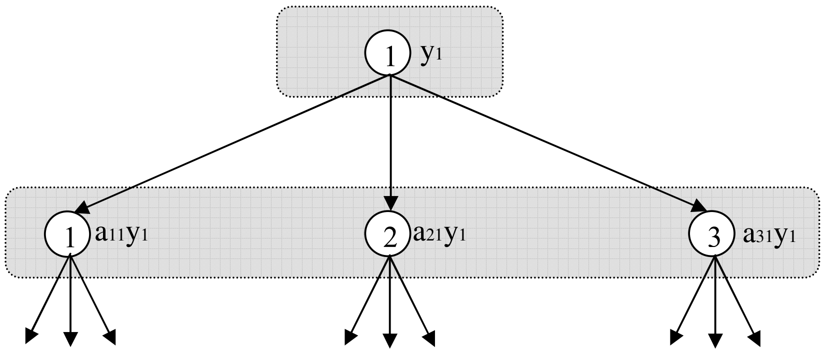


Figure 2.2: Tree representation of direct inputs for a demand y_1 of sector 1 in a three sector economy

Each layer of production is represented in the same line. The final demand from sector 1 (y_1) can be visualized in the first layer. The direct inputs to sector 1 determined by the final demand y_1 can be visualized in the second layer: $a_{11}y_1$ from sector 1, $a_{21}y_1$ from sector 2 and $a_{32}y_1$ from sector 3.

From this, if we use the *amount of environmental impact per unit of output from sector i* (F_i) to scale the output in every single node according to equation 2.12, we get the environmental impact of node *i*.

We are interested here by the environmental output of the inputs required, according to the production perspective. This suggests that we can compare the three nodes and find which has the smallest environmental impact.

However, the inputs to the supplying sectors and the inputs to the suppliers' suppliers need also to be taken into account. Because the same sector names can be used in different tiers and even many times in the same tier, a way of identifying the sectors in every instance is needed. We call the succession of sectors linked by a specific requirement a *path*. Path $i \rightarrow j \rightarrow k$ suggests that sector *i* require an input from sector *j* which in turn, requires an input from sector *k*. Sectors *i, j* and *k* are differentiated by the output requested from them in different layers of production, not by name. Path $1 \rightarrow 3 \rightarrow 1$ describes the fact that a final demand from sector 1 requires an output from sector 3 which in turn, requires a different output from sector 1.

Using the inter-industry coefficients we can calculate the output in the next layer for every sector. For example, for an output of $a_{21}y_1$ units from sector 2, the following are needed: inputs of $a_{12}a_{21}y_1$ units from sector 1 (path $1 \rightarrow 2 \rightarrow 1$); $a_{22}a_{21}y_1$ units from sector 2 (path $1 \rightarrow 2 \rightarrow 2$); and $a_{32}a_{21}y_1$ units from sector 3 (path $1 \rightarrow 2 \rightarrow 3$). The associated environmental impacts with these outputs are $F_1 a_{12}a_{21}y_1$ for path $1 \rightarrow 2 \rightarrow 1$, $F_2 a_{22}a_{21}y_1$ for path $1 \rightarrow 2 \rightarrow 2$ and $F_3 a_{32}a_{21}y_1$ for path $1 \rightarrow 2 \rightarrow 3$.

Generalizing, the environmental impact of path $i \rightarrow j \rightarrow k$ can be calculated with equation

$$f_{i \rightarrow j \rightarrow k} = F_k a_{kj} a_{ji} y_i \tag{2.9}$$

In a more explanatory form, equation 2.9 can be understood as:

$$f_{i \rightarrow j \rightarrow k} = env.impact\ per\ unit\ of\ output\ from\ k \times \frac{output\ of\ k}{output\ of\ j} \times \frac{output\ of\ j}{output\ of\ i} \times final\ demand\ i$$

From the production perspective studied here, it is the activity from the end of the chain which causes the environmental impact.

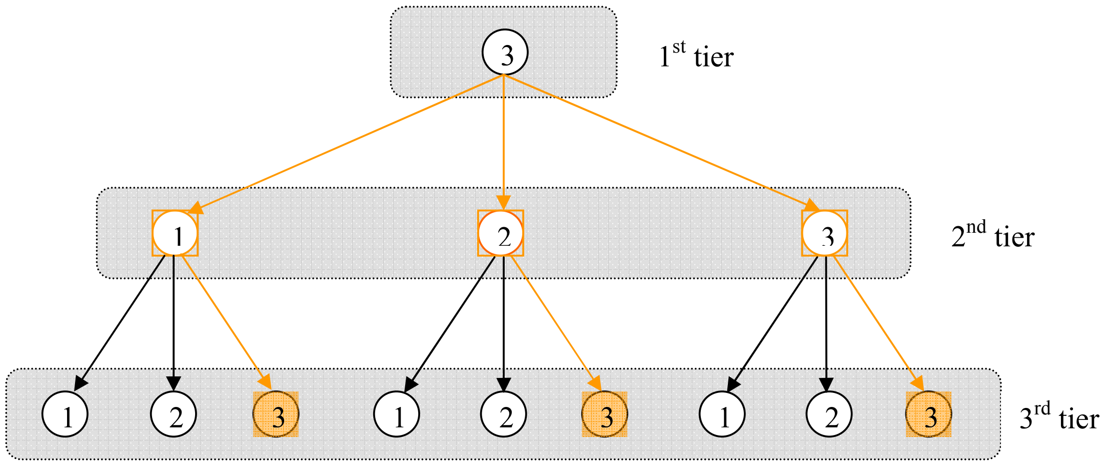


Figure 2.3: Tree representation for the first 3 tiers of a three sector economy (melt metals, light offices, electricity) for a final demand of melting one unit of metals

Now let us return to the electricity example from the end of subchapter 2.2. The three sectors, presented in figure 2.3, are: electricity (identified by “3” in the figure), metals melting ("1") and lighting offices ("2"). We use the above formula to differentiate between the 3 instances of using electricity in the third tier: as its own requirement, as a requirement of the metals melting sector, as a requirement of lighting offices sector.

Let us consider a totally hypothetical case in which melting 1 unit of metal requires 0.02 units from itself, 0.5 units of office lighting and 0.1 units of electricity; lighting offices requires 0.1 units of melting metal, 0.03 units from itself and 0.4 units of electricity; electricity production requires 0.1 units from melting metals, 0.1 units of office lighting and 0.04 units of itself; the environmental impact is 3 units stressor/1 unit melting metals, 5 units stressor/1 unit of lighting and electricity require 1 unit stressor/1 unit electricity. The inter-industry requirements matrix (A), final demand (y) and stressor matrix (F) are presented below:

$$A = \begin{bmatrix} 0.02 & 0.1 & 0.1 \\ 0.5 & 0.03 & 0.1 \\ 0.1 & 0.4 & 0.04 \end{bmatrix}; \quad y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \quad F = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix};$$

Doing the calculations according with the methodology presented in this chapter, we get:

(1) environmental impact at first tier: e_1

$$e_1 = F\hat{y} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

All impact is concentrated to the electricity sector (1 unit); the other the sectors are not engaged in production system yet, so that's why they show no impact at this tier.

(2) environmental impact at second tier: e_2

$$x_2 = Ay = \begin{bmatrix} 0.02 & 0.1 & 0.1 \\ 0.5 & 0.03 & 0.1 \\ 0.1 & 0.4 & 0.04 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.04 \end{bmatrix}$$

$$e_2 = F\hat{x}_2 = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.04 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.5 & 0.04 \end{bmatrix}$$

The impact at second tier is of 0.3 stressor units for melting metals sector, 0.5 stressor units for lightning offices sector and 0.04 stressor units for electricity sector.

(3) environmental impact at third tier: e_3

$$x_3 = A^2 y = Ax_2 = \begin{bmatrix} 0.02 & 0.1 & 0.1 \\ 0.5 & 0.03 & 0.1 \\ 0.1 & 0.4 & 0.04 \end{bmatrix} \times \begin{bmatrix} 0.1 \\ 0.1 \\ 0.04 \end{bmatrix} = \begin{bmatrix} 0.016 \\ 0.057 \\ 0.0516 \end{bmatrix}$$

$$e_3 = F\hat{x}_3 = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 0.016 & 0 & 0 \\ 0 & 0.057 & 0 \\ 0 & 0 & 0.0516 \end{bmatrix} = \begin{bmatrix} 0.048 & 0.285 & 0.0516 \end{bmatrix}$$

The impact at third tier is of 0.048 stressor units for melting metals sector, 0.285 stressor units for lightning offices sector and 0.0516 stressor units for electricity sector.

Resuming, electricity sector has an environmental impact of 1 stressor unit in first tier, 0.04 stressor units in the second tier and 0.0516 stressor units in the third tier.

The impact at first tier corresponds to path 3:

$$f_{path3} = f_3 y_3 = 1 \times 1 = 1$$

The impact at second tier corresponds to path $3 \to 3$:

$$f_{3\to 3} = f_3 a_{33} y_3 = 1 \times 0.04 \times 1 = 0.04$$

The impact to the third tier however, corresponds to three different paths: $3 \to 1 \to 3$, $3 \to 2 \to 3$ and $3 \to 3 \to 3$. The calculated value (0.0516 stressor units) is not differentiated to each of the above paths.

To differentiate the impacts we use the formula 2.9:

$$f_{3\to 1\to 3} = f_3 a_{31} a_{13} y_3 = 1 \times 0.1 \times 0.1 \times 1 = 0.01$$

$$f_{3\to 2\to 3} = f_3 a_{32} a_{23} y_3 = 1 \times 0.4 \times 0.1 \times 1 = 0.04$$

$$f_{3\to 3\to 3} = f_3 a_{33} a_{33} y_3 = 1 \times 0.04 \times 0.04 \times 1 = 0.0016$$

From the total 0.0516 stressor units corresponding to electricity use in tier 3, 0.01 stressor units occur in metal melting sector, 0.04 stressor units are produced by offices lighting and 0.0016 stressor units are produced in electricity sector.

If we want to rank the paths ending with sector electricity, in the descending value of the environmental impacts, we have: (1) f_{path3} (1 stressor unit), (2) $f_{3\to 2\to 3}$ (0.04 stressor units), (3) $f_{3\to 3}$ (0.04 stressor units), (4) $f_{3\to 1\to 3}$ (0.01 stressor units), (5) $f_{3\to 3\to 3}$ (0.0016 stressor units). Because the second and the third paths have the same environmental impact, the ranking for them is arbitrary.

In this example we used a very simplified three sector economy and calculate the emissions only until the third tier. For more complex systems and more tiers, the number of combinations grows exponentially. For an industry with 10 sectors, there will be 10^{10} combinations possible until the 10th tier. The United States has an input-output table with ~500 sectors and if we want to go to the product level, using a LCA database, we can have a matrix with a few thousands rows and columns. The trees constructed from these matrices are very big, even for a small number of tiers. In conclusion, in practice it is not possible to consider all these processes, even if everyone has its specific environmental impact. The solution is to 'prune' the tree for reducing the number of possible solutions. We present here too different ways of pruning the tree: one way is to use IO methodology and will be explained in chapter 3; another way is to use a specialized algorithm which is explained below.

2.3.2 Specialized algorithm description

Peters and Hertwich (2006) used a dynamic tree data structure to extract the necessary paths. Their method has 2 steps: (1) construct the tree structure and (2) read the tree by 'de-constructing' it.

First it is important to explain what tree structure is and how to extract the paths using the structure fields of a general tree structure

The tree structure is based on the representation from figure 2.2. The tree spanning from a node is a structure, characterized by three fields: (1) the contribution of the node, (2) the path leading to the node and (3) an array of pointers to the next sub-trees. Each pointer makes the connection between the tree and one of its sub-trees.

Adding a sub-tree to the previous structure determine creating a new structure. The path field of the new structure includes the pointers to the previous structures and the pointer to itself.

Reading the tree by deconstructing it means to use the array of pointers to identify the successions of tree structures. Each pointer is connected with the sector in the main node.

The sectors from the main nodes of these tree structures create the paths. The contribution of every path is found in the first field of the last structure: the contribution of the node. This value is calculated with formula 2.9.

Next part presents how to construct the particular tree structure determined by the final demand starting from the general tree structure.

A recursively algorithm checks all sub-trees deciding if they are kept or deleted. If the sub-tree is kept, the pointer to the main node of the sub-tree becomes part of the path characterizing the sub-tree structure. If the sub-tree is not kept, the tree is said to be 'pruned'.

Main structure has a finite number of tiers. Sub-trees spanning from nodes at the last tier are automatically deleted. For all others sub-trees, the algorithm decide if a sub-tree is added or deleted to the previous tree based on the total environmental contribution of the entire sub-tree. If the contribution is above a given threshold, the sub-tree is added to the previous tree. If the contribution is below a given threshold, the sub-tree is deleted from the previous tree.

The total environmental contribution of the entire sub-tree (originating in a node and identified by a path) is calculated with formula 2.4 applied to the main node of the sub-tree. For every sub-tree spanning from a node, the algorithm follows three steps. The next example is for a sub-tree spanning from a node at the 3rd tier which can be find by the pointers indicating path $i \to j \to k$.

(1) calculates the final demand vector at the main node *k* (i.e. $y_{i\to j\to k} = a_{kj}a_{ji}y_i$)

(2) calculates the output associated with the final demand at the node (i.e. $x_{i\to j\to k} = Ly_{i\to j\to k}$)

(3) calculates the direct and indirect environmental impacts in the sub-tree below that node (i.e. $subtree_{i\to j\to k} = f_k x_{i\to j\to k}$)

As we mentioned before, the value calculated above is compared with the threshold to find if the sub-tree is added to the previous tree or deleted.

The resulting tree will have an irregular structure, due to the varying depths that the algorithm penetrates in searching for new nodes (see figure 3.3 in chapter 3).

After the tree has been constructed and the paths recovered, the main algorithm sorts the paths in descending order and prints them in an Excel file.

This algorithm was programmed in Matlab version 7. Because the Matlab programming language was not designed for using dynamic data structures, the authors had to employ inefficient code to represent null pointers (Peters and Hertwich, 2006).

The performance of this algorithm will be used for benchmarking the result of the new Gams SPA in chapter 4.

2.4 Mathematical programming

In many cases, the applications of mathematical programs "have been so successful that their use has passed out of operational research department to became an accepted routine planning tool" (Williams 1999).

Using mathematical programming in solving real life situations requires a model. A model is a very concise and comprehensive representation of the reality, in its most essential features needed for study. A mathematical model of the production network will translate the relationships between the sectors, as elements of the network, into mathematical relationships as equations, inequalities, logical interdependencies, etc. The model does not include the data as the quantification of the relationships between its elements. This means that a model can run on different data sets and this feature makes it very useful for the decision process in general.

There are a lot of advantages in using mathematical models: (1) a greater understanding of the problem, arising from the struggle to formulate the problem in a theoretical form, (2) a mathematical model may suggest solutions which are not very evident from the beginning, (3) it is possible to experiment different solutions, especially when their effect might be harmful in reality (example: a new tax system).

Beside their versatility by using different data sets, models are especially useful when a solution can be obtained in a very short time, compared with manual computation. The models with very complex relationships between their elements or/and containing large data sets might require a lot of time and effort for obtaining a solution using hand computations. The development of computer programming made it possible to do all the computation automatically, speeding up the process a lot.

The user does not even require knowledge of algorithms for solving a linear program. Commercial package programs like GAMS have very efficient built-in algorithms for solving a linear program. The only requirements are having a good model and using the syntax in a correct way.

The aim of any mathematical programming model is to make the system more efficient that it was before. In mathematical form, efficiency can be represented by an expression which needs to be maximized or minimized. That expression is the *objective function*. Other expressions known as *main constraints* can be built in order to model the restrictions or interactions of the system which limit decision. These expressions must not exceed some specified value $(\leq)$ or must not fall below a specified value $(\geq)$ or must exactly equal a specified value $(=)$.

The decisions to be taken are represented by the *decision variables*. Some limits (called variable-type constraints) can be placed on the type of these variables. The variables can be non-negative (positive), integer, binary or even non-restricted (free). By contrast, the data characterizing the relationships and requirements between the elements of the system which are measured and considered fixed are known as *input parameters.* (Rardin, 1998)

When the variables are continuous and both the objective function and the constraints are linear expressions, the model is called a *linear program* (LP).

When there are conventional variables mixed with integer variables, the model is said to be a *mixed-integer programming model* (MIP).

Most practical decision problems restrict the integer to only two values, 0 or 1. They are used to represent 'yes' or 'no' decisions (Williams, 1999). This feature will be used in the program developed in the next chapter.

A very important issue about integer programming models is the fact that solving them is much more difficult, more time-consuming and requires several times as many computations as similar size linear programming models (Williams, 1999).

One of the main advantages of a linear programming model for SPA is the fact that is mathematically feasible. The mathematical representation of the input-output model takes the form of a linear system, as in equations 1.7: $(I - A)x = y$. This is a *set of constraints*, in which we can identify the *decision variables* as *x* and *input parameters* as *(I - A)*. Finding the paths with the greatest environmental impact can be formulated as an *objective function*. It is possible to formulate the problem so as all decision variables to be *binary variables*. (see chapter 3). The result is a MIP model for SPA.

The Gams site about MIP models stresses on the characteristics of MIP compared with similarly sized pure linear programs: require dramatically more mathematical computation, require significant amounts of physical memory and take enormous amounts of time to solve. http://www.gams.com/dd/docs/solvers/cplex.pdf

These types of problems are solved by Gams using CPLEX solver. This solver employs a generic algorithm which will be briefly presented in the next chapter. According to the description from the Gams original site, CPLEX solver includes "state-of-the-art implementations of simplex and barrier algorithms" http://www.gams.com/solvers/solvers.htm#CPLEX . This evaluation shows an increased likelihood of obtaining better results than a more conservative approach to the same problem.

Peters and Hertwich' algorithm use a conservative approach, because the tree search examines all solutions one by one. This approach can be inefficient because it will follow many paths with a very small value before ruling them out. But the algorithm is designed specifically for this problem, so no additional constraints are needed in order to apply it.

CPLEX solver is very fast, as mentioned above, but the problem needs some adaptations to make it suitable for the solver.

In conclusion, Matlab-SPA advantage of being tailor specifically for the problem is contra balanced by disadvantage in algorithm efficiency. Gams-SPA has the advantage of a very efficient algorithm but the disadvantage of a poorly formulated problem. These contradictory features makes impossible to say which method is more suitable for a SPA without testing. This is the aim of this thesis and the SPA algorithm developed in the next chapter will be benchmarked against the existing algorithm in chapter 4.

Chapter 3

Mathematical programs for Structural Path Analysis

This chapter starts with a review of the SPA problem. Then three mixed integer formulations of increased optimization of the problem are presented. Implementation details presented in the second subchapter will bring into light the challenges associated with coordinating Matlab and Gams applications and how these challenges have been tackled for solving this problem.

3.1 Mixed integer program for SPA

IO analysis and LCA have the same methodology with the specification that 'sectors' in IOA are replaced by 'processes' in LCA. Because of the common methodology, a SPA for an IO system is perfectly applicable to a LCA system. The IO nomenclature used in the previous chapter will also be used in this chapter, along with LCA nomenclature, to explain the mathematical programs for SPA. This will make references to the previous chapter easier to understand.

The SPA model presented here finds all paths describing the process relationships that link the final delivered product and the emitting processes. We use the tree representation of the economy, as defined in the previous chapter (see figure 2.3). For restricting the number of paths, two conditions are imposed: (1) use maximum 10 tiers and (2) ignore all paths with the environmental impact below a certain threshold value *Tol*.

This section presents the theoretical aspects of three mixed integer programs for SPA in the order of increased optimization. The implementation details will be presented in the next subchapter.

Given a set of sectors/processes S, with elements *sec*, the following input parameters are defined:

1. $a_{sec,sec'}$ as the amount of sector/process *sec* required per unit amount of sector/process *sec'*; these input parameters are the elements of the inter-process requirements matrix/inter-industry coefficient matrix: *A*.

$$a_{\text{sec},\text{sec}'} = \left\{ A(\text{sec},\text{sec}') \mid \text{sec} \in S \textit{ and } \text{sec}' \in S \right\} \tag{3.1}$$

2. f_{sec} as the amount of stressor per unit amount of sector/process *sec*; these are the elements of the vector of stressors per sectors *F*.

$$f_{\text{sec}} = \left\{ F(\text{sec}) \mid \text{sec} \in S \right\} \tag{3.2}$$

3. y_{sec} as the amount of final demand placed upon sector *sec*; these are the elements of the vector of final demand/external demand y:

$$y_{\sec} = \{y(\sec) \mid \sec \in S\} \tag{3.3}$$

Every *path* until a certain tier is constructed as a succession of sectors/processes, with one sector/process per each component tier, respecting the tier order. These sectors/processes are the *elements of the path*; the number of sectors/processes gives the *length of the path*. Because a path contains only one sector per tier the length of the path corresponds to the maximum tier reached by the path.

Not all sectors/processes can be elements of the path. As a result of presorting in Matlab, only a reduced number of sectors/processes are considered at each of the 10 existing tiers. These subsets are presented in table 3.1 below:

Subset name	containing the sectors/processes at tier	subset elements
I	***1st tier***	$i = \{\sec \mid \sec \in I\}$
J	***2nd tier***	$j = \{\sec \mid \sec \in J\}$
K	***3rd tier***	$k = \{\sec \mid \sec \in K\}$
L	***4th tier***	$l = \{\sec \mid \sec \in L\}$
M	***5th tier***	$m = \{\sec \mid \sec \in M\}$
N	***6th tier***	$n = \{\sec \mid \sec \in N\}$
Q	***7th tier***	$q = \{\sec \mid \sec \in Q\}$
R	***8th tier***	$r = \{\sec \mid \sec \in R\}$
T	***9th tier***	$t = \{\sec \mid \sec \in T\}$
U	***10th tier***	$u = \{\sec \mid \sec \in U\}$

Table 3.1: Subsets defining

Any combination of subset elements taken in tier order defines a possible path. Paths can be grouped based on their length. Table 3.2 shows in the second column all groups of paths up to each tier:

Tier	Paths groups at tier	Path name	Path value
1st tier	i	$\boldsymbol{b_i}$	$\boldsymbol{e_i}$
2nd tier	$i \to j$	$\boldsymbol{b_{ij}}$	$\boldsymbol{e_{ij}}$
3rd tier	$i \to j \to k$	$\boldsymbol{b_{ijk}}$	$\boldsymbol{e_{ijk}}$
4th tier	$i \to j \to k \to l$	$\boldsymbol{b_{ijkl}}$	$\boldsymbol{e_{ijkl}}$
5th tier	$i \to j \to k \to l \to m$	$\boldsymbol{b_{ijklm}}$	$\boldsymbol{e_{ijklm}}$

6th tier	$i \to j \to k \to l \to m \to n$	**b_{ijklmn}**	**e_{ijklmn}**
7th tier	$i \to j \to k \to l \to m \to n \to q$	**$b_{ijklmnq}$**	**$e_{ijklmnq}$**
8th tier	$i \to j \to k \to l \to m \to n \to q \to r$	**$b_{ijklmnqr}$**	**$e_{ijklmnqr}$**
9th tier	$i \to j \to k \to l \to m \to n \to q \to r \to t$	**$b_{ijklmnqrt}$**	**$e_{ijklmnqrt}$**
10th tier	$i \to j \to k \to l \to m \to n \to q \to r \to t \to u$	**$b_{ijklmnqrtu}$**	**$e_{ijklmnqrtu}$**

Table 3.2: Paths groups, names and values, per tiers

Every path to be found is a *decision variable* in this model, denoted by *b*. The elements of the path became the elements of associated decision variable. The decision variables can be grouped by length in the same way as the underlying path, see the third column of table 3.2.

The decision variables are of *binary type*, because every possible path is either considered or ignored in the optimization model.

The *impact* of every possible path is presented as a parameter in the 4th column of table 3.2. The impact of each path, denoted by *e*, can be calculated using formula 2.9 from the values of *A, F* and *y*.

The problem is to determine the paths $b_i, b_{ij}, b_{ijk}, ...b_{ijklmnqrtu}$ as presented in Table 3.2. For low thresholds, the number of paths is too large and not all can be return. Only the paths with impact above the threshold value will be returned. The reason for choosing this criterion for reducing the number of paths will became evident after we discuss memory implications for path transfer from Gams to Matlab.

For a path $i \to j \to k$, the condition can be written as:

$$b_{ijk} Tol \leq e_{ijk} \tag{3.4}$$

There are two cases to consider here:

a. The value of the path is above the threshold value *Tol* and in this case $b_{ijk} = 1$.
b. The value of the path is below the threshold value *Tol* and in this case $b_{ijk} = 0$.

Similar conditions can be written for all paths.

The mathematical formulation can now be stated as:

$$\max \sum_i e_i b_i + \sum_i \sum_j e_{ij} b_{ij} + \sum_i \sum_j \sum_k e_{ijk} b_{ijk} + \sum_i \sum_j \sum_k \sum_l e_{ijkl} b_{ijkl} + \sum_i \sum_j \sum_k \sum_l \sum_m e_{ijklm} b_{ijklm}$$
$$+ \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n e_{ijklmn} b_{ijklmn} + \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q e_{ijklmnq} b_{ijklmnq}$$
$$+ \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r e_{ijklmnqr} b_{ijklmnqr} + \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r e_{ijklmnqr} b_{ijklmnqr}$$
$$+ \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \sum_t e_{ijklmnqrt} b_{ijklmnqrt} + \sum \sum \sum \sum \sum \sum \sum \sum \sum \sum e_{ijklmnqrtu} b_{ijklmnqrtu} \quad (3.5)$$

$$\text{s.t. } b_i Tol \le e_i \qquad \forall i \in I \qquad (3.6)$$
$$b_{ij} Tol \le e_{ij} \qquad \forall i \in I, \forall j \in J \qquad (3.7)$$
$$b_{ijk} Tol \le e_{ijk} \qquad \forall i \in I, \forall j \in J, \forall k \in K \qquad (3.8)$$
$$b_{ijkl} Tol \le e_{ijkl} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L \qquad (3.9)$$
$$b_{ijklm} Tol \le e_{ijklm} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M \qquad (3.10)$$
$$b_{ijklmn} Tol \le e_{ijklmn} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, \forall n \in N \qquad (3.11)$$
$$b_{ijklmnq} Tol \le e_{ijklmnq} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, \forall n \in N, \forall q \in Q \qquad (3.12)$$
$$b_{ijklmnqr} Tol \le e_{ijklmnqr} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, \forall n \in N, \forall q \in Q, \forall r \in R \qquad (3.13)$$
$$b_{ijklmnqrt} Tol \le e_{ijklmnqrt} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, \forall n \in N, \forall q \in Q, \forall r \in R, \forall t \in T \qquad (3.14)$$
$$b_{ijklmnqrtu} Tol \le e_{ijklmnqrtu} \qquad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, \forall n \in N, \forall q \in Q, \forall r \in R, \forall t \in T, \forall u \in U \qquad (3.15)$$

$$b_i, b_{ij}, b_{ijk}, b_{ijkl}, b_{ijklm}, b_{ijklmn}, b_{ijklmnq}, b_{ijklmnqr}, b_{ijklmnqrt}, b_{ijklmnqrtu} \text{ binary} \qquad (3.16)$$

The total impact of all paths is just a summation of individual impacts of every single path, as part of a group of paths (equation 3.6). Each of the equations 3.6-3.15 is designed for a group of paths and ensures that for each returned path in each group, the environmental impact is above the threshold value. The type of the decision variables is defined in equation 3.16. This is *the most basic form of the SPA model*.

Every new tier adds a new group of decision variables. The *number of decision variables* in the new group can be calculated by multiplying the number of decision variables from the previous group by the number of elements (sectors/processes) from the tier reached. If N_I, N_J, N_K...N_U are the number of elements from subsets *I, J, K*...*U*, the total number of decision variables for a 10-tier tree are:

$$\underbrace{N_I}_{\substack{\textit{no. of paths}\\ \textit{until 1st tier}}} + \underbrace{(N_I \times N_J)}_{\substack{\textit{no. of paths}\\ \textit{until 2nd tier}}} + \underbrace{(N_I \times N_J \times N_K)}_{\substack{\textit{no. of paths}\\ \textit{until 3rd tier}}} + \ldots + \underbrace{(N_I \times N_J \times N_K \times \ldots N_U)}_{\substack{\textit{no. of paths}\\ \textit{until 10th tier}}} \quad (3.17)$$

The *number of constraints* in a group corresponds with the number of new decision variables generated by every new tier considered. Equation 3.17 shows how to calculate the number of constraints.

Example: for a tree with one element in the first tier, five elements in the second tier and seven elements in the third tier, there are: one path until first tier, one corresponding decision variable and one underlying constraint. The second tier adds 1x5=5 paths, equivalent with five decision variables and five constraints in equation 3.8; the third tier adds 1x5x7=35 paths, equivalent with 35 decision variables and 35 constraints in the third group of constraints. The same rule applies for the rest of tiers.

Every new tier adds a new group of decision variables and a new group of constraints. If we group the decision variables and the constraints according to the tier where they are generated, we obtain a *block angular structure* (Williams, 2006). In this structure, the coefficients of each decision variable form *blocks of coefficients*, grouped by tiers. For the objective function, these coefficients are represented by the impacts associated with each decision variable describing a path. They are grouped as line vectors at each tier. The columns of the same coefficients form in this structure the right hand side blocks of coefficients. The diagonally placed blocks of coefficients are known as *submodels*. The submodels are square matrices in which all elements are equal to *Tol*. The dimensions are given by the number of decision variables and the number of constraints, which are equal.

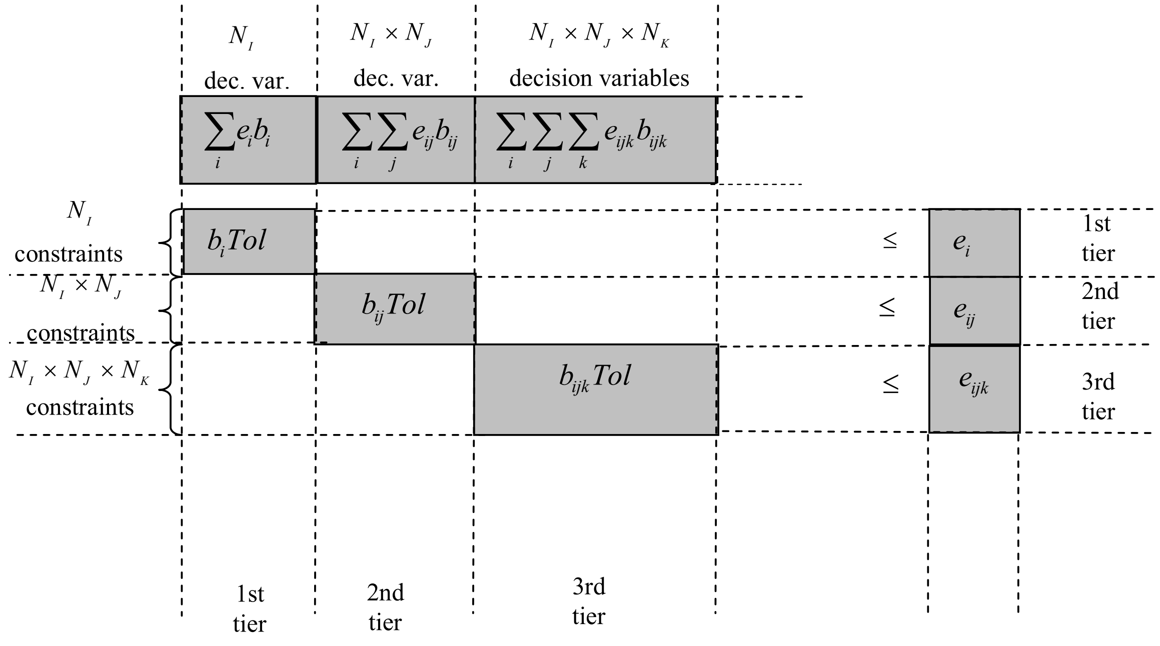

N_I dec. var.
$N_I \times N_J$ dec. var.
$N_I \times N_J \times N_K$ decision variables
$\sum_i e_i b_i$
$\sum_i \sum_j e_{ij} b_{ij}$
$\sum_i \sum_j \sum_k e_{ijk} b_{ijk}$
N_I constraints
$N_I \times N_J$ constraints
$N_I \times N_J \times N_K$ constraints
$b_i Tol$
$b_{ij} Tol$
$b_{ijk} Tol$
$\leq$
e_i
e_{ij}
e_{ijk}
1st tier
2nd tier
3rd tier
1st tier
2nd tier
3rd tier

Figure 3.1 Block angular structure of the model

This model does not have constraints containing decision variables from different tiers. Because of this, a number of *subproblems* can be formed by selecting submodels at each tier and the corresponding portion of the objective. For every tier optimizing this structure simply amounts to optimizing each subproblem. This property of the model is exploited in two ways: (1) for solving the model and (2) for adapting the size of the model to the individual inputs, the necessity of which is explained below.

The number of *decision variables* and *constraints* increases very fast with each new tier as shown above. Also, the model is designed to be appropriate for *very large input data systems*. These two features generate *very large models*. Executing and solving linear programs which describe such models usually require a lot of memory and high computation times. If possible, it is recommended to keep the size of the model as small as possible. Luckily, we can adapt the size of the model to the individual inputs and tackle in this way the memory and time issues.

As a preprocessing result, sometimes all sectors below a certain tier are removed from the tree representation of the economic network. The resulting tree will have only T tiers, with $T \leq 10$. In this tree the longest possible path will be of T tiers long. Paths longer than T do not exist. Consequently, the decision variables underlying these paths are zero and the constraints for tiers higher than T do not apply.

We exploit the angular structure of the model by removing the variables and constraints which do not apply and consequently reducing the size of the model. Unfortunately, T is an endogenous variable calculated in the presorting phase. A less functional alternative is to solve separately 10 models, each model containing first T subproblems, to be used as needed. This alternative can be avoided by introducing a binary switch s_t for each tier.

$$s_t = \begin{cases} 1 & \textit{if } t \leq T \\ 0 & \textit{otherwise} \end{cases} \tag{3.18}$$

The switches s_t are defined as above and are used to control the groups of decision variables and constraints from each subproblem. The variables and constraints from the first T subproblems are switched on while the variables and constraints from the rest of tiers will be switched off. The resulting problem will always have only the minimum number of decision variables and constraints.
Exemplifying: if $T=7$ then all values $s_1, s_2, \ldots s_7$ are equal to 1 and s_8, s_9 and s_{10} are equal to 0. Only the first 7 subproblems are kept for optimization, making sure that we solve only a reduced version of the model every time we change the input parameters.

The second version of the SPA problem includes two changes:

(1) On/off switches has been added for each group of decision variables and constraints. The constraints have been reversed in order to make the switches more visible.

(2) The impact of each path, *e*, has been replaced by the expression used to calculate it. This was necessary because the impact e is not a given parameter, being calculated from the parameters A, F and y.

The second version of the SPA problem is defined as in equations 3.19-3.30.

Gams generates the model before solving it. This includes generating decision variables and constraints and calculating the coefficients in objective function and in constraints. The number of decision variables, the number of constraints and the complexity in calculating all coefficients mentioned above have a direct effect on the size of the model.

The solution is not affected if Gams generates the model by listing decision variables and constraints for all paths, even at the tiers which are not used. However, the bigger the model, the more memory is used to generate it and subsequently to solve it. The size of the model also affects the time to generate and solve it. The aim of the model is to solve as many individual cases as possible, with a given memory and in reasonable time.

The version presented in appendix C includes changes designed to increase functionality, reduce memory usage and increase computational speed. This version was developed specifically developed for implementation in Gams. These changes regard the way equations are generated by Gams. They make sure that only the minimum version of the model is generated before solving. The direct effect consists of reduced memory for listing the equations and consequently, for solving the model. This version also accounts for the highest speed in solving the model. Less efficient alternative formulations will be discussed in the next chapter.

The Gams version of the model presented below has an objective function (equation 3.31) formed from ten *expressions*, each expression corresponding to the total impact of all paths having the same length. The environmental impacts of all paths form the *coefficients* of the decision variables. These coefficients are grouped according to the lengths of the paths.

Gams accepts logical conditions in the implementation. This feature allows introducing the main constraints in the objective function as logical expressions.

The expressions in the objective function are evaluated twice at each tier t.

The *first condition* regards the entire expression. The expression is considered or ignored based on the value of s_t parameters: if $s_t = 1$, the expression is considered; if $s_t = 0$, the expression is ignored.

For the expressions which are considered, the *second condition* regards the coefficients of each decision variable, corresponding to the environmental impact of each path from those groups of paths. If the calculated coefficients are above the threshold value, they are considered as part of the objective function; if the calculated coefficients are below the threshold value, the path is ignored and the coefficient are not considered as parts of

$$
\begin{aligned}
\max \quad & s_1 \sum_i f_i y_i b_i \\
& + s_2 \sum_i \sum_j f_j a_{ji} y_i b_{ij} \\
& + s_3 \sum_i \sum_j \sum_k f_k a_{kj} a_{ji} y_i b_{ijk} \\
& + s_4 \sum_i \sum_j \sum_k \sum_l f_l a_{lk} a_{kj} a_{ji} y_i b_{ijkl} \\
& + s_5 \sum_i \sum_j \sum_k \sum_l \sum_m f_m a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklm} \\
& + s_6 \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n f_n a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklmn} \\
& + s_7 \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q f_q a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklmnq} \\
& + s_8 \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r f_r a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklmnqr} \\
& + s_9 \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \sum_t f_t a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklmnqrt} \\
& + s_{10} \sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \sum_t \sum_u f_u a_{ut} a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i b_{ijklmnqrtu} \qquad (3.19)
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.} \quad & s_1 f_i y_i \ge b_i Tol && \forall i \in I && (3.20) \\
& s_2 f_j a_{ji} y_i \ge b_{ij} Tol && \forall i \in I, \forall j \in J && (3.21) \\
& s_3 f_k a_{kj} a_{ji} y_i \ge b_{ijk} Tol && \forall i \in I, \forall j \in J, \forall k \in K && (3.22) \\
& s_4 f_l a_{lk} a_{kj} a_{ji} y_i \ge b_{ijkl} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L && (3.23) \\
& s_5 f_m a_{ml} a_{lk} a_{kj} a_{ji} y \ge b_{ijklm} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, && \\
& && \forall m \in M && (3.24) \\
& s_6 f_n a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i \ge b_{ijklmn} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, && \\
& && \forall n \in N && (3.25) \\
& s_7 f_q a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i \ge b_{ijklmnq} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, && \\
& && \forall n \in N, \forall q \in Q && (3.26) \\
& s_8 f_r a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i \ge b_{ijklmnqr} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, && \\
& && \forall n \in N, \forall q \in Q, \forall r \in R && (3.27) \\
& s_9 f_t a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i \ge b_{ijklmnqrt} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, \forall m \in M, && \\
& && \forall n \in N, \forall q \in Q, \forall r \in R, \forall t \in T && (3.28) \\
& s_{10} f_u a_{ut} a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i \ge b_{ijklmnqrtu} Tol && \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L, && \\
& && \forall m \in M, \forall n \in N, \forall q \in Q, \forall r \in R, \forall t \in T, \forall u \in U && (3.29)
\end{aligned}
$$

$$
b_i, b_{ij}, b_{ijk}, b_{ijkl}, b_{ijlkm}, b_{ijklmn}, b_{ijklmnq}, b_{ijklmnqr}, b_{ijklmnqrt}, b_{ijklmnqrtu} \quad \text{binary} \qquad (3.30)
$$

the objective function.

The model version presented below shows the expressions to evaluate at each tier. The criteria for evaluating the expressions are explicitly presented. However, the order of the evaluation is not as obvious as in Gams implementation.

$$
\begin{aligned}
\max \quad & \underbrace{\sum_i \underbrace{f_i y_i}_{\text{coef. 1st tier>Tol}} b_i}_{\text{expresion 1st tier: } s_1>o}
+ \underbrace{\sum_i \sum_j \underbrace{f_j a_{ji} y_i}_{\text{coef. 2nd tier>Tol}} b_{ij}}_{\text{expression 2nd tier: } s_2>o}
+ \underbrace{\sum_i \sum_j \sum_k \underbrace{f_k a_{kj} a_{ji} y_i}_{\text{coef. 3rd tier>Tol}} b_{ijk}}_{\text{expression 3rd tier: } s_3>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \underbrace{f_l a_{lk} a_{kj} a_{ji} y_{\mathrm{i}}}_{\text{coef. 4th tier>Tol}} b_{ijkl}}_{\text{expression 4th tier: } s_4>o}
+ \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \underbrace{f_m a_{ml} a_{lk} a_{kj} a_{ji} y_i}_{\text{coef. 5th tier>Tol}} b_{ijklm}}_{\text{expression 5th tier: } s_5>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \sum_n f_n \underbrace{a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i}_{\text{coef. 6th tier>Tol}} b_{ijklmn}}_{\text{expression 6th tier: } s_6>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \underbrace{f_q a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i}_{\text{coef. 7th tier>Tol}} b_{ijklmnq}}_{\text{expression 7th tier: } s_7>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \underbrace{f_r a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i}_{\text{coef. 8th tier>Tol}} b_{ijklmnqr}}_{\text{expression 8th tier: } s_8>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \sum_t \underbrace{f_t a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_{it}}_{\text{coef. 9th tier>Tol}} b_{ijklmnqrt}}_{\text{expression 9th tier: } s_9>o} \\
& + \underbrace{\sum_i \sum_j \sum_k \sum_l \sum_m \sum_n \sum_q \sum_r \sum_t \sum_u \underbrace{f_u a_{ut} a_{tr} a_{rq} a_{qn} a_{nm} a_{ml} a_{lk} a_{kj} a_{ji} y_i}_{\text{coef. 10th tier>Tol}} b_{ijklmnqrtu}}_{\text{expression 10th tier: } s_{10}>o} \qquad (3.31)
\end{aligned}
$$

$$\text{s.t.} \quad b_i, b_{ij}, b_{ijk}, b_{ijkl}, b_{ijlkm}, b_{ijklmn}, b_{ijklmnq}, b_{ijklmnqr}, b_{ijklmnqrt}, b_{ijklmnqrtu} \quad \text{binary} \qquad (3.32)$$

This algorithm for generating the objective function can be also explained by using pseudo-code:

```
Algorithm: GENERATE_MODEL
for t=1 to 10
        evaluate expression at tier t
        if $s_t$ = 1
            for all paths until tier t
                  evaluate impact of the path
                  if impact > Tol
                        consider coefficient
                  else
                        do not consider coefficient
        else
            go to next expression
end
return reduced model
```

The second step in solving the problem consists of using the Cplex algorithm to solve the reduced version obtained in the first phase.

```
Algorithm: EXECUTE_MODEL(reduced model)
Cplex
return(solution)
```

The Gams site (http://www.gams.com/dd/docs/solvers/cplex.pdf) offers information about the Cplex algorithm. For problems with integer variables as this one, Cplex uses a branch and cut algorithm which solves a series of linear problems derived from relaxing the integrality constraints. If the solution is integer the problem is solved. Otherwise, further constraints derived from the non-integer solution found are added to the problem, further constraining it. The resulting subproblems are solved with the same procedure, until an integer solution is found or the problem becomes infeasible.

We showed how this problem can be decomposed into subproblems for each tier used. Because the number of paths up to a tier depends on the number of paths up to the previous tier, the output of a problems become the input for the next one. This feature can be exploited by the algorithm to speed up model execution.

The total impact needs to be calculated only for the paths found through the optimization process described above. The impact of each path is calculated with formula 2.9. However, changes need to be made in calculating these parameters. In order to respect the logical structure of the thesis, these changes will be justified and presented in the next subchapter.

3.2 Implementation details

The previous chapter illustrated the necessity of representing the production system/economy as a tree, with individual production layers situated at the same level (figure 2.3). The methodology for calculating the contribution of different processes/sectors at the same level was presented in table 2.2. Formula 2.9 shows how to calculate the environmental impact of a given path.

The number of paths generated for an LCA production system is too large to take them all into consideration. Based on this consideration, we discussed the necessity of 'pruning' the tree generated by the final demand, in order to eliminate some of the paths. The most logical way to do this is to eliminate the paths with very low environmental impact. A threshold is chosen, so that all paths with environmental impact below the threshold are ignored. The tree is reduced if the maximum number of layers is limited.

There are multiple ways of 'pruning' the tree. Subchapter 2.3.1 presented the Matlab-SPA developed by Peters and Hertwich (2006). They incorporate this procedure into their specialized algorithm as a step at each iteration. This thesis comes with a different approach: 'pruning' the tree before applying the generalized algorithm searching for the paths.

Function **presorting.m** is implemented in Matlab and performs a presorting of the elements of the paths (as defined in the previous subchapter) by removing the ones with environmental contribution smaller than a given threshold. It returns the presorted elements to be taken into consideration at each individual tier in the form of a matrix with 10 columns, each column describing the corresponding tier. The lines correspond to each of the elements of the network.

The procedure has 5 steps:
(1) calculate total emissions associated with the final demand using formula 2.4
(2) calculate output matrix per tiers using formula 2.5
(3) scale output per tiers with pollution intensity to get environmental impact per sectors and tiers. Every element represents environmental impact of the sector (direct and indirect) and is calculated using formula 2.9
(4) divide environmental impacts per sectors and tiers by the total emissions to get relative environmental impacts per sectors and tiers
(5) for every sector and tier, if the relative impact per sector and tier is smaller than the threshold (as percentage) the sector is ignored at that tier.

In order to compare the results for different inputs, the threshold is expressed as percentage of total emissions. For comparison reasons, the environmental impacts of the elements at each tier must also be expressed as a percentage of total emission.

To exemplify presorting, we consider the same model of a three sector economy from the previous chapter, this time for a final demand of one unit from sector 1 and with four maximum tiers, instead of 10 tiers.

The tree generated by the final demand up to the 4th tier is presented in figure 3.2. This is the tree before presorting.

The matrix below presents the environmental impact of each sector at each tier, relative to the total impact of the final demand.

$$\begin{array}{r c} & \begin{array}{cccc} \underline{1^{st}\text{ tier}} & \underline{2^{nd}\text{ tier}} & \underline{3^{rd}\text{ tier}} & \underline{4^{th}\text{ tier}} \end{array} \\ \begin{array}{r} \text{1st sector} \\ \text{2nd sector} \\ \text{3rd sector} \end{array} & \begin{bmatrix} 100\% & 2\% & 6\% & 2.5\% \\ 0 & 50.4\% & 3.5\% & 5.2\% \\ 0 & 3.62\% & 7.5\% & 1\% \end{bmatrix} \end{array}$$

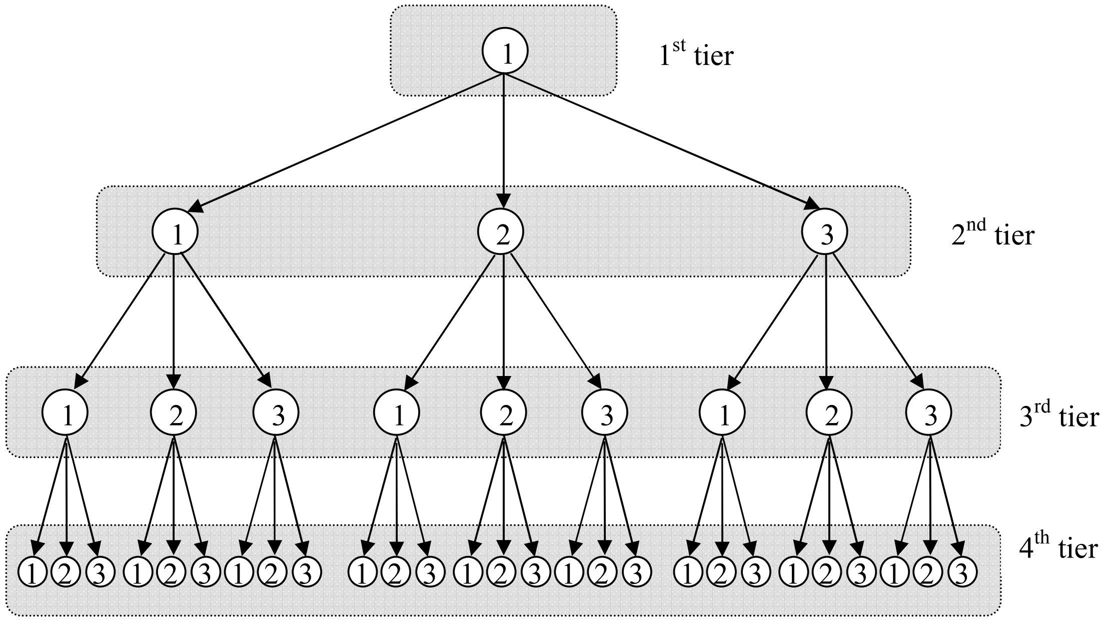


Figure3.2: Tree representation of a three sector economy for a final demand from sector 1

Any value can be chosen as the tolerance, but it needs to be low enough in order to not exclude paths with significant values. In this case $Tol = 3.6\%$ was chosen.

The last step consists of comparing all the elements of the matrix above with the tolerance chosen.
At 1st tier, for the final demand, 100%>3.6%, so this sector at this tier will not be deleted.
At 2nd tier, sector 1 is removed because 2 %< 3.6%. Sectors 2 and 3 are kept, because 50.4%>3.6% and respectively 3.62%>3.6%.
At 3rd tier, sectors 1 and 3 are kept, because 6%>3.6% and respectively 7.5%>3.6%. Sector 2 is deleted because 3.5 %< 3.6%.
At 4th tier, sectors 1 and 3 are deleted, because 2.5 %< 3.6% and respectively 1 %< 3.6%. Sector 2 is kept because 5.2%>3.6%.

The tree after presorting looks like in figure 3.3. Sector 1 was removed from the 2nd tier along with all paths deriving from it. Sector 2 removed at the 3rd tier cancelled all paths using it in the third tier. Because sector 2 is the only sector remaining at the 1st tier, the paths existing at the 3rd tier are extended until the 4th tier.

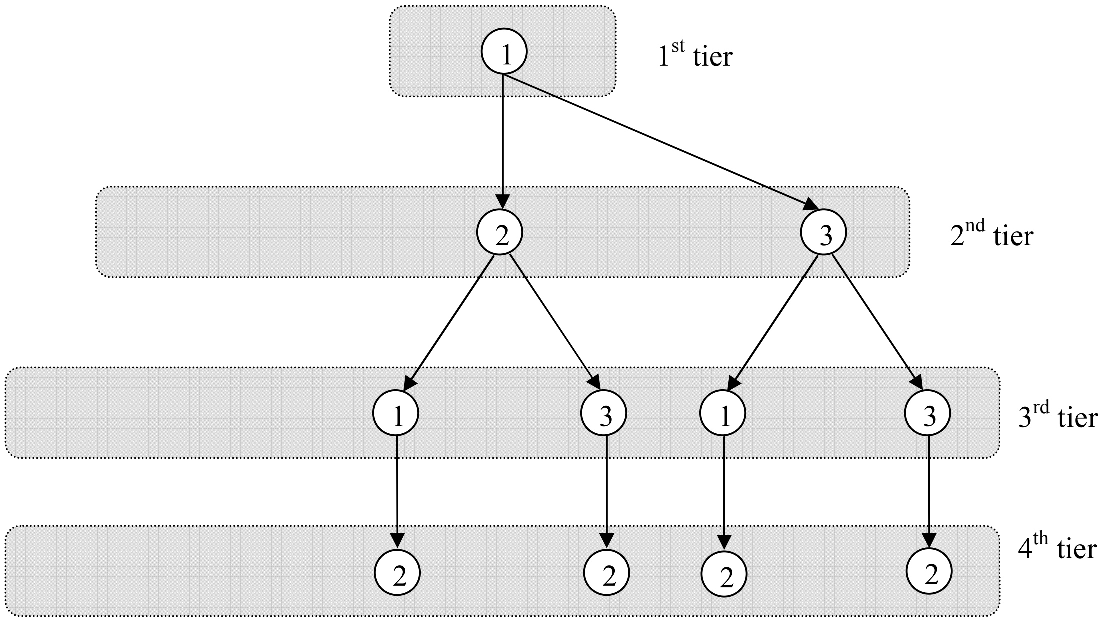


Figure 3.3: Tree representation of a three sector economy for a final demand from sector 1, after presorting

The sectors remaining at each tier after preprocessing are transferred to Gams as sets *I, J,K,L,M,N,Q,R,T,U*. These sets are used in the optimization model presented in subchapter 3.1. This procedure, along with other pre-processing requirements, is explained below.

The main program is **spa.m** and is implemented in Matlab R2008b. It prepares the input data for the optimization procedure in Gams, processes the optimization result and sorts the obtained paths. Each step requires adjustments to be made in order to assure *compatibility between Gams and Matlab*. Both the Gams file and the Matlab file calling for it need to be in the same directory for the call to be effective.

The link between Matlab and Gams is assured by the Matgams package. This software gives Matlab users the ability to use all the optimization capabilities of Gams, and allows visualization of Gams models directly within Matlab. There are two alternative ways of interfacing Matlab and Gams: (1) passing data to and from a Gams program via parameters to a 'gams mex interface' and (2) using two mex procedures to read and write a gdx file which can be interpreted by built-in procedures in Gams. (http://www.cs.wisc.edu/math-prog/matlab.html)

The interface dictates the adjustments to be made to assure compatibility between Gams and Matlab. Here, we used the first alternative of interfacing Matlab and Gams, the 'gams mex interface'. The second alternative was not tested.

The adjustments mentioned above differentiate according to the direction of data transfer between Matlab and Gams. We will consider first adjustments triggered by data transfer from Matlab to Gams, then those triggered by transfers from Gams to Matlab.

1. Adjustments triggered by data transfer from Matlab to Gams

Four types of data are transferred from Matlab to Gams: one set, identified as *sect*; ten subsets, identified as *i, j, k, l, m, n, q, r, t, u*; parameters defined using set and subsets: *A, F* and *y*; independent parameters: *Tol* and s_t coefficients.

Sect is the set containing the names of all sector/processes from the economic network. This data is loaded from the input file as a cell array, each cell containing the name of a sector, formatted as string.

Adjustment 1: indexing set elements
The names loaded in this form contained comma, dots, etc which are not allowed by Gams syntax. This is why the names constituting the set elements have been replaced by indices. Suitable default indices in this situation are 1, 2, 3…*m*, where *m* in the total number of elements. This conversion will have to be reversed after solution is sent back to Matlab.

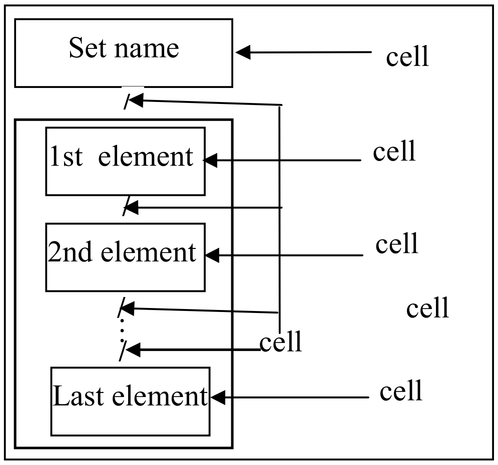
The set *sect* has 10 subsets: *i, j, k, l, m, n, q, r, t, u*. Elements of these subsets are the sectors/processes identified in the presorting phase. The order of subsets corresponds to the tier order. Each subset contains the sectors/processes at the corresponding tier. Example: subset *k(sect)* contains the sector/processes from the 4th tier. The specific syntax shows that *k* is a subset of set *sect*.

Adjustment 2: treating unused subsets
We showed in the previous chapter that sometimes all sectors/processes are removed from ending tier(s) as a result of presorting. The subsets corresponding to this tier(s) are supposed to be empty. But Gams does not accept empty sets/subsets in the chosen data transfer type. Even if these subsets will not be used to create paths, they are not allowed to remain empty. First element of set *sect* is arbitrarily chosen as the unique element of these sets.

The sets and subsets will be saved in a file *set.gms* during compilation time and read as input by Gams.

Adjustment 3: assembling sets in a table
One of the main requirements of this program is to transfer input data and to recover the solution in Matlab in one system call. The Matgams software used to transfer the set and the subsets from Matlab to Gams offers alternatives, but they can not fulfill the above

requirement. The solution found is to assemble all sets and subsets in a table which is written in Gams during the system call. This table has the same structure as the *set.gms* file generated by typing the sets directly in the Gams file.



Every set has two components: name and content, separated by a cell containing the "/" sign.

Both name and content need to be of cell type.

The content is a cell array with elements of type cell, placed on rows and separated by cells containing the "/"sign.

Figure 3.4: Set structure in Matlab used to create set.gms

In the set.gms file, all sets are placed on rows and are separated by "/". The assembled table has only one column. The number of rows is given by cells containing set names, plus cells containing set elements, plus all cells containing the separation character "/".

In transferring values between Gams and Matlab, the basic object is a structure. A structure has three fields: (1) *name*, holding the Gams name of the object; (2) *val*, used for values; (3) *labels*, fields of the declared parameters.

Adjustment 4: defining parameters as structures

Parameter A has two dimensions, each defined as sectors/processes. Parameters F and y have only one dimension, the same sectors/processes used for parameter A. Structures are defined for each of these parameters for transferring from Matlab to Gams. The *name* fields in each structure need to correspond with the name used in Gams for the parameter transferred by the structure.

The label field in a structure is a cell array. The size of this cell array is 1 x p, where p is the number of dimensions of the *val* field. Each entry in the cell array is another cell array of size 1 x d, where d is the dimension of the corresponding dimension of the *val* field. This second cell array holds the strings that will be used as *labels* for the *val* parameter.

According to the above presentation, each value of the A parameter is identified by two labels, each from "sect" set; each value of the F and y parameters is identified by one label, from "sect" set. In this particular case, when the set elements are indexed and all elements of the set are used as *labels*, the indices in the original representation from Matlab correspond with the *labels* in Gams.

Parameters *Tol* and s_t are read as scalars in Gams. They are not transferred as structures from Matlab. The name of this type of parameters needs to be the same in Matlab and in Gams. The value from Matlab is read in Gams directly in *val* field.

All parameters are written in the matdata.gdx file. Another file, matdata.gms, updates the Gams file by reading the new values of the parameter from matdata.gdx file. Thus, if the file matdata.gms does not exist, the original data from matdata.gdx file is used. Otherwise, the values of the parameters are killed at Gams compilation time and replaced by their new values.

2. Adjustments triggered by data transfer from Gams to Matlab

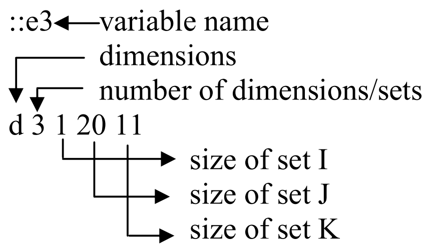
The paths are transferred from Gams to Matlab using the batch utility “matout”. This utility changes the output format; adjustments need to be done in order to retrieve paths in Matlab in a useful format.

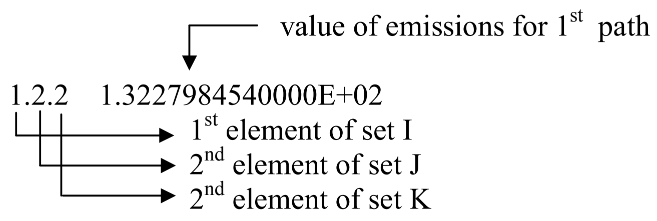
In Gams, the paths are interpreted as tuples. Theoretically, a tuple in Gams is a subset of some sets which encapsulates some conditionals regarding the elements of the sets it is based on. In this case, the elements of the path correspond to elements from subsets *I, J, K* ... and are connected together by the value of the path (see subchapter 3.1).

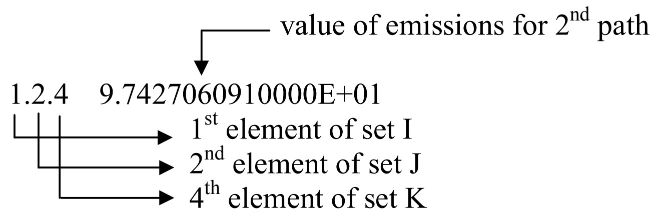
The format of the output changes in the transfer process to a file that can be read back into Matlab. That file is called “matsol”. Previously, we mentioned that the basic format for transferring data is the structure. According to this, in the “matsol” file, a path is converted to a structure with three fields: a *name*, a *value* and a *label*. For each group of paths, defined as in subchapter 3.1, the name is common for all paths in the group, but the value and labels are different.

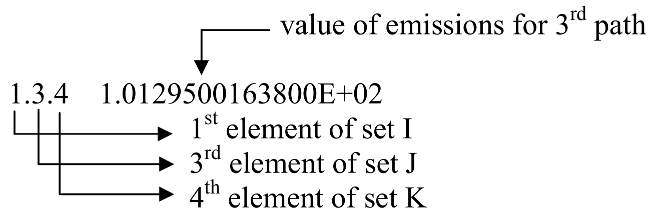
For each group of paths, the “matsol” file lists the *name* of the group, the number of dimensions for the paths, and the size of each subset used to define the paths. Then, for each path found as a result of the optimization process, “matsol” lists first a cell array of the indices, then the value of the path. The indices return the order of the tuple elements in the subsets mentioned above.

Example: e3(i,j,k) is a tuple, defined on sets I, J and K so that one element from set I, one element from set J and one element of set K exist simultaneously. The path created by the elements of the tuple is denoted by i-j-k. The output in “matsol” is presented below for a list of three paths from this group, paths obtained in the optimization process.


```
::e3
d 3 1 20 11
```


```
1.2.2    1.3227984540000E+02
```


```
1.2.4    9.7427060910000E+01
```

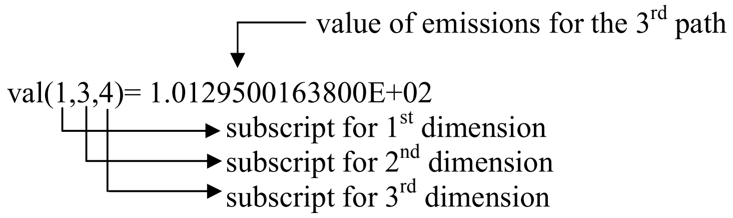

```
1.3.4    1.0129500163800E+02
```

The output from "matsol" for each group of paths with n dimensions is read in Matlab as a n-dimensional matrix. The values of the paths are elements in the n-dimensional matrix. The array of indices from the "matsol" file becomes the subscripts of these elements.

Example: the output in Matlab for the previous example is a 3-dimensional 1x20x11 matrix of type double in which every element can be accessed by using subscripts. Each subscript indexes a dimension. The value of emissions for a path corresponds to an element of the matrix. The subscripts of this element are derived from the array of indices corresponding to the path in the "matsol" file.

val(1,2,2)= 1.3227984540000E+02

val(1,2,4)= 9.7427060910000E+01


value of emissions for the 3rd path
val(1,3,4)= 1.0129500163800E+02
subscript for 1st dimension
subscript for 2nd dimension
subscript for 3rd dimension

Adjustment 5: recovering paths from the output representation in Matlab

Function **getpath.m** recovers the paths (value and elements) from the multidimensional representation. For each group of paths, it first identifies the subscripts for each path, and then uses the subscripts to recover the indexed element names.

If the nth dimension of a n-dimensional path has only one element, that path is not read correct in Matlab from the "matsol" file. The last index from the list of indices characterizing the path is lost in the conversion to Matlab's multidimensional representation. If the last dimension in a n-dimensional representation has only one element, Matlab ignores this dimension by default. The function script is adapted to take this situation into account by adding the missing subscript to the list of subscripts obtained from the dimensions recovered directly.

Adjustment 6: reconvert the indexed element names to the real names

Function **sorting.m** reconverts the numerical name indices to the real sector/process name for every found path. Foe example, names like "1","2","3" are converted to "agriculture", "manufacturing", "transport". The function also sorts the paths in descending order of impact value.

The path structure has three fields: (1) *name,* of type string, (2) *impact value*, of type double and (3) *labels* – the elements of the path –, of type cell. The label cell contains a cell array, each entry in the cell array being another cell containing the name of the sector/process. Figure 3.5 presents 3 paths from group $i \to j \to k$ (identified as 1, 2, and 3) stressing on the type of component elements.

Only one command is used to execute the Gams file with a system call. The command feeds the inputs and collects the outputs from Gams. The number of output variables in this command needs to correspond with the number of output parameters from Gams.

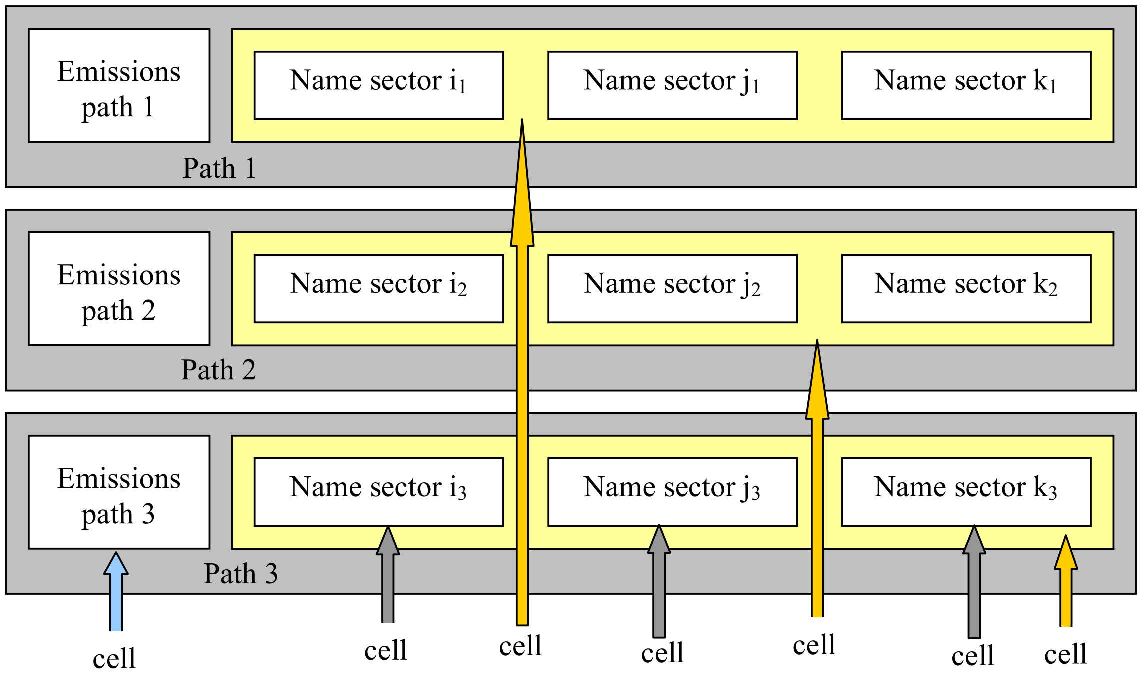


Figure 3.5 Paths and the type of component elements

The number of paths fulfilling the optimization criteria is unknown. Even if no path from a certain group is found, we still need to have an output for Matlab, for the reason explained above. Obviously, the impact of all paths from this group will be zero. The “matsol” file will record these paths as follows:

::e3
number of dimensions/sets
d 3 1 29 18
size of set i
size of set j
size of set k

Matlab reads this data as a matrix of three dimensions with one element on the 1st dimension, 29 elements on the 2nd dimension and 18 elements on the 3rd dimension. All elements of this matrix are zero. In fact, Matlab allocates memory for a matrix which will not be used at all, because none of the possible paths fulfills the optimization criteria.

For 10-dimensional matrices the amount of memory allocated to read unused paths is very high. We already mentioned in this chapter that memory is an issue for this model and adjustments have been made in the optimization program to reduce it (see s_t parameters).

Introducing s_t parameters we reduced the necessary memory for the execution of the optimization program. This time adjustments need to be made in order to reduce the amount of memory used by Matlab to interpret the zero-value paths found by the Gams program.

There are two things we need to take into account for identifying the file in which the changes need to be made:

1. The "matout" file controls the Gams output in the "matsol" file. Since we don't want to edit the "matout" file, there is nothing we can do to change this output, at least not for this optimization version of the model.

2. We cannot change the way data is transferred between the "matsol" file and Matlab

The most obvious choice is to make some post-optimization changes in Gams file. These changes need to adjust the way Gams calculates the zero-value paths before transferring them to the "matsol" file.

These zero-value paths have three fields: *name*, *value* and *dimension.* Among these fields, the *dimension* field alone creates the memory problem. In Gams, multidimensional parameters can be returned by appending the appropriate indexing sets after the name of the parameter. If the indexing sets are not appended after the name of the parameter, just one dimensional value is transferred to the "matsol" file and from there it is read by Matlab as a one-dimensional parameter. Logically, we want the parameters with zero in the value field to be transferred to the "matsol" file without the indexing sets.

Example: e5 is transferred as a one-dimension parameter in the first expression below and as a five-dimension parameter in the second expression.

```
$libinclude matout e5
$libinclude matout e5 i j k l m
```

The first expression is coded for the eventuality that no path is found in the optimization process. The second expression is coded for the eventuality that at least one path is found in the optimization process. These two cases are exclusive, but we can find the case triggered by a specific input through a simple check: if no path fulfills the optimization criteria in a group of paths, they all have value zero and their total impact is zero; if at least one path fulfills the optimization criteria, the total impact for all paths in the group is larger than zero. This endogenous test can be introduced in the Gams file, after the optimization procedure, in order to identify which one of the above two coding cases to use.

We want to introduce an "if" condition to employ one coding case or the other according to the value of the total impact. Unfortunately, Gams does not accept both expressions at compilation.

Adjustment 7: calculating alternative values for groups of paths with 0 value each

The solution resides in manipulating the name of the group of paths of zero-value each. In transferring data from Gams to Matlab, the name of the structures defined for the groups of paths can be discarded. Matlab receives only a matrix with values, which can be identified by any name in the output elements from the expression calling Gams. The structure name can be removed by a Matlab command which sets the Gams output to '*std*'.

The algorithm preparing data for transfer from Gams to Matlab is presented below:

```
Algorithm: CALCULATE_EMISSIONS (path group t)
calculate total paths impact of all paths in the group
if  impact = 0
        f = impact
        include f without indexing sets
else
        e = impact
        include e with indexing sets
end
return(one-dimensional f or multi-dimensional e)
```

So, a group of paths in which all paths have zero value are transferred to the "matsol" file as a one-dimensional parameter named *f* of zero value:

```
::f4
d 1 1
1    0.0000000000000E+00
```

With these alternative parameters, only the minimum amount of physical memory is used to record them in Matlab.

All these adjustments are specifically done to assure compatibility between Gams and Matlab, to increase the functionality of the model and to deal with the memory and time issues raised by running these two programs. Next chapter will use real data to show how the main program works.

Chapter 4

Analysis

This chapter presents three case studies used to test the functionality of the Gams-SPA program. The technical performance of the model will be assessed by benchmarking the results of Gams-SPA against the results of Matlab-SPA for the three cases mentioned above. The features included in the model will be justified by comparing the results of this program version with older versions.

4.1 Case studies

SPA can be conducted referring to a stand-alone product or to an output of an industry.

For a *product,* SPA identifies the most polluting products and processes used at some stage in its production. A unit of service can also be a product. All products and processes used as inputs at any production stage are the elements of the product network, called LCA type network.

For each production process we need to know the amount of all direct inputs (products and processes) required. This data has to be assembled in a square matrix formed by the elements of the production network. This is the *inter-process requirements matrix*. Each element a_{ij} is measured as the amount of input i required per unit of input j. Matrix data can be obtained directly from the production process or can be taken from already-compiled databases. The number of elements included depends on data availability.

This SPA is designed to evaluate only one type of environmental impact at a time. We need to know the direct environmental impact of one unit of each element. This *stressor* data is presented as a line *vector*, indexed by elements of the network. Data is measured as stressor units per products and process. It can be obtained from direct measurements, but usually is taken from databases.

This SPA studies one product at a time. We need to construct a column vector indexed by the elements of the network. All its values will be 0, with one exception: for the studied element (product or service) it takes value 1. This is called the vector *of external demand of processes.*

SPA is intended to find the paths describing the process relationships that link the final delivered product with the emitting processes. A path is returned as a link of network elements starting from the product or service chosen (left) and ending with the polluting sector (right). The ending polluting sector requires inputs from the previous element which on its turn requires inputs from the element placed in front of it and so on, until the final delivered product is reached.

Similarly to the product network, we define the economic network, also named the IO network, in which all elements are *sectors*. SPA is conducted for a specific output of a sector. Sector output can be evaluated in physical units or in monetary units. The matrix of all sectors is called the *inter-industry coefficient matrix*. Each element a_{ij} is measured as the amount of sector i required per unit of stressor j. Stressor data is represented as in the description of the product network, using sectors instead of products. The vector used to point the sector for which the SPA is conducted is now called *the final demand vector.*

Usually, statistic bureaus from any country gather and calculate the data needed, in monetary units. Because of the inconsistency in reporting, data require manipulation to make the complete data set consistent. For data sets including more than an economy, even more manipulation of data is necessary. This procedure introduces a lot of errors in the data set. The errors can result from aggregation, allocation of trade data, mapping a new classification system, exchange rates, inflation measures, etc.(Peters and Hertwich, 2006). This is the biggest source of uncertainty in the model and the results have to be treated accordingly.

Three data sets of increased complexity have been used to test the program. In all cases data has been provided as .mat files, containing the matrix of products or sectors, the stressor vector, the external demand/final demand vector, the identity matrix and a *vector containing the name of the processes/sectors* used to define the above elements. Using this format is not compulsory, but it was employed here so that the same input file could be used for testing both Gams-SPA and Matlab-SPA. Another five variables have been added to accommodate Matlab-SPA, but our program deletes them immediately after loading. As an alternative, the variables needed only for this program can be loaded individually.

Test cases have been developed based on three data sets of increasing complexity. They are presented below. Data was recorded at the second iteration when all parameters, except threshold value, were already saved in matdata.gms.

Case 1 is based on a 10-sector model of the Norwegian economy. The stressor chosen for calculating impacts is CO_2. Data for the inter-industry coefficient matrix and vector of stressors is provided by the Industrial Ecology program. The input file is NorIO.mat. The final demand was chosen as one unit from the manufacturing sector. SPA identifies the paths connecting the manufacturing sectors with the most polluting sectors at different stages in the economic network. If we ignore all paths with the impact below 0.5% of total emissions, 17 paths remain. These paths cover 86.1% of total emissions. They are listed in Table 4.1.

Because all sectors have been aggregated in 10 big groups, the resolution of the information in the results is rather low. However, we see that 43.6% of the total CO_2 emissions (0.0627 kg CO_2) generated by one unit of manufacturing occur in the first tier. Manufacturing and transportation are by far the most polluting sectors, creating 16 paths from 22 paths found. Agriculture and mining have relatively small contributions to the total CO_2 impact in the second, third and fourth tier, approximately 2.5% both. Rest of the sectors have a contribution below 0.5% and are ignored.

Table 4.1 All paths above a threshold value of 0.5% for CO2 emissions starting from one unit of manufacturing (left) and ending with the polluting sector (right); The 17 paths represent 86.1% of total emissions

Rank	***Contribution (%)***	***Path***
1	43.60	Manufc
2	14.85	Manufc→ Manufc
3	7.09	Manufc→ Transp
4	5.06	Manufc→ Manufc→ Manufc
5	2.41	Manufc→ Manufc→Transp
6	2.29	Manufc→ Transp→ Transp
7	1.75	Manufc→ Agric
8	1.73	Manufc→ Mining
9	1.72	Manufc→ Manufc→ Manufc→ Manufc
10	0.92	Manufc→ Agric→ Manufc
11	0.82	Manufc→ Manufc→ Manufc→Transp
12	0.78	Manufc→ Manufc→ Transp→ Transp
13	0.74	Manufc→Transp→ Transp→ Transp
14	0.60	Manufc→ Manufc→ Agric
15	0.59	Manufc→ Manufc→ Mining
16	0.59	Manufc→ Manufc→ Manufc→ Manufc→ Manufc
17	0.56	Manufc→ Constr

Note: The sector abbreviations are given in Appendix F.

Case 2 is based on a 481-sector of the US economy for 1997/1998. The stressor chosen for calculating impacts is CO_2. Data is provided by the Industrial Ecology program. The input file is oil_drill.mat. A "petroleum and natural gas well drilling" service was chosen as final demand here. The service unit was arbitrarily established at $25000 worth of drilling. SPA identifies the paths connecting the above-mentioned unit of drilling service with the most polluting sectors at different stages in the economic network.

Total impact is 19848 Kg of CO_2 for a threshold of 0.1%. The first 44 paths with impact above 0.1% from the total emission value are listed in Table 4.2. More than half of the emissions (54.89%) are direct emissions from the drilling in the first tier. Sectors: "blast furnaces and steel mills", "cement, hydraulic" and "electric services" dominate first tiers. Chemicals and minerals used in drilling have a considerable influence in the total emissions, along with equipments and transportation.

This specific example was chosen because it employs a particularity of some of the paths: the contribution of a sector is smaller than the contribution of its input sectors: The 30th path: Drilling→Motors→**Stamping**→Steel mills has a contribution of 0.17% of total impact, but Drilling→Motors→**Stamping** has been ignored, because its contribution is below 0.1%.

Table 4.2 All paths above a threshold value of 0.1% for CO2 emissions starting from a drilling service (left) and ending with the polluting sector (right); The 44 paths represent 78.25% of total emissions

Rank	***Contribution (%)***	***Path***
1	54.89	Drilling
2	4.32	Drilling→ Steel mills
3	3.36	Drilling→ Cement
4	2.73	Drilling→ Electricity
5	1.10	Drilling→ Chemicals
6	1.05	Drilling→ Refining
7	0.67	Drilling→ Steel mills→Steel mills
8	0.67	Drilling→ Metal parts→Steel mills
9	0.54	Drilling→ Chemicals→Electricity
10	0.54	Drilling→ Minerals
11	0.52	Drilling→ Land T.
12	0.51	Drilling→ Cement→Electricity
13	0.50	Drilling→ Equipm. manuf
14	0.49	Drilling→Steel mills→Electricity
15	0.46	Drilling→ Air T.
16	0.44	Drilling→ Drilling
17	0.44	Drilling→Equipm. manuf→Steel mills
18	0.39	Drilling→Motors
19	0.36	Drilling→Rail T.
20	0.33	Drilling→Lime
21	0.28	Drilling→Trade→Electricity
22	0.27	Drilling→Oil_reused
23	0.25	Drilling→Metal parts
24	0.24	Drilling→Chemicals→Chemicals
25	0.22	Drilling→Oil_reused→Refining
26	0.19	Drilling→Equipment→Steel Mills
27	0.19	Drilling→Trade
28	0.19	Drilling→Equipm. manuf→Electricity
29	0.18	Drilling→Chem. Prep
30	0.17	Drilling→Motors→Stamping→Steel mills
31	0.16	Drilling→Equipment
32	0.16	Drilling→Steel Mills→Raw mat.
33	0.15	Drilling→Motors→Motors parts
34	0.14	Drilling→Water T.
35	0.13	Drilling→Steel parts
36	0.13	Drilling→Equipm. manuf→Metals
37	0.13	Drilling→Refining→Raw oil
38	0.12	Drilling→Cement→Cement
39	0.12	Drilling→Chemicals→Chemicals→Electricity
40	0.11	Drilling→Equipm. parts
41	0.11	Drilling→Motors→Motors parts→Steel mills
42	0.11	Drilling→Minerals→Electricity
43	0.10	Drilling→Steel mills→Steel Mills→Steel mills
44	0.10	Drilling→Metal parts→Steel Mills→Steel mills

Note: The sector abbreviations are given in Appendix F.

Case 3 is build upon LCA data set. Data, provided by Industrial Ecology program, is compiled from Ecoinvent database, for both the inter-process requirement matrix and the stressor matrix. The original data contained 2549 processes and 4345 stressors. The external demand chosen here is one transoceanic tanker (the 2333rd process in enumeration). This product has only two direct stressors: heat waste and NMVOC (non-methane volatile organic compounds, unspecified origin). The second was chosen for testing. From the 4345x2549 stressors matrix, only the line corresponding to the NMVOC stressor was chosen (1234th line).

Gams requires scaling of coefficients, so that their magnitude is centered on one and the ratio between the highest value and the lowest one ranges between 0.01 and 100. This requirement is detailed in the document mccarlgamsuserguide.pdf, available from the site www.gams.com.

No stressor was found to comply with this requirement. From the 2549 processes recorded in the database, 530 processes have positive impacts regarding this stressor. The values range between 1.39E-10 kg NMVOC per one unit from process: "Treatment, pig iron production effluent, to wastewater treatment, class 3/CH U" and 237000 kg NMVOC per one unit of process "operation, maintenance airport/RER U". Only nineteen processes have impact between 1 kg and 237000 kg NMVOC per process unit.

These values are far beyond the range required by Gams. Adjustments needed to be done in order to be able to test the model on LCA database. All 511 values below 1 have been zeroed in the stressor vector. The remaining values, between 1kg/unit and 237000 kg/unit have been reduced 1000 times. This scaling assures that Gams can handle the data input. The input file obtained by making these changes is tanker1.mat.

The aim of this thesis is not to perform a SPA for the transoceanic tanker, but to develop a functional model. That is why we traded the solution for the overall model.

Table 4.3 lists the paths for a threshold value of 0.01%.

Table 4.3 All paths above a threshold value of 0.01% for NMVOC emissions starting from a transoceanic tanker (left) and ending with the polluting sector (right); The 44 paths represent 99.84% of total emissions

Rank	*Contribution (%)*	*Path*
1	99.5	Tanker
2	0.09%	Tanker→ Reinforcing steel→ Steel,converter→Pig iron→Sinter,iron→Transport tanker→Maintenance tanker
3	0.07%	Tanker→ Reinforcing steel→ Steel,electric→Transp. lorry 32 t →Lorry 40t
4	0.043%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron→Sinter,iron→Transport barge→Maintenance barge
5	0.042%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron →Transport freight ship→Maintenance freight ship
6	0.024%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron→Sinter,iron→ Transport freight ship→ freight ship
7	0.022%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron →Transport barge→Maintenance barge
8	0.021%	Tanker→ Reinforcing steel→ Lorry 40t
9	0.019%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron→Coal→ Transport freight ship→Maintenance freight ship
10	0.012%	Tanker→ Reinforcing steel→ Steel,converter→ Pig iron → Transport freight ship→Maintenance freight ship

The total impact is 13.267 kg NMVOC and in this particular case, almost all of it is concentrated in the first tier. Very reduced impacts of NMVOC, compared with the direct impact of tanker, are associated with maintenance of different types of ships used to transport the iron used for constructing the tanker. Lorry transportation has the same importance as sea transportation, as obvious from the third and eighth path.

4.2 Technical performance

This chapter presents the technical performance of the Gams SPA program in terms of time used for compilation and execution, versus the threshold used in the presorting phase. Times obtained in Gams-SPA are benchmarked against times obtained by running Matlab-SPA developed by Peters and Hertwich (2006) on the same data files, for each individual threshold value.

Gams SPA require only two inputs: the file to test and the threshold to cut the paths.

Case 1 results are listed in Table 4.4 and plotted in figure 4.1 below.

Tiers needed	Threshold (%)	Gams SPA	Matlab SPA	Number of paths	Emissions covered
1 tier	43	0.4	0.05	1	43.60%
	35	0.4	0.05	1	43.60%
2 tiers	34	0.4	0.05	1	43.60%
	15.25	0.4	0.05	1	43.60%
3 tiers	15.26	0.4	0.05	1	43.60%
	7.11	0.4	0.05	2	58.46%
4 tiers	7.1	0.4	0.05	2	58.46%
	3.4	0.4	0.05	4	70.60%
5 tiers	3.39	0.4	0.05	4	70.60%
	1.65	0.4	0.05	9	80.52%
6 tiers	1.64	0.5	0.05	9	80.52%
	0.84	0.5	0.05	10	81.43%
7 tiers	0.83	0.5	0.05	10	81.43%
	0.6	0.5	0.05	13	83.78%
	0.43	0.5	0.05	17	86.11%
8 tiers	0.42	0.5	0.05	17	86.11%
	0.3	0.5	0.05	21	87.35%
	0.21	0.5	0.05	25	88.39%
9 tiers	0.2	0.5	0.05	28	88.99%
	0.15	0.7	0.05	35	90.19%
	0.11	1	0.05	37	90.50%
10 tiers	0.1	1.1	0.05	43	91.09%
	0.09	2.2	0.05	47	91.47%
	0.08	2.2	0.05	49	91.63%
	0.07	3.1	0.06	50	91.71%
	0.06	3.5	0.07	57	92.18%
	0.053	4.8	0.08	67	92.75%
Over 10 tiers	0.052	4.8	0.08	69	92.85%
	0.04	8.7	0.09	78	93.30%
	0.01	141.2	0.2	202	95.73%
	0.005	376.2	0.4	333	96.64%
	0.001	X	1.8	974	97.98%

Table 4.4 Time versus threshold comparison between Matlab SPA and Gams SPA in case 1, number of paths and covered emissions for each threshold value.

The size of the problem increases when the threshold is reduced, affecting the time necessary to obtain the results. We see how a lower threshold determines higher execution time for both Gams-SPA and Matlab-SPA versions. The times obtained by running Matlab-SPA are consistently lower for all threshold values tested. Matlab-SPA times increase slow at the beginning but at high tiers we see a very strong increase: from 8.7 seconds for a tolerance of 0.04% to 141.2 seconds for a tolerance of 0.01% and 376.2 seconds for a tolerance of 0.005%. This increase corresponds to the increase in the number of paths found. The value of the paths is smaller and smaller. The total emissions covered by the paths found increases fast at small tiers and much slower at higher tiers. The 124 new paths found by reducing the threshold from 0.04% to 0.01% cover only 2.43% of total emissions.

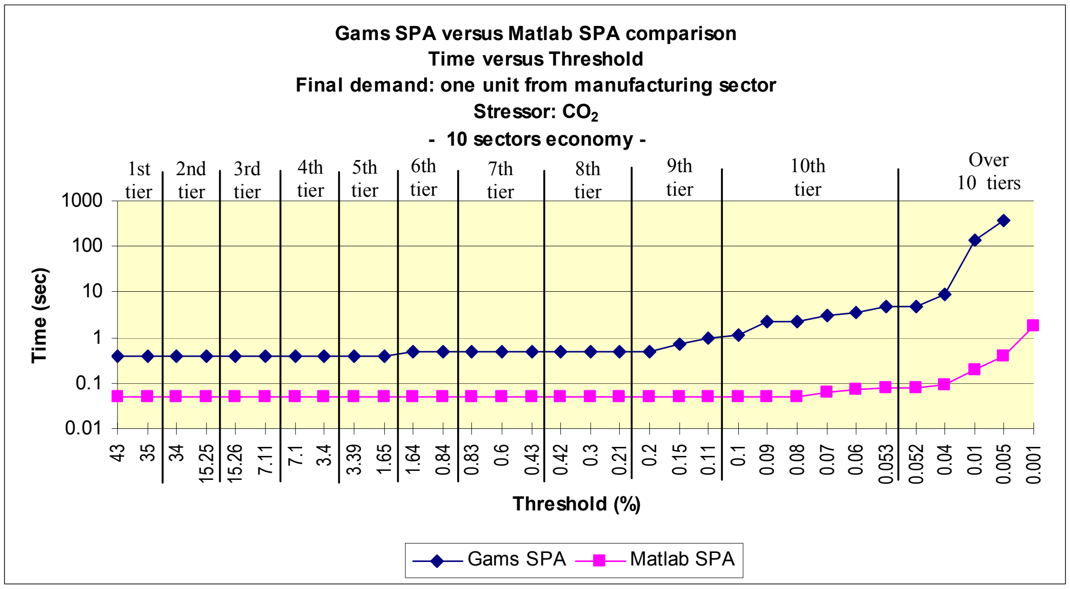


Figure 4.1 Time versus Threshold comparison for Matlab SPA and Gams SPA in case 1

In order to test the system at such reduced tolerance, more than 10 tiers are necessary. This requirement was explicitly presented here, but the program cut off the paths after 10 tiers. For all threshold value tested, no path has been found to extend until the 10th tier. The longest path found for a tolerance of 0.005% extends until the 9th tier.

No values have been obtained by running Gams SPA at threshold 0.001 %. If we look at the time trend we observe an exponential increase for very low threshold values. The size of the problem increases so much that the memory required to generate the equations and to solve them runs out.

Case 2 was tested on a 481-sector data set, which is bigger than the previous one. The running times obtained for both Matlab-SPA and Gams-SPA are presented in Table 4.5. The trend is visible in figure 4.2: until the end of the 8th tier, the running times for Gams-SPA are close but still higher than Matlab-SPA's running times. The increase is very strong at the 9th and 10th tiers: from 35.9 seconds for a threshold of 0.1% to 2105 seconds for a threshold of 0.06%. As in case 1, the total emissions covered by the paths found increase fast at small tiers and much slower at higher tiers. The 16 new paths found by reducing the threshold from 0.1% to 0.074% cover only 1.33% of total emissions.

Tiers needed	Threshold (%)	Gams SPA	Matlab SPA	Number of paths	Emissions covered
1 tier	54.8	1	0.1	1	54.89%
	7.37	1	0.1	1	54.89%
2 tiers	7.36	1	0.1	1	54.89%
	4.47	1	0.1	1	54.89%
3 tiers	4.46	1	0.1	1	54.89%
	3	1	0.1	3	62.56%
	2.46	1	0.1	4	65.30%
4 tiers	2.45	1	0.1	4	65.30%
	1.7	1	0.1	4	65.30%
	1.32	1	0.1	4	65.30%
5 tiers	1.31	1	0.12	4	65.30%
	1	1	0.15	6	67.45%
	0.74	1	0.2	6	67.45%
6 tiers	0.73	1.1	0.2	6	67.45%
	0.67	1.1	0.3	6	67.45%
	0.6	1.1	0.4	8	68.77%
	0.5	1.1	0.5	13	71.38%
	0.43	1.1	0.6	17	73.22%
	0.412	1.1	0.6	17	73.22%
7 tiers	0.411	1.1	0.6	17	73.22%
	0.38	1.1	0.6	18	73.61%
	0.34	1.1	0.6	19	73.97%
	0.3	1.2	0.6	20	74.30%
	0.28	1.2	0.65	21	74.58%
	0.27	1.2	0.7	22	74.86%
	0.26	1.3	0.7	22	74.86%
	0.232	1.4	0.7	24	75.34%
8 tiers	0.231	1.4	0.7	24	75.34%
	0.23	1.5	0.7	24	75.34%
	0.22	1.5	0.7	24	75.34%
	0.21	1.6	0.8	25	75.56%
	0.2	1.6	0.9	25	75.56%
	0.19	1.8	1	26	75.75%
	0.18	2	1.1	28	76.13%
	0.17	2.2	1.1	29	76.30%
	0.16	2.7	1.5	30	76.47%
	0.15	3.6	1.5	33	76.94%
	0.14	7.4	1.7	34	77.09%
	0.1304	10.9	1.8	36	77.35%
9 tiers	0.1303	13.5	1.8	36	77.35%
	0.1	35.9	2	44	78.26%
	0.074	551	2.6	60	79.59%
10 tiers	0.073	701	2.8	60	79.59%
	0.06	2105	4	68	80.13%
	0.01	X	16	307	84.82%
	0.005	X	30	492	86.08%
	0.001	X	120	1940	89.11%

Table 4.5 Time versus threshold comparison between Matlab SPA and Gams SPA in case 2, number of paths and covered emissions for each threshold value.

The memory required to run Gams-SPA for thresholds above 0.01% exceeded the capacity of the system. At those tiers, Matlab-SPA also records a rapid increase in the time used to found the paths. The rapid increase in the number of paths found by Matlab-SPA (until 1940 paths at threshold 0.001%) is a sign of how fast the model grows at those tiers.

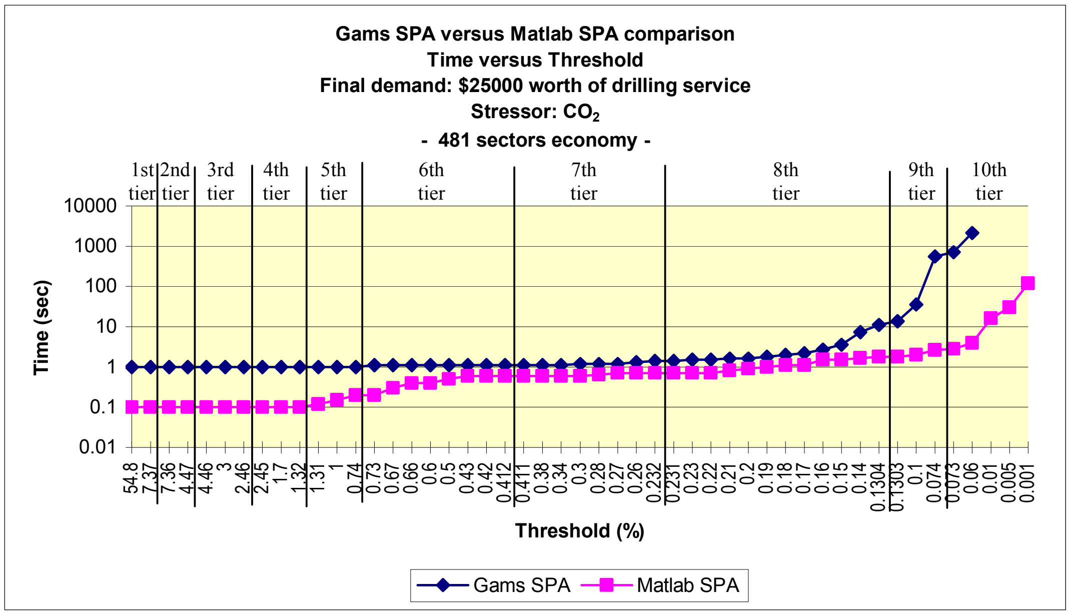


Figure 4.2 Time versus Threshold comparison for Matlab SPA and Gams SPA in case 2

<u>**Case 3**</u> brings a change in situation. The Gams-SPA and Matlab-SPA running results obtained are presented in table 4.6 and plotted in figure 4.3.

Tiers needed	Threshold	Gams SPA	Matlab SPA	Number of paths	Emissions covered
1 tier	99.4	0.6	0.4	1	99.49%
	0.478	0.6	0.5	1	99.49%
2 tiers	0.477	0.6	0.5	1	99.49%
	0.4	0.6	0.5	1	99.49%
	0.315	0.6	0.5	1	99.49%
3 tiers	0.314	0.6	0.5	1	99.49%
	0.3	0.6	0.5	1	99.49%
	0.296	0.6	0.5	1	99.49%
4 tiers	0.295	0.6	0.5	1	99.49%
	0.24	0.6	0.5	1	99.49%
	0.177	0.6	0.5	1	99.49%
5 tiers	0.176	0.6	0.5	1	99.49%
	0.16	0.6	0.5	1	99.49%
	0.142	0.6	0.5	1	99.49%
6 tiers	0.141	0.6	0.5	1	99.49%
	0.12	0.6	0.5	1	99.49%

	0.11	0.6	0.5	1	99.49%
7 tiers	0.1	0.6	0.5	1	99.49%
	0.09	0.6	0.8	1	99.49%
	0.08	0.6	1.7	2	99.58%
	0.07	0.6	2.2	3	99.65%
	0.06	0.6	2.1	3	99.65%
	0.05	0.6	2.4	3	99.65%
	0.04	0.6	3.0	5	99.74%
	0.03	0.6	3.5	5	99.74%
	0.02	0.6	6.0	8	99.58%
	0.015	0.6	6.5	9	99.83%
8 tiers	0.014	0.6	6.5	9	99.83%
	0.012	0.6	7.2	9	99.83%
9 tiers	0.011	0.6	7.2	10	99.84%
	0.01	0.6	7.2	10	99.84%
10 tiers	0.009	0.6	7.2	10	99.84%
	0.0093	0.7	7.4	10	99.84%
	0.005	0.7	12.2	15	99.88%
	0.004	0.7	12.9	15	99.88%
	0.003	0.7	18.1	19	99.89%
	0.002	0.8	23.1	25	99.90%
	0.0015	1	29.3	30	99.91%
	0.0014	1.2	29.3	31	99.914%
	0.001		52.2	33	99.92%

Table 4.6 Time versus threshold comparison between Matlab SPA and Gams SPA in case 3, number of paths and covered emissions for each threshold value.

For all threshold values until 0.0014%, the running times obtained through Gams-SPA are consistently lower than the running times obtained through Matlab-SPA. The trend is almost stable until the 6th tier for both SPA versions. After the 6th tier the Gams-SPA keeps the same trend as before, while Matlab-SPA records an increased trend. We can conclude that for this specific case, Gams-SPA is superior to Matlab-SPA. This superiority is kept only as it has memory for running the model. At threshold 0.001% the memory required by Gams-SPA for listing equations and solving the system exceeds the amount available (1.86Mb).

In this particular case, almost all impact occurs in the first tier. The paths found are long (see table 4.3) and have a very small impact. First path have a contribution of 99.49% while all other paths found for a threshold of 0.014% have a contribution of 0.43%.

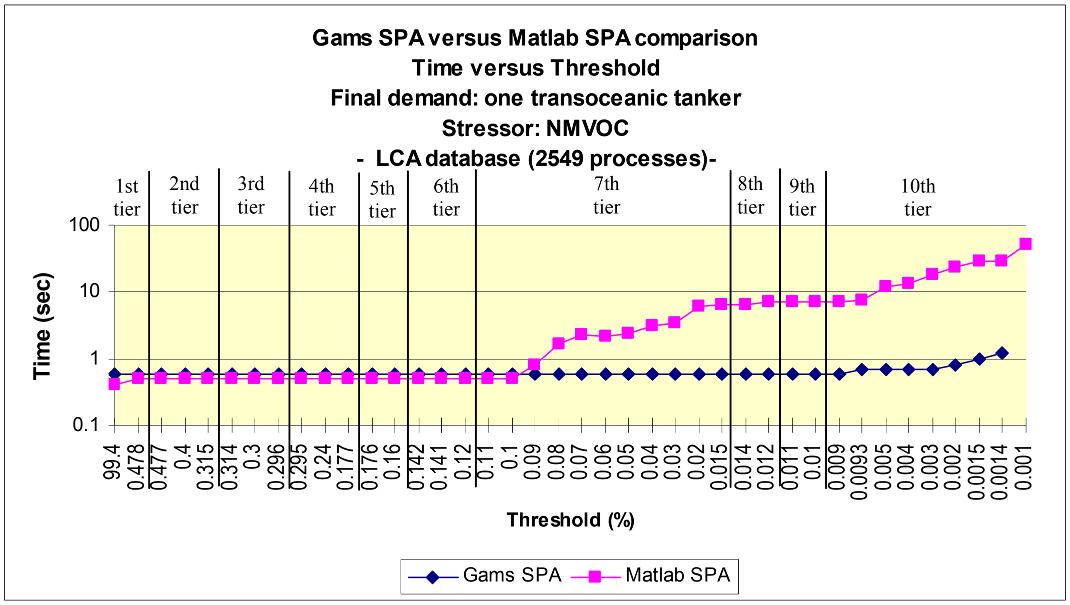


Figure 4.3 Time versus Threshold comparison for Matlab SPA and Gams SPA in case 3

Matlab sparse matrices are written out to the matdata.gms file ordered by rows. This is carried out in the interface by transposing the sparse matrix internally, and then writing out the appropriate values in row oriented format. This greatly increases the speed of reading in large sparse matrices into Gams. Dense matrices are not treated in this manner, so such matrices may read into Gams more slowly. This feature can contribute to the reduced times obtained with the sparse matrix in test case 3, compared with the dense matrices in test cases 1 and 2.

If we compare the results for the previous three cases, it seems that Gams-SPA is superior to Matlab-SPA for big data systems, as long as memory is available for computations. However, before taking conclusions it is worth to revise the older versions of Gams-SPA in order to give more insights on how the memory is allocating in computations and how computation time is influenced by syntax.

Computation times are presented in table 4.7 for two old versions, named version A and version B. Input data from case 2 was used for obtaining the results.

The Gams models previously developed are biased in the same way: (1) unused subsets were preprocessed in Gams so as to contain all sectors, instead of only one sector used in actual version and (2) a different way of separating used tiers by unused tiers: "card" operators instead of s_t parameters in actual version.

The "card" operator counts first the total number of sectors used. Then it is used to count the number of elements from the last subset of a group of paths extending until a tier t. This number corresponds to the total number of paths extending until tier t (see formula 3.17 and attached clarification). If the number of elements in the last subset is equal to the total number of sectors used, tier t does not create any paths and the corresponding expressions in the objective function are zeroed. This formulation was dropped because it is less intuitive than using s_t parameters. However, running times are the same for both versions.

Tiers needed	**Threshold %**	**Version A time (sec)**		**Version B time (sec)**
1 tier	54.8	1.1	≤	1.1
	7.37	1.1		1.1
2 tiers	7.36	1.1	≤	1.1
	4.47	1.1		1.1
3 tiers	4.46	1.2		1.2
	3	1.2	≤	1.3
	2.46	1.6		1.7
4 tiers	2.45	1		1.1
	1.7	2	≤	2.2
	1.32	2.2		2.4
5 tiers	1.31	1.2		1.2
	1	5.3		5.4
	0.74	13	≤	13.9
6 tiers	0.73	2		2
	0.6	6.1		6.6
	0.5	14.8	≤	16
	0.43	18		19.5
	0.42	42.1		45.1
	0.412	43.6		45.2
7 tiers	0.411	3.5		3.5
	0.38	5.1		5.3
	0.34	8.5		8.9
	0.3	11.6	≤	12.1
	0.28	15		16
	0.27	19.4		21.3
	0.26	64.2		69
	0.232	83.9		88.7
8 tiers	0.231	7.7		5.2
	0.23	8		5.3
	0.22	8.4		5.8
	0.21	12.5		9.9
	0.2	15.6		12.1
	0.19	18.9		14.7
	0.18	22.2	≥	16.5
	0.17	36.5		27.6
	0.16	49.8		37.4
	0.15	86.5		67.4
	0.14	215.8		188.4
	0.1304	369.3		333

Table 4.7 Computation times for two old versions

Common:

- "card" operator instead of s_t switches
- unused tiers contained all sectors

Version 1:
objective function lists all terms associated with paths

Version 2:
objective function does not list the terms associated with unused paths

From one tier to 7 tiers needed for optimization, the time values are equal or slightly bigger for the second version compared with the first version.

If 8 tiers are needed for optimization, the time required by the second version is smaller than the time required by the first version.

Making a parallel with the actual Gams version of the model, the sorting of the used tiers from the unused tiers discussed above for versions A and B corresponds with the first sorting criterion from actual version.

Version A and version B differentiate by the fact that version B included the second criterion, exactly in the same way it is designed for the actual Gams implementation, but version A didn't. The time obtained for both two Gams versions can be seen in table 4.7.

The variation determined by tiers is common for both versions and will be discussed later. Now only the relative time values are compared. If less than 7 tiers are needed for generating the model, the version without the second condition performs better in terms of time. For the threshold values requiring 8 tiers to search for paths, second condition reduces the time in which the results are obtained. The memory required for lower threshold values exceeded the available memory.

The time results are in a small range for all cases, but it seems that for an increased number of computations, the second condition can reduce the total computation time. This was the reason for which the second sorting criterion was kept in the final version.

The next table (4.8) is used to show how the first criterion influences the memory and running times. Version B presented lists all elements in unused subsets. For version C and actual version (MIP_spa) the number of elements in the unused subsets are calculated differently: actual version has only one element, the same as first subset; Version C transfers from Matlab to Gams the first p elements, where p is chosen as the variable which does not correspond to the number of elements in other sets. Example: if subset I has one element, subset J has two elements and subset K has five elements, p takes value 3, because values 1 and 2 are already used to count the number of elements in subsets I and J. If subset K would have three elements, p would take value 4. If all numbers from 1 to 10 happens to correspond to the number of elements from each subset, then p takes value 11 and 11 elements will be transferred in each unused set.

Tabel 4.8 presents the times for all three versions mentioned above. Version B time results are plotted against the times obtained with the actual version in figure 4.4. If we look first at the times obtained for version B, we observe a "saw" shape. For every new tier the running times first drop compared with the times obtained at the previous tier, than increase fast. For computations of more than 8 tiers available memory was not enough. We conclude that the number of elements in unused sets is very important for the total time. We can assume that counting the sets elements inside the Gams program is an expensive procedure. It is possible that Gams memorizes the number of elements in each subset from previous iterations, but for each new element added by a lower threshold at actual tier applies the operator for each new path created. Because of these feature it is so important not to have the entire list of set elements in the unused sets.

Reducing the number of element in the unused sets canceled the results' dependence on the number of tiers needed for optimization. It also made available a lot of memory since the results can also be obtained for thresholds requiring nine tiers.

Tiers needed	**Threshold %**	**Version B time (sec)**	**Version C time (sec)**	**MIP–spa time (sec)**
1 tier	54.8	1.1	1.4	1
	7.37	1.1	1.4	1
2 tiers	7.36	1.1	1.4	1
	4.47	1.1	1.4	1
3 tiers	4.46	1.2	1.4	1
	3	1.3	1.4	1
	2.46	1.7	1.4	1
4 tiers	2.45	1.1	1.4	1
	1.7	2.2	1.4	1
	1.32	2.4	1.4	1
5 tiers	1.31	1.2	1.4	1
	1	5.4	1.4	1.1
	0.74	13.9	1.5	1.1
6 tiers	0.73	2	1.5	1.1
	0.6	6.6	1.5	1.1
	0.5	16	1.5	1.1
	0.43	19.5	1.5	1.1
	0.42	45.1	1.5	1.1
	0.412	45.2	1.5	1.1
7 tiers	0.411	3.5	1.5	1.1
	0.38	5.3	1.5	1.1
	0.34	8.9	1.4	1.1
	0.3	12.1	1.9	1.2
	0.28	16	1.7	1.2
	0.27	21.3	1.6	1.2
	0.26	69	1.9	1.3
	0.232	88.7	2	1.4
8 tiers	0.231	5.2	2	1.4
	0.22	5.8	2	1.5
	0.21	9.9	2.5	1.6
	0.2	12.1	2.3	1.6
	0.19	14.7	2.6	1.8
	0.18	16.5	2.8	2
	0.17	27.6	3.6	2.2
	0.16	37.4	4	2.7
	0.15	67.4	5.6	3.6
	0.14	188.4	11.5	7.4
	0.1304	333	17.2	10.9
9 tiers	0.1303	X	19.4	13.5
	0.1	X	54.1	35.9
	0.074	X	817	550.7
10 tiers	0.073	X	X	701
	0.06	X	X	2105
	0.01	X	X	X
	0.005	X	X	X

Table 4.8 Running times for three different versions of Gams programs

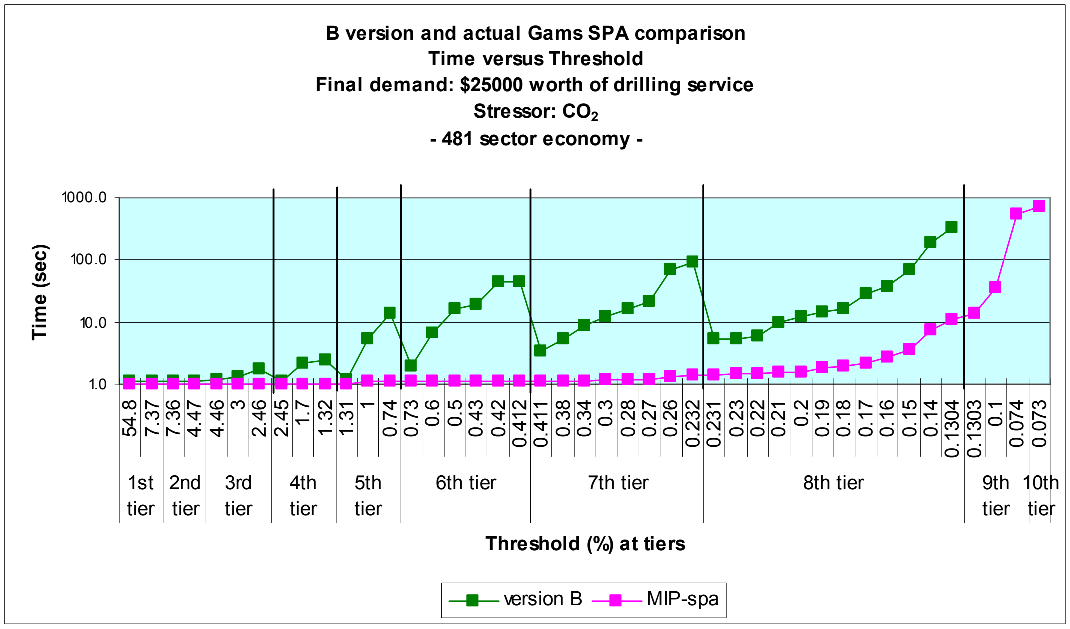


Figure 4.4 Time versus Threshold comparison for old and new "first criterion"

Table 4.8 presents the alternative times results for version C, with four elements in the unused sets and actual MIP_spa version with only one element in unused sets. Compilation times are smaller for the actual version and less memory was used for similar optimizations. If the last result of version C was of 817 seconds for a threshold of 0.074%, in the actual version the same available memory was used to optimize the model at a threshold of 0.06% and to find eight new paths (see table 4.6).

An even older version tried to implement the basic form of SPA, as presented in chapter 3. Calculating path values upfront and including the constraints in the model, as presented, was a very expensive way of optimizing the model, both in term of time and memory. This experience showed that it is better not to calculate the emission values up-front, but to introduce them in objective function in detailed form instead.

We conclude that the total number of sectors from a network counts less than the number of computations Gams needs to do in order to create the model and solve it. For sparse data systems, as presented in case 3, Gams performs better than Matlab, which searches all network nodes one at a time. For dense data systems Matlab performs better than this program version. It is number of computations which count and not the size of input files.

Chapter 5

Discussions

SPA is a very useful tool for reducing environmental impact at global and local level. It can be used by individual producers directly in their effort to reduce the impact of their products and activities. Governments and consumers have an indirect effect through the requirements they put upon producers. These requirements take the form of orders they transfer to the producers or even restrictive actions placed upon the producers as restrictions from buying products with high environmental impact. Governments can also use taxes, subsidies and standards for shaping the form of the final demand, with the declared aim of reducing the overall environmental impact.

Producers, governments and consumers need to know where they have the highest leverage in reducing emissions associated with the products they target. SPA is necessary to identify the chain of processes and sectors offering this leverage because if provides a comprehensive insight in the products network.

The existing methods for this type of analysis are formulated as specialized algorithms. An alternative is to explore the potential utility of mathematical programming for contribution analysis in general and SPA in particular, which is still unknown. The underlying idea is that a good program should be able to perform various types of contribution analysis, including SPA, with minor adaptations and the same generic solver. However, for a model developed to solve multiple problems we can expect to have disadvantages compared with specialized algorithms.

The aim of this thesis is not to provide a final program, but to explore different formulations and approaches, their requirements and limitations. The insights gained from this research need to be used to develop a working version for SPA using mathematical programming. This program is intended to be used as a starting point for future development in SPA.

5.1 Achievements

A mixed integer program was developed for SPA and implemented in Gams. The SPA model presented here finds the paths describing the process relationships that link the final delivered product and the emitting processes.

A "gams mex interface" was used to pass data between Gams and Matlab. This approach required extensive assistance from Matlab. We point out here requirements and limitations of this approach and solutions found to them.

Because the size of the problem was too big in its original phase, pre-processing was required to reduce it. This was done in Matlab. First, the data range was limited to ten production layers. Classical contribution analysis and Taylor series expansion were used to evaluate the impact inside the individual layers of production system. The results were used in the next round of pre-processing. This refers to cutting off some processes if their impact is below a threshold value, before initiating the optimization process. The remaining processes were assembled in sets and formatted for Gams.

The model implemented in Gams was functional for all types of inputs. It included ten sets and ten equations and returned ten groups of paths. Several formulations have been tried for the optimization model and the current version has been found to be the most effective in terms of memory and time required for optimization. This performance was obtained by exploiting the block angular structure of the model to reduce the size of the problem to the minimum. Smaller models need less memory and time to run.

From a technical point of view, binary parameters have been used to switch on and off parts of the model. This change came with a very big reduction in memory and time. The second change was introduced to control the coefficients of the objective function in order to return only paths with impact above a certain threshold. This change reduced compilation time. Other formulations needed more time and memory to solve the same problem.

Three processes have been recognized to consume time and/or memory: (1) transferring data from Matlab to Gams; (2) transferring paths from Gams to Matlab and (3) compiling and executing the Gams program.

Time could not be recorded separately in these three processes. However, running the same optimization twice on the same data gave reduced times at the second iteration, compared with the first. This is because data was already saved by the interface in the first iteration. No connection could be established between the size of the input parameters and the time needed for data transfer from Matlab to Gams. Comparing time records from Matlab with time records from Gams, we conclude that most of the time was used for executing the program.

Time grows exponentially with the number of elements in the unused sets. When unused sets contained all products, time increased fast when more paths from the same length were found and dropped a lot when the search extended to the next tier. This problem was solved by employing only one element in each unused set, but the cause of this behavior remains unclear. Transferring empty sets to Gams was not a viable option because trying to define equations on empty sets was reported as an error in Gams.

The amount of memory required for compiling and executing the optimization program has been reduced by employing the binary parameters used as switches. Using only one element in the unused tier also reduced memory since optimization ran successfully for lower threshold values. The amount of memory necessary for transferring data from Matlab to Gams could not be quantified.

Transferring the obtained paths from Gams to Matlab was very memory intensive. Matlab read the paths as multi-dimensional variables and the amount of memory used for this type of inputs grows very fast with each new dimension employed. This is the only way in which the transfer software knows to handle tuples. As long as decision variables were defined as tuples in the program, there were no adjustments that could be done to reduce the time for the found paths.

The number of output parameters required by Matlab for the system call to Gams needs to correspond with the number of parameters calculated in Gams. The calculated parameters were transferred to Gams as multidimensional variables, even the one with zero value. The memory required by Matlab to read all these multi-dimensional parameters was very large. If this transfer had been allowed for paths with zero value without further pre-processing in Gams, available memory would have been exhausted even for small data systems. This memory problem was solved by employing one-dimensional alternative parameters with different names to be used for transferring zero-value paths taking advantage that name recognition in Matlab could be switched off.

Matlab was necessary for pre-processing and formatting sets and parameters for transfer to Gams. It was also used to recover the elements of the paths from multidimensional representation. However, when the last element of a path was defined based on a one-element set, that element was lost in transformation because Matlab ignores the corresponding dimension by default. This case had to be handled separately in the Matlab code.

In transferring data between Matlab and Gams the sets had to be assembled in a big table and printed in Gams from Matlab. No other solution was found in documentation for sending input data and recovering results in the same system call. The formatting procedure had to take into account the need to avoid including in the labels used as set names non-alphanumeric symbols such as dots, commas, slashes, etc. The solution found was to convert the real names into number-indexed names before sending them to Gams. Of course, the conversion needed to be reversed after paths have been recovered in Matlab.

Scaling is required in Gams for parameters so that their magnitude is centered on one and the ratio between the highest value and the lowest one ranges between 0.01 and 100. No stressor in the LCA database was situated in this category, so Gams was not able to read the parameter in the original version. In order to allow testing on this database, the stressor values for some products have been zeroed. This is a very important limitation in applying this program on the LCA database.

A condition could have been introduced in the program to limit the number of paths. This condition was not introduced for the reasons explained below.

Running SPA with the specialized algorithm made obvious the fact that a very large number of paths is obtained for low threshold values. We cannot hope to obtain all these paths in Gams using the current formulation of the problem because of the high memory costs associated with running and transferring paths in Matlab. The logical conclusion was to limit the number of paths so that both execution and transferring procedures could be accommodated with the amount of memory available.

Limiting the number of paths can be done in two ways: (1) choosing a maximum number of paths and (2) choosing only the paths above a certain threshold value.

The threshold determines the size of the model and the execution times. However, the number of paths determines the amount of memory necessitated by transferring the paths to Gams. Only a limited amount of memory is available. If too many paths are requested, the system will run out of memory. The number of requested paths needs to be reduced gradually until optimization is successful or until not even one path can be transferred. This methodology requires excessive testing and can be used for future research.

Instead of the procedure described in the paragraph above, we chose to use the same parameter to deal both with execution times and with transfer times. Using a threshold limits the size of the model, execution times, memory required for execution and memory required for data transfer. This system is easier to employ and guarantees that execution time and transfer times are connected.

After developing, the program was tasted on all databases available and used for selected case studies, as required.

From the case studies, we found that the running time for the model increases when the threshold is reduced, as long as memory is available. A relatively small number of paths covers most of the emissions and each new path found has a lower impact than the previous one.

For the Norwegian economy, information resolution is too low for decision-making, but it is obvious that manufacturing and transport sectors dominate the economy and influence each other.

Performing the SPA for a "drilling service" as part of the US economy we found that the service itself produces more than half of the CO_2 involved in its production. Sectors: "blast furnaces and steel mills" and "electric services" create many paths. Taking measures to reduce their impact will consequently reduce the impact of the drilling service. However, the total impact of the above mentioned sectors needs to be estimated using contribution analysis in order to include the contribution of all paths which are not listed here.

Performing SPA on the LCA data system for a transoceanic tanker required changing stressor data. The result points that almost all impact comes from the product itself. However, no politics can be recommended in this particular case because reality was changed. We traded an exact solution for the overall model and demonstrated both the general applicability of the program on sparse data systems and its superior results.

After testing the model on different databases, we were supposed to fulfill the following goals:

The first goal was to find out which algorithm is faster for a specific type of data. We can conclude that the specialized algorithm performs better than our generalized algorithm on dense data systems. However, for sparse data systems the generalized algorithm performs better.

The second goal was to find out the limits of the algorithms, respectively the biggest data system they can be used for. We found out that the size of the data system it is not important. The data structure matters more.

The third goal was to determine the weaknesses of the linear programming model. Increased memory usage, reduced computational speed and high dependence on Matlab can be pointed as the most important weaknesses of the program. This program is functional, but it still needs to be improved in order to overcome these problems.

5.2 Assessment of results

This program was only tested on three cases. The same paths have been found for both types of SPA. However, the utility of paths in the decision-making process depends on the type of database used for calculating them.

If database resolution is too low, as for the Norwegian economy, paths found cannot be used for decision making. US database resolution makes the paths much more useful, but the entire contribution of the sector needs to be considered at each chosen tier in order to estimate the total impact. Exemplifying, there are five paths based upon the contribution of sector “electricity” in the list of paths presented and there may be more paths which are not listed. Placing a requirement on this sector will affect all paths, not only the listed ones. In addition, the inevitable errors in compiling large databases as this one suggest that care should be taken when interpreting the results of the decision-making process.

LCA database high resolution should be very useful for producers, because it affects their technology directly. Unfortunately, this database could not be used in its original format because of Gams requirements for data range. Changing this data showed the applicability of this program on LCA-size databases with coefficients in the ranges required by Gams program.

We identified a case when a four-element path has an impact higher than that of the path constructed with its first three elements. However, other particular cases need to be tested as well: (1) finding more paths as the one presented above, but with more than one element difference in length; (2) performing a SPA on a sector without direct emissions; (3) seeing how Gams SPA handles a sector without direct emissions but with polluting inputs. The existence of more of these sectors in the database might increase the time and memory necessary for obtaining the results. This increase can be determined by the requirement that all these inputs are considered as possible paths.

The particular cases identified above and maybe other of this type, which have not been identified yet, demonstrate the necessity of performing additional tests for validating this program.

We already mentioned that this program is just a functional intermediary version for doing a SPA using operational research methods. Time result dependence on the number of unused sectors was minimized, but we could not explain why this dependence exists. Also, coefficient testing was employed based on one test case. These problems suggest that more testing is necessary in order to improve the current program version.

5.3 Future research

Future research needs to be conducted for improving this version of SPA with the aim of reducing the memory usage and increasing computational speed. Introducing a condition to limit the number of paths, as explained in subchapter 5.1, can be useful in understanding more about memory usage and allocation.

A non-linear program generally uses less memory than a MIP. If solvers are available, this is a very promising direction of research.

Alternative solutions might be found by using the other interface available for data transfer from Gams to Matlab: using two mex procedures to read and write a gdx file which can be interpreted by built-in procedures in Gams. Alternatively, writing the paths found directly in Excel is another option to consider and evaluate in terms of times and memory resources used. To avoid external problems created by inefficient software, it might be a good thing to try to reduce its dependency on external programs.

The final version of the program must be able to perform different types of contribution analysis. It must also run for any size of data input, equally dense and sparse and accommodate thresholds as low as 0.001%. The performance in terms of running times and memory usage must be good enough for using it in decision-making process by producers and governments.

This mixed integer program shows that mathematical programming is a very useful tool for contribution analysis in general and SPA in particular. Preliminary testing shows its advantage for performing analysis on sparse data systems compared with the classic method involving Matlab. However, Matlab's specialized algorithm performs better for dense data systems. Many of the requirements and limitations created by the software involved in different steps have proved manageable. Future testing and research it is needed for obtaining a final version with performance comparable or even superior to the existing specialized algorithm.

Acknowledgments

I hereby express my sincere acknowledgments to my supervisors Anders Hammer Strømman and Richard Wood for their suggestions, support and encouragement during this thesis.

I also want to thank my family and friends for the patience, love and support they offered constantly during this thesis.

Bibliography:

IPPC official web site (last visited June 10th, 2009) – Assessment reports http://www.ipcc.ch/ipccreports/index.htm

Greenhouse Gas Inventory Data on UNFCCC website (last visited June 10th, 2009) http://unfccc.int/ghg_data/items/3800.php

Gams official website – Cplex information (last visited June 18th, 2009) http://www.gams.com/dd/docs/solvers/cplex.pdf

Gams official website-MIP information (last visited June 18th, 2009) http://www.gams.com/dd/docs/solvers/cplex.pdf

Michael C. Ferris Matlab and Gams: Interfacing Optimization and Visualization Software, Mathematical Programming Technical Report 98-19, November 1998 (last visited June 3, 2009)
http://www.cs.wisc.edu/math-prog/tech-reports/98-19.pdf

Matlab and Gams: Interfacing Optimization and Visualization Software, the University of Winsconsin (last visited June 3, 2009)
http://www.cs.wisc.edu/math-prog/matlab.html

Defourny, J. and Thorbecke, E. (1984) Structural path analysis and multiplier decomposition within a social accounting matrix framework, *Economic Journal*, vol. 94, pp. 111-136.

Dorfman, Robert and Samuelson Paul A. and Solow, Robert M (1958) Linear programming and economic analysis, McGraw-Hill, pp. 41-45

Dunchin, Faye (1994) Input Output Analysis and Industrial Ecology, The Greening of Industrial Ecosystems, pp. 61-68.

Dunchin, Faye and Lange, Glenn-Marie (1995) The choice of technology and associated changes in prices in the U.S. economy, *Structural Change and Economic Dynamics,* vol. 6, pp. 335-357

Dunchin, Faye (2004) Input-output economics and material flows, Rensselaer working Papers in Economics, No. 0424 http.rpi.edu/dept/economics/www/workingpapers/

Dunchin, Faye (2005) A world trade model based on comparative advantage with m regions, n goods, and k factors, *Economic System Research*, vol. 17, No. 2, pp. 141-162.

Hertwich, Edgar (2005) Lifecycle approaches to sustainable consumption: A critical review, *Environmental Science and Technology*, vol. 39, No. 13, pp. 4673-4684

Leontief, Wassily (1941) The Structure of American Economy: 1919-1929, Oxford University Press, New York 1941.

Lenzen, Manfred (2001) A generalized input-output multiplier calculus for Australia, *Economic System Research*, vol. 13, No.1, pp. 65-92.

Peters, Glen P. and Hertwich, Edgar G. (2006) Structural analysis of international trade: Environmental impacts of Norway, *Economic Systems Research*, vol. 18, No.2, pp.155 – 181.

Rardin, Ronald L.(1998) Optimizations in Operations Research, Pearson Education 1998, pp. 23-27.

Strømman, Anders H. and Dunchin, Faye (2006) A world trade model with bilateral trade based on comparative advantage, *Economic Systems Research*, vol. 18 No. 3, pp. 281-297.

Strømman, Anders H. and Hertwich, Edgar G. and Dunchin, Faye (2009) Shifting trade patterns as a means of reducing global carbon dioxide emissions – A multiobjective analysis, *Journal of Industrial Ecology*, vol. 13, No. 1, 2009 pp. 38-57.

Weisz, Helga and Dunchin, Faye (2006) Physical and monetary input-output analysis: What makes the difference? , *Ecological Economics* vol. 57 pp. 534-541.

Williams, Paul H. (2006) Model Building in mathematical programming 4th edition, John Wiley and Sons Inc., England, 2006 pp. 3-5, 42-43,144-145, 152.

APPENDIX A: General Algorithm for Master Problem

File name: Spa.m

```
% spa.m performs structural path analysis on a Leontief type system
% with 1 stressor

clc
clear all

% 1.Matlab and Gams programs default variable (do not change):

%    Tmax    - maximum tiers programmed now in Gams and Matlab;
% can be increased/decreased, but require changing Gams and Matlab code in a
% consistent way (add sets/output variables name and content in Matlab->Gams,
% update parameters and equations in Gams)

Tmax=10;

% 2.Input variables to change:

%      a)input file (.mat). Files from:
% K:\indecol\PROJECTS\MatlabLCAMarch2008\InputData
% require preprocessing for benchmark reason. This includes choosing the final
% demand vector y and the vector of selected stressor Fstr.

inputfile = 'C:\Documents and Settings\vlad\Desktop\MatlabLCAMarch2008
\InputData\tanker1.mat';

load(inputfile);

%      b)Tol_rel(threshold) is the percentage at which the emissions are
% ignored

Tol_rel=0.07;

% 3.Loaded input variables

%   3.1 Used for function spa ('sec' is used for sector)
%       a)A(sec,sec)     - Inter-industry coefficient matrix
%       b)I (sec,sec)    - Identity Matrix
%       c)F(sec;sec)     - Stressor intensity matrix
%       d)y(sec,1)       - final demand vector
%       e)PRO(sec;sec)   - matrix with names for sectors

%   3.2 Required by function lca for benchmarking purposes
%       f)cat=ones(sec,2)       - modified impact categories
%       g)C=[1,0;0,1]           - modified characterization matrix
%       h)CAT=[GWP GWP]         - modified vector of impact category
%       i)STR(2,5)=CO2          - modified stressor (CO2) matrix
%       j)IMP=[GWP GWP;GWP GWP]- modified matrix of impact category

% clear unnecessary variables
clear cat C CAT STR IMP;

% 4. Endogenous variables - nomenclature
%       Fstr           - vector of emissions intensities for selected
```

```
                        stressor
%       nr_sec        - total number of sectors in A matrix
%       Tol           - threshold absolute value
%       R             - matrix of presorted sectors
%       T             - maximum number of tiers required
%       t             - current tier
%       last_tier     - last tier of the longest path found
%       e_total       - total emissions associated with final demand
%       e_covered     - emissions covered by all paths found
%       Sectors_names - vector of names for sectors
%       Sectors_index - vector of indexed names for sectors
%       sect          - set containing all sectors(indexed names)
%       sd_sect       - set sect declaration for Gams
%       sd_k          - set k declaration for Gams
%       k(sect)       - name of set k in Gams
%       s3_set        - content of set k (i,j,k...correspond to 1,2,3..)
%       s3            - switch for turning on/off calculations for set k
%       e3            - array of emissions value for paths covering 3 tiers
                       (paths i-j-k)
%       Pmax          - number of paths found
%       PATH_sorted   - structure recording for every path found the
%                       emissions and the succession of sectors with their
%                       original names

%% Presorting

[R,Tol,T,nr_sec,e_total,Fstr] = presorting(Tmax,Tol_rel,A,I,F,y);

%% Preparing sets for Gams

% Get sectors'names

Sectors_names=PRO(:,1);

% Indexing sectors'names

Sectors_index=cell(nr_sec,1);
for i=1:nr_sec
    Sectors_index{i}=num2str(i);
end

% Set 'sect' content declaration

sect = Sectors_index;

% Active sets content declaration, switch turned on

for t=1:T
    eval(['s' num2str(t) '_set = Sectors_index(logical(R(:,' num2str(t)
')));'])
    eval(['s' num2str(t) '=1;'])
end

% Inactive sets content declaration: one sector (Gams does not accept empty
sets), switch turned off

for t=(T+1):Tmax
    eval(['s' num2str(t) '_set = Sectors_index(logical(R(:,1)));'  ])
    eval(['s' num2str(t) '=0;'])
end

% clear unnecessary variables
clear t Sectors_index R
```

```
% Assembling sets for transfer to Gams

% Post declaration

pd = cell(1,1);
pd{1,1} = 'SETS';

% Set type and name declaration

sd_sect = cell(1,1);
sd_i = cell(1,1);
sd_j = cell(1,1);
sd_k = cell(1,1);
sd_l = cell(1,1);
sd_m = cell(1,1);
sd_n = cell(1,1);
sd_q = cell(1,1);
sd_r = cell(1,1);
sd_t = cell(1,1);
sd_u = cell(1,1);

sd_sect{1,1} = 'sect';
sd_i{1,1} = 'i(sect)';
sd_j{1,1} = 'j(sect)';
sd_k{1,1} = 'k(sect)';
sd_l{1,1} = 'l(sect)';
sd_m{1,1} = 'm(sect)';
sd_n{1,1} = 'n(sect)';
sd_q{1,1} = 'q(sect)';
sd_r{1,1} = 'r(sect)';
sd_t{1,1} = 't(sect)';
sd_u{1,1} = 'u(sect)';

% Total lines

slsh = cell(1,1);
slsh{1,1} = '/';

% Table assembly

cellset =[pd;sd_sect;slsh;sect;slsh;
                sd_i;slsh;s1_set;slsh;
                sd_j;slsh;s2_set;slsh;
                sd_k;slsh;s3_set;slsh;
                sd_l;slsh;s4_set;slsh;
                sd_m;slsh;s5_set;slsh;
                sd_n;slsh;s6_set;slsh;
                sd_q;slsh;s7_set;slsh;
                sd_r;slsh;s8_set;slsh;
                sd_t;slsh;s9_set;slsh;
                sd_u;slsh;s10_set;slsh;
                ];

tabsize = size(cellset,1);

% Delete old file

delete set.gms
```

```
% Gams Optimization

% Optimization started

fprintf('\r GAMS optimization \n')

tic

% Write File

fid = fopen('set.gms','w');
for line = 1:tabsize
    fprintf(fid,'%s\n',char(cellset(line,:)));
end
status = fclose(fid);

% Setting GAMS output mode

gams_output = 'std';

% Formatting Data for Gams Export

Matrix.name='A';
Matrix.val=A;
Matrix.labels={sect,sect};

Stressors.name='F';
Stressors.val=Fstr;
Stressors.labels={sect};

demand.name='y';
demand.val=y';
demand.labels={sect};

% clear unnecessary variables
clear A I F Fstr PRO tabsize line sd_sect sd_i sd_j sd_k sd_l sd_m...
    sd_n sd_q sd_r sd_t sd_u fid pd slsh cellset;

% optimization

[e_covered,e1,e2,e3,e4,e5,e6,e7,e8,e9,e10] = gams('MIP_gams',Matrix,...
Stressors,demand,'Tol','s1','s2','s3','s4','s5','s6','s7','s8','s9','s10');

% clear unnecessary variables
clear  Matrix Stressors demand s1 s2 s3 s4 s5 s6 s7 s8 s9 s10;

toc

% Prepare data for sorting

% List emissions for 10 tiers in the order received from Gams

e_list={e1,e2,e3,e4,e5,e6,e7,e8,e9,e10};




% Find last nonzero emission value/longest path length

for index_list=length(e_list):-1:1
```

```
   if length(find(e_list{index_list}))~= 0
       last_tier=index_list;
       break
   end
end

% Calculate total number of paths

Pmax=0;
for index_list=1:last_tier
    Pmax=Pmax+length(find(e_list{index_list}));
end;

% List all sets

sets_list=...
{s1_set,s2_set,s3_set,s4_set,s5_set,s6_set,s7_set,s8_set,s9_set,s10_set};

clear index_list


% Sorting

fprintf('\r sorting \n')

tic

PATH_sorted=sorting(y,e_list,sets_list,Sectors_names,Pmax,last_tier);

toc


% Printing results

fprintf('\r Emissions covered: %d',e_covered)

fprintf('\r Total emissions: %d',e_total)

percent=e_covered/e_total*100;

fprintf('\r The %d paths determined (listed below) cover %d percent of total
emissions\r',Pmax,percent)

 for i=1:length(PATH_sorted)
     fprintf('\r %g \n', PATH_sorted{i});
     for j=1:size(PATH_sorted{i,2},2)
         disp(char(PATH_sorted{i,2}{1,j}));
     end
 end

 clear i j ans e_list sets_list
```

APPENDIX B: Matlab function for presorting

File name: Presorting.m

```
function [R,Tol,T,nr_sec,e_total,Fstr] = presorting(Tmax,Tol_rel,A,I,F,y)

% presorting.m performs a presorting of the sectors at every tier by
% removing the ones with the environmental contribution less than a given
% threshold. It returns a matrix of presorted sectors (sectors by tiers)
% with values of 1 for sectors and tiers with environmental contribution
% more than the threshold and 0 for sectors and tiers with environmental
% contribution less than the threshold.

% Steps:
% 1) calculate total emissions
% 2) calculate output matrix per tiers
% 3) scale output per tiers with pollution intensity to get environmental
%    impact per sectors and tiers (remaining _emissions). Every element
%    represents environmental impact of the sector (direct and indirect)
%    at respective tier
% 4) divide environmental impacts per sectors and tiers with the total
%    emissions to get relative environmental impacts per sectors and tiers
%    (remaining_emissions_rel)
% 5) for every sector and tier, if the relative impact per sector and tier
%    is smaller than the threshold (as percentage) the sector is ignored
%    at that tier.


% Get the stressor vector
Fstr=F(1,:);

% Calculate Leontief inverse
L=inv(I-A);

% Get the number of sectors
nr_sec = length(A(:,1));

% Calculate total emissions associated with the final demand
e_total=Fstr*L*y;

% Calculate output_matrix, remaining emissions matrix and remaining
% emissions matrix in relative values for every tier

output_matrix(:,1)=y;
remaining_emissions=zeros(nr_sec,Tmax);
remaining_emissions_rel=zeros(nr_sec,Tmax);

remaining_emissions(:,1)=Fstr*L*diag(y);
remaining_emissions_rel(:,1)=remaining_emissions(:,1)/e_total*100;

for t = 1:Tmax
    output_matrix(:,t+1)= A*output_matrix(:,t);
    remaining_emissions(:,t+1)=Fstr*L*diag(output_matrix(:,t+1));
    remaining_emissions_rel(:,t+1)=remaining_emissions(:,t+1)/e_total*100;
end
```

```
% define and calculate the matrix of presorted sectors

R = ones(nr_sec,Tmax);
R(:,1) = y;

for index_sec = 1:nr_sec
     for t = 2:Tmax
         if (remaining_emissions_rel(index_sec,t) < Tol_rel)
             R(index_sec,t)=0;
         end
     end
end

% find last tier with nonzero values (necessary number of tiers)

for i=Tmax:-1:1
    if sum(R(:,i))~= 0
        T=i;
        break
    end
end

% calculate threshold in absolute value
Tol=Tol_rel/100*e_total;

end
```

APPENDIX C: Gams program for optimization

File name: MIP_gams.gms

```
$include set.gms

ALIAS(sect,cest);

PARAMETERS       A(sect,cest)
                 F(sect)
                 y(sect)     ;

SCALAR    Tol, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10  ;

$if exist matdata.gms $include matdata.gms


FREE VARIABLES

e_covered                 Emissions covered by selected paths ;

BINARY VARIABLES

b1(i)                     path: i
b2(i,j)                   path: i-j
b3(i,j,k)                 path: i-j-k
b4(i,j,k,l)               path: i-j-k-l
b5(i,j,k,l,m)             path: i-j-k-l-m
b6(i,j,k,l,m,n)           path: i-j-k-l-m-n
b7(i,j,k,l,m,n,q)         path: i-j-k-l-m-n-q
b8(i,j,k,l,m,n,q,r)       path: i-j-k-l-m-n-q-r
b9(i,j,k,l,m,n,q,r,t)     path: i-j-k-l-m-n-q-r-t
b10(i,j,k,l,m,n,q,r,t,u)  path: i-j-k-l-m-n-q-r-t-u   ;


EQUATIONS
emissions_covered         objective function

emissions_covered..       e_covered =e=

  sum( (i)$(s1=1), (F(i)*y(i)*b1(i))$(F(i)*y(i)>Tol) )

+ sum( (i,j)$(s2=1), (F(j)*A(j,i)*y(i)*b2(i,j))$(F(j)*A(j,i)*y(i)>Tol)  )

+ sum( (i,j,k)$(s3=1),
(F(k)*A(k,j)*A(j,i)*y(i)*b3(i,j,k))$(F(k)*A(k,j)*A(j,i)*y(i)>Tol)  )

+ sum( (i,j,k,l)$(s4=1),
(F(l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b4(i,j,k,l))$(F(l)*A(l,k)*A(k,j)*A(j,i)*y(i)>To
l)  )

+ sum( (i,j,k,l,m)$(s5=1),
(F(m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b5(i,j,k,l,m))$(F(m)*A(m,l)*A(l,k)*A(k,j
)*A(j,i)*y(i)>Tol)  )
+ sum( (i,j,k,l,m,n)$(s6=1),
(F(n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b6(i,j,k,l,m,n))$(F(n)*A(n,m)*A(m
,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol)  )
```

```
+ sum( (i,j,k,l,m,n,q)$(s7=1),
(F(q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b7(i,j,k,l,m,n,q))$(F(q)*A
(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol)  )

+ sum( (i,j,k,l,m,n,q,r)$(s8=1),
(F(r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b8(i,j,k,l,m,n,q,r)
)$(F(r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol)  )

+ sum( (i,j,k,l,m,n,q,r,t)$(s9=1),
(F(t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b9(i,j,k,l,m
,n,q,r,t))$(F(t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>T
ol)  )

+ sum( (i,j,k,l,m,n,q,r,t,u)$(s10=1),
(F(u)*A(u,t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)*b10(i
,j,k,l,m,n,q,r,t,u))$(F(u)*A(u,t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k
,j)*A(j,i)*y(i)>Tol)  )
;

MODEL MIP_gams/emissions_covered/;

SOLVE MIP_gams using MIP maximizing e_covered;


$libinclude matout e_covered.l


PARAMETERS

*calculating emissions if the paths exists

e1(i)                    emissions for path: i
e2(i,j)                  emissions for path: i-j
e3(i,j,k)                emissions for path: i-j-k
e4(i,j,k,l)              emissions for path: i-j-k-l
e5(i,j,k,l,m)            emissions for path: i-j-k-l-m
e6(i,j,k,l,m,n)          emissions for path: i-j-k-l-m-n
e7(i,j,k,l,m,n,q)        emissions for path: i-j-k-l-m-n-q
e8(i,j,k,l,m,n,q,r)      emissions for path: i-j-k-l-m-n-q-r
e9(i,j,k,l,m,n,q,r,t)    emissions for path: i-j-k-l-m-n-q-r-t
e10(i,j,k,l,m,n,q,r,t,u) emissions for path: i-j-k-l-m-n-q-r-t-u

*0 is the alternate emissions value if the paths do not exist

f2          alternative emission value for path: i-j
f3          alternative emission value for path: i-j-k
f4          alternative emission value for path: i-j-k-l
f5          alternative emission value for path: i-j-k-l-m
f6          alternative emission value for path: i-j-k-l-m-n
f7          alternative emission value for path: i-j-k-l-m-n-q
f8          alternative emission value for path: i-j-k-l-m-n-q-r
f9          alternative emission value for path: i-j-k-l-m-n-q-r-t
f10         alternative emission value for path: i-j-k-l-m-n-q-r-t-u
;


e1(i)$b1.l(i) = (F(i)*y(i))$(F(i)*y(i)>Tol);
$libinclude matout e1 i

f2=sum((i,j),b2.l(i,j)) ;
if(f2 = 0,
$libinclude matout f2
```

```
);
if(f2 ne 0,
e2(i,j)$b2.l(i,j) = (F(j)*A(j,i)*y(i))$(F(j)*A(j,i)*y(i)>Tol) ;
$libinclude matout e2 i j
);

f3=sum((i,j,k),b3.l(i,j,k)) ;
if(f3 = 0,
$libinclude matout f3
);
if(f3 ne 0,
e3(i,j,k)$b3.l(i,j,k)= (F(k)*A(k,j)*A(j,i)*y(i))$(F(k)*A(k,j)*A(j,i)*y(i)>Tol);
$libinclude matout e3 i j k
);

f4=sum((i,j,k,l),b4.l(i,j,k,l)) ;
if(f4 = 0,
$libinclude matout f4
);
if(f4 ne 0,
e4(i,j,k,l)$b4.l(i,j,k,l) =
(F(l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol);
$libinclude matout e4 i j k l
);

f5=sum((i,j,k,l,m),b5.l(i,j,k,l,m)) ;
if(f5 = 0,
$libinclude matout f5
);
if(f5 ne 0,
e5(i,j,k,l,m)$b5.l(i,j,k,l,m) =
(F(m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>
Tol);
$libinclude matout e5 i j k l m
);

f6=sum((i,j,k,l,m,n),b6.l(i,j,k,l,m,n)) ;
if(f6 = 0,
$libinclude matout f6
);
if(f6 ne 0,
e6(i,j,k,l,m,n)$b6.l(i,j,k,l,m,n) =
(F(n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(n)*A(n,m)*A(m,l)*A(l,k)*A(k,j
)*A(j,i)*y(i)>Tol);
$libinclude matout e6 i j k l m n
);

f7=sum((i,j,k,l,m,n,q),b7.l(i,j,k,l,m,n,q)) ;
if(f7 = 0,
$libinclude matout f7
);
if(f7 ne 0,
e7(i,j,k,l,m,n,q)$b7.l(i,j,k,l,m,n,q) =
(F(q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(q)*A(q,n)*A(n,m)*A(m,l
)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol) ;
$libinclude matout e7 i j k l m n q
);

f8=sum((i,j,k,l,m,n,q,r),b8.l(i,j,k,l,m,n,q,r)) ;
if(f8 = 0,
$libinclude matout f8
);
if(f8 ne 0,
e8(i,j,k,l,m,n,q,r)$b8.l(i,j,k,l,m,n,q,r) =
(F(r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(r)*A(r,q)*A(q,n
)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol);
$libinclude matout e8 i j k l m n q r
);

f9=sum((i,j,k,l,m,n,q,r,t),b9.l(i,j,k,l,m,n,q,r,t)) ;
if(f9 = 0,
$libinclude matout f9
);
if(f9 ne 0,
e9(i,j,k,l,m,n,q,r,t)$b9.l(i,j,k,l,m,n,q,r,t) =
(F(t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(t)*A(t,r
)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol);
$libinclude matout e9 i j k l m n q r t
);

f10=sum((i,j,k,l,m,n,q,r,t,u),b10.l(i,j,k,l,m,n,q,r,t,u)) ;
if(f10 = 0,
$libinclude matout f10
);
if(f10 ne 0,
e10(i,j,k,l,m,n,q,r,t,u)$b10.l(i,j,k,l,m,n,q,r,t,u) =
(F(u)*A(u,t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i))$(F(u
)*A(u,t)*A(t,r)*A(r,q)*A(q,n)*A(n,m)*A(m,l)*A(l,k)*A(k,j)*A(j,i)*y(i)>Tol) ;
$libinclude matout e10 i j k l m n q r t u
);
```

APPENDIX D: Matlab function for recovering paths from Gams

File name: getpath.m

```
function [path_em,path_lab]=getpath(en,varargin)
% getpath.m recovers the paths from the multidimensional representation;
% not to be used for e1

% Matlab interprets the values received from Gams as elements of class
% double in an n-dimensional matrix. For every individual path, we need to
% recover the subscripts corresponding to each dimension in order to use
% them later in identifying the sectors contributing to the path.

% Every path has 2 parts: emission value (double) and labels (1 cell);
% labels cell contains a cell array, each cell with the indexed sector name,
% respecting path order
% Example: for e3 elements containing 3 sectors each:
% for path1 from sectors a-b-c
% em1 and lab1(cell)=1st sect.a(cell)+ 2nd sect.b(cell)+ 3rd sect.c(cell)
% for path2 from sectors a-b-d
% em2 and lab2(cell)=1st sect.a(cell)+ 2nd sect.b(cell)+ 3rd sect.d(cell)
% for path3 from sectors a-g-h
% em3 and lab3(cell)=1st sect.a(cell)+ 2nd sect.g(cell)+ 3rd sect.h(cell)
% for path4 from sectors a-v-d
% em4 and lab4(cell)=1st sect.a(cell)+ 2nd sect.v(cell)+ 3rd sect.d(cell)

% path_em is the vector of emissions values [em1 em2 em3 em4]
% path_lab is a cell array with path'labels [lab1 lab2 lab3 lab4]

% Get the vector of indices corresponding to the paths en in the
% n-dimensional array
elem_vector=find(en);
% Example: find(e3)=[4 9 11 20] => 4th,9th,11th and 20th elements of e3
% are different from 0, so there are 4 paths

% Get the number of paths
nelems = length(elem_vector);

% Get the vector of dimensions for the n-dimensional array (the size of
% the array)
dim_vector=size(en);
%Example: size(e3)=[1,20,11] => e3 values have 3 dimensions, with 1 element
%on the first dimension, 20 elements on the second dimension and 11 elements
%on the third dimension

% Get the total number of dimensions
ndims = length(dim_vector);


path_em = zeros(nelems,1);  % vector of emission values
path_lab = cell(nelems,1);  % cell array of labels

for j = 1:nelems
    % get the vector of subscripts
        command = '[i1';
        for i=2:ndims
            command = [command ',i' num2str(i)];
        end
        command = [command '] = ind2sub(dim_vector,elem_vector(j));'];
        eval(command);
    % Example: [i1,i2,i3]=ind2sub([1,20,11],e3(9)) returns i1=1,i2=3,i3=4
    % [1,3,4] is the vector of subscripts of the 9th element of array e3
    % (nonzero) in a 1x20x11 dimensions matrix.

    % making sure that last dimensions with 1 element are not lost (this
    % dimensions are ignored by Matlab)
        if length(varargin)>ndims
            for k=(ndims+1):length(varargin)
                eval(['i' num2str(k) '=1;']);
            end
        end

    % record path emission value
        path_em(j) = en(elem_vector(j));

    % Apply each subscript to the corresponding set (corresponding to the
    % variable elements) to get indexed sector name
    % Example: s1_set(1)=1; s2_set(3)=85; s3_set(4)=104

        % recover first indexed sector name
        path_lab{j,1}{1,1} = char(varargin{1}(i1));

        % recover the rest of the indexed sectors names
        for i=2:length(varargin)
            command = ['nextel = char(varargin{' num2str(i) '}(i' num2str(i)
'));'];
            eval(command);
            path_lab{j,1}{1,i} = nextel;
        end
end
```

APPENDIX E: Matlab function for sorting paths

File name: sorting.m

```
function [PATH_sorted] = sorting(
y,e_list,sets_list,Sectors_names,Pmax,last_tier )
% sorting.m reconverts the numerical name index into the real
% sector names and sorts the paths in ascending order.

% PATH_sorted is a 2 dimensional complex structure, lines corresponding to
% the paths. First column records emission values for each path. Second
% column records name of the sectors contributing to each path.

% PATH_em is a numeric array of emission values for all paths
% Path_lab is a cell array with contributing sectors to all paths

% Initialize the structure columns for all Pmax paths
PATH_em=zeros(Pmax,1);
PATH_lab=cell(Pmax,1);

% Start list of paths
sort_index=1;

% First path is unique and corresponds to the final demand placed on 1
% chosen sector; the emission value is e1.
PATH_em(1)=e_list{1};
PATH_lab{1,1}=Sectors_names(find(y));

% Get the paths in the order of the increasing number of component sectors
% Example: e2,e3,e4...

command_in = ' [path_em,path_lab] = getpath(e_list{' ;
command_int = '},sets_list{1';
command_fin = '});';
for t=2:last_tier
        % tiers corresponds to the paths of certain number of sectors
        % Example: t=4 refers to the paths with 4 contributing sectors: e4

        command_int = [command_int '},sets_list{' num2str(t)];

        % make sure it only search paths which exist
        % Example: if e2=0 there are no paths with 2 contributing sectors
        if   ~isempty(find(e_list{t}))

            command=[ command_in num2str(t) command_int  command_fin ];
            eval(command);
            % Example: [path_em,path_lab] = getpath(e3,s1_set,s2_set,s3_set)
            % returns: path_em=[em1,em2,em3,em4] and path_lab=[1 85 104;
            % 1 42 67; 1 30 79; 1 10 15 ] (see getpath.m) where path_lab
            % lines contain only indexed sectors names for the 4th path
            % existent with 3 contributing sectors each

            % add new paths of the same length in PATH_sorted structure
            % Example: if there is 1 path with one contributing sector and
            % 5 paths with two contributing sectors, the 4 paths with three
            % contributing sectors will be added from line 7 to line 11
            % first add emissions
            for pos_index=1:length(path_em)
                PATH_em(sort_index+pos_index)=path_em(pos_index);

                % second add original sector names (deducted from indexed
                % sectors names obtained with getpath function above)
                for k=1:length(path_lab{pos_index,:})
                    PATH_lab{sort_index+pos_index}{k}= ...
                        Sectors_names(str2double(path_lab{pos_index}{1,k}));
                end
            end

            % increment index for the next paths
            sort_index=sort_index+length(path_em);
        end
end

% Get the array of indices for sorting paths in descending order
[PATH_em_sorted,Im]=sort(PATH_em,'descend');

% Use the indices to sort the second column recording the names of the
% sectors contributing to each path.

PATH_lab_sorted=cell(Pmax,1);
for j=1:Pmax
    PATH_lab_sorted(j)=PATH_lab(Im(j));
end

% Transfer the numeric array of emissions in a corresponding cell array
% containing emissions
PATH_em_sorted=num2cell(PATH_em_sorted);

% Assemble structure
PATH_sorted=[PATH_em_sorted  PATH_lab_sorted];

end
```

APPENDIX F: Sectors abbreviation table

Abbreviated name	Original name
Agric	Agriculture
Manufc	Manufacturing
Transp	Transportation
Constr	Constructions
Air T.	Air transportation
Stamping	Automotive stampings
Steel mills	Blast furnaces and steel mills
Cement	Cement, hydraulic
Chem prep.	Chemicals and chemical preparations, n.e.c.
Equip. manuf.	Construction machinery and equipment
Raw oil	Crude petroleum and natural gas
Electricity	Electric services (utilities)
Metal parts	Fabricated structural metal
Equipm. parts	Industrial and commercial machinery and equipment, n.e.c.
Chemicals	Industrial inorganic and organic chemicals
Metals	Iron and steel foundries
Raw mat.	Lime
Oil_reused	Lubricating oils and greases
Minerals	Minerals, ground or treated
Motor parts	Motor vehicle parts and accessories
Motors	Motor vehicles and passenger car bodies
Equipment	Oil and gas field machinery and equipment
Drilling	Petroleum and natural gas well drilling
Refining	Petroleum refining
Rail T.	Railroads and related services
Steel parts	Steel wiredrawing and steel nails and spikes
Land T.	Trucking and courier services, except air
Water T.	Water transportation
Trade	Wholesale trade

Table 4.9 Sectors abbreviation table