Working Paper

Optimization-Based Analysis of a Simplified Ozone Model

Paweł M. Białoń

WP-96-134
November 1996

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Optimization-Based Analysis of a Simplified Ozone Model

Paweł M. Białoń

WP-96-134
November 1996

Working Papers are interim reports on work of the International Institute for Applied Systems Analysis and have received only limited review. Views or opinions expressed herein do not necessarily represent those of the Institute, its National Member Organizations, or other organizations supporting the work.

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Preface

This paper describes the results of research conducted by the author during the 1996 Young Scientists Summer Program at the International Institute for Applied Systems Analysis (IIASA). The research was done in the collaboration with the Methodology of Decision Analysis (MDA) and Transboundary Air Pollution (TAP) Projects. The TAP Project has been developing an ozone model which aims at analyzing various policy options that would result in a reduction of the tropospheric ozone concentration. Such a reduction can be achieved through reductions in emissions of two precursors: nitrogen oxides (NO_x: NO and NO_2) and volatile organic compounds (VOCs). One of the main reasons for developing and examining ozone models is to find cost-effective strategies for reducing ozone concentrations to levels below those assumed to be acceptable at various locations (grids).

Finding cost-effective strategies that lead to reductions in ozone concentrations requires formulating and solving sequences of nonlinear optimization problems that correspond to different assumptions adopted for specific policy options. Generating and solving sequences of such problems is a challenging task, that is not only interesting from both the methodological and software-engineering points of view, but also has practical applications.

The research reported in this paper aimed at developing a software tool to generate optimization problems that can be solved by several solvers. The tool is intended to support negotiations leading to an updated Nitrogen Oxides Reduction Protocol, being carried out within the framework of the United Nations Economic Commission for Europe (UNECE) Convention on Long-Range Transboundary Air Pollution (LRTAP).

This paper summarizes the methodological and engineering issues related to this task and documents the developed software. The ozone model specification is documented in two forms: in its original form and in the form of a mathematical programming problem that is actually solved by various solvers.

The length of the research period (three months in the YSSP and one month as a staff member of the TAP Project) was too short for the development of a final version of an optimization-based decision support tool. Although the tool is already useful for optimization-based problems analysis, several enhancements will likely be needed. Therefore, this paper describes the current stage of the ongoing research.

Abstract

This paper describes the development of a software optimization tool that aims at finding the optimal strategy for decreasing the ozone concentration in Europe.

A simplified ozone model has been used in the research. This model describes the relations between reductions of certain industry pollutants and the resulting ozone concentration abatement, as well as the costs associated with given emission reductions. The task of finding the most cost-effective strategy results in a difficult nonlinear problem comprising thousands of variables and constraints. Moreover, other difficulties can arise, such as issues connected with problem conditioning and solution sensivity to parameter changes. These potential difficulties lead to the question of software robustness. It was decided to apply three different optimization algorithms in order to ensure that the tool would be able to solve different optimization problems generated during different scenario analyses. The task of joining three different nonlinear solvers (CFSQP, Conopt, and MINOS) with completely different interfaces also turned out to be interesting from the software-engineering point of view. The object-oriented programming technique provided a simple way to divide the software parts into those that were common to all three solvers and those that were solver-specific. The model and the methodological and engineering difficulties in its analysis are summarized in this paper. The paper also contains a description of the software tool developed, as well as conclusions and further suggestions motivated by the first results.

Keywords: nonlinear optimization, optimization solvers, decision support, air pollution, robust

Contents

Optimization-Based Analysis of a Simplified Ozone Model

Paweł M. Białoń *

1 Introduction

Excessive concentration of tropospheric ozone has been recognized as an important air quality problem in Europe. It is possible to decrease the ozone concentration by reducing emissions of particular air pollutants. Once a model describing the ozone formation process was created, it became possible to develop a software optimization tool to assist policy negotiators in finding the optimal ozone reduction strategy.

A simplified ozone model describes the tropospheric ozone concentration in Europe. Because tropospheric ozone exposure causes damages to human health and vegetation, the established maximum ozone concentration levels should not be exceeded. The model describes the relationships between emissions of various pollutants [nitrogen oxides (NO_x), volatile organic compounds (VOCs)] and the resulting ozone concentration. A reduction in the ozone concentration can be achieved by reducing these emissions. The second part of the model describes the costs associated with reducing these emissions to certain levels. The research reported in this paper deals with an optimization-based approach to finding minimum-cost emission reductions that will bring the ozone concentration below levels specified for each grid. The total cost of the emission reduction constitutes the actual optimization goal (to be minimized).

The mathematical formulation of the simplified ozone model can be found in Heyes et al. [1]. Unlike previous models used for similar purposes (for example, RAINS - Regional Acidification INformation and Simulation), the simplified ozone model is nonlinear. Because each country in Europe is considered a separate pollution emitter and the ozone concentration is measured in a few hundred points, the dimensions of the problem become considerable. The model contains about two thousand variables and about two thousand constraints. With the introduction of nonlinearity, new difficulties arise.

A linear model with a few thousand variables can be considered middle-sized; a nonlinear model of the same size is already big. A precise analysis of the model suggests that the resulting optimization problem can have several solutions, not all of which may be acceptable. A nonlinear model of this size is almost certainly ill-conditioned, because there are no clear rules for scaling nonlinear models. It should be emphasized that this was the first time in the history of the cooperation between the MDA and TAP projects that a problem of this nature was undertaken.

A prototypical version of software for optimization (cf [8]) provided a good starting point for the approach described here. Use of the prototype has shown that numerical difficulties require that special attention be paid to software robustness.

*Participant in the Young Scientists Summer Program at the International Institute for Applied Systems Analysis (IIASA) in 1996. Home institute: Institute of Control and Computing Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland.

The tool user is interested in a multivariant analysis of his or her model and will not be satisfied with one optimization run. The software should be able to handle various optimization problems related to parameter changes. Thus, the software developer should not consider his or her product reliable after a few successfully performed tests, but should attempt to foresee future difficulties. One way to increase the robustness of the optimization procedure is to use several solvers, each with a different optimization method. Each solver may prove to be relevant for a particular problem; however, in nonlinear programming it is almost impossible to choose the most efficient method in advance.

Beside the approach described above, other known techniques have been used to tackle the difficulties encountered, for example, the regularization technique. Suitable data preprocessing has resulted in a notable simplification of the optimization problem.

Three different solvers were applied in the software package: **CFSQP** (C Code for Feasible Sequential Quadratic Programming; University of Maryland, MD, USA), **Conopt** (ARKI Consulting and Development, Denmark), and **MINOS** (Stanford University, CA, USA). It is planned to eventually also apply a fourth nonlinear solver, **DIDASN++**, developed at the *Institute of Automatic Control at the Warsaw University of Technology.*

The task of using several solvers is interesting from the software-engineering point of view. Each solver has a different interface (the way of formulating an optimization problem). A considerable effort was made to isolate those parts of the software that were common to all the solvers. The object-oriented programming approach has simplified this task, because it allows for handling common parts in base classes and provides solver-specific interfaces through inherited classes. Therefore, the software was developed in the C++ programming language. Since the solvers were available only for the Unix operating system, the software now runs under Unix only. However, the developed software is written in standard C++; therefore, porting it to another platform should be easy.

Another problem that required considerable attention was how to handle the data. The data that define the model come from different sources (e.g., databases developed in a PC environment and results of statistical analyses performed in a Unix environment). Therefore, it was necessary to combine the data into a form that is easy to maintain and use. For this purpose the Hierarchical Data Format (HDF) library was used. HDF is a public domain library that supports handling data in an efficient binary format; it is also portable between several hardware and software platforms.

The software for optimizing emission reductions is now ready and works with a full data set. Because it is a very new product, some modifications will certainly have to be introduced. One possible extension might be to apply multi-criteria model analysis. Currently, the problem analysis is based on a single-criterion optimization. It would be greatly enhanced by application of multi-criteria model analysis, because in fact we deal with two conflicting goals, namely, to minimize the tropospheric ozone concentration and to minimize the cost of such an operation.

This paper is organized in the following way. Section 2 describes the ozone model and the preliminary optimization problem derived from it. Section 3 provides an analysis of the problem's optimization properties and the necessary modifications introduced during the conversion of the original problem formulation into a mathematical programming problem formulation. Sections 4 and 5 contain respectfully a description of the software structure and the software usage. The first results are presented in Section 6, together with some conclusions and suggestions for further development.

2 Problem Overview

In this section the simplified ozone model in the form obtained from modelers is described and the resulting optimization problem is outlined. The simplified ozone model is described in Heyes et al. [1]. See Heyes and Schöp [4] for a description of the origin of the model. The formulation described here differs from the final mathematical programming problem formulation because the final formulation is adjusted to account for various methodological, numerical, and software demands. The transformations performed and the final formulation will be described in subsequent sections. The final problem formulation takes into account various modifications described in this paper, and is given in Appendices A and B.

2.1 Subject

The structure of the ozone model provided by the modelers is shown in Figure 1.

The model consists of two parts. The first part describes the dependencies between the tropospheric ozone concentration and the emissions of certain types of pollutants. Currently, two types of pollutants are distinguished, namely, nitrogen oxygens (NO_x) and volatile organic compounds (VOCs). The model describes a complex set of chemical reactions. Emissions of each type of pollutant are measured in approximately 40 geographical areas in Europe, called emitters. ozone concentrations are measured in approximately 600 geographical grids, called receptors.

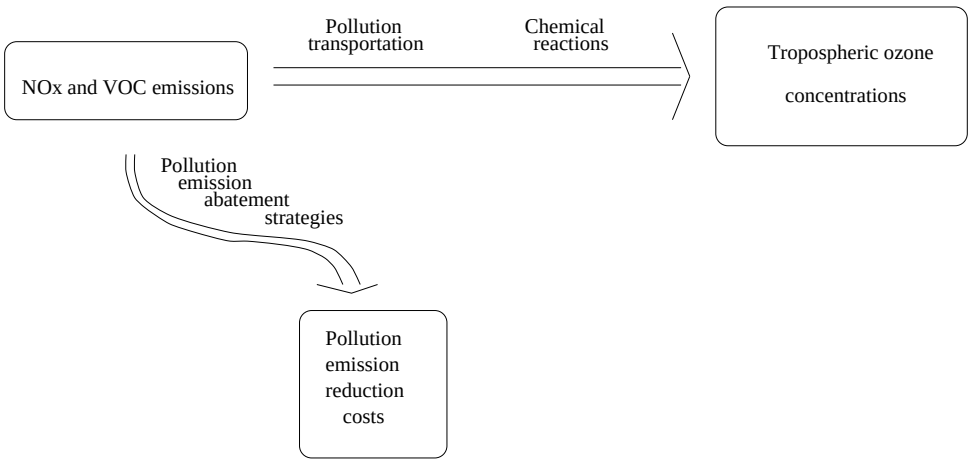

Figure 1: Ozone model structure

Pollution emission in any given emitter influences the ozone concentrations in several receptors. The model takes into account the effects of atmospheric transportation of pollution. Because ozone exposure is undesirable, we aim at decreasing ozone concentrations, which implies that, in general, we should try to decrease the pollutant emissions.
The emissions of particular emitters can be decreased by applying cost-reduction technologies, for example in various areas of industry. Any reduction implies the costs of introducing the associated technology. This dependency is described by the second part of the model. Clearly, we should try to reduce of the emission abatement costs.

The problem of reducting the ozone concentration can be formulated as the process of finding levels of emission reductions that can be achieved at minimal costs and that

decreases the ozone concentrations in particular grids below specified levels.

2.2 Preliminary Model Definition

Let indices $i \in I$, $j \in J$ correspond to emitters and receptors, respectively. The numbers of elements in I and J correspond, respectively, to the number of countries (about 40) and the number of grids (about 600).

The following decision variables are defined:

n_i - annual emission of NO_x at the ith emitter; and

v_i - annual emission of VOCs at the jth emitter.

Each of the decision variables defined for $i \in I$ is implicitly bounded by a corresponding domain of piece-wise-linear (PWL) function that defines costs associated with the emission reduction.

The following state variables are in the model:

en_j - effective emissions of NO_x experienced at the jth receptor;

ev_j - a representation of another nonlinear term (an alternative formulation to using effective emissions of VOCs) experienced at the j-th receptor; and

o_j - resulting ozone concentration at the jth receptor.

State variables are defined for each receptor ($j \in J$). The formulas used for evaluating en_j and ev_j are given in the form of *constraints*:

$$en_j = \sum_{i \in I} e_{ij} n_i + enn_j, \tag{1}$$

$$ev_j = \sum_{i \in I} d_{ij} v_i, \tag{2}$$

where enn_j are the given effective natural emissions of NO_x.

Equation (1) expresses the pollution transportation effects. Equation (2) cannot be interpreted in such a way; it simply defines the nonlinear term ev_j in the statistical model.

The ozone concentration in the jth receptor is given by

$$o_j = k_j + \sum_{i \in I} (a_{ij} v_i + b_{ij} n_i + \gamma_{ij} n_i^2) + \alpha_j en_j^2 + \beta_j en_j ev_j \tag{3}$$

The following parameters appear in the model definition:

- Transfer coefficients: a_{ij}, b_{ij}, e_{ij}, d_{ij};
- Effective natural emissions of NO_x: enn_j
- Background ozone concentrations: k_j;
- Parameters of nonlinear terms: α_j, β_j, γ_{ij}; and
- Five parameters of each function plus values of minimum and maximum emissions that define the domain of each function (see 3.5).

The following two sets of functions define for each emitter the annual costs related to reducing corresponding emissions to a certain level:

$cn_i(n_i)$ - annual cost related to reducing the level of NO_x emission to n_i; and

$cv_i(v_i)$ - annual cost related to reducing the level of VOC emission to v_i.

It is assumed that each cost function is convex and strictly monotonic. Originally, the functions were defined as PWL functions. Each PWL function was defined by a set of ordered pairs (interpreted as points), each pair (point) denoted by (y, c(y)), where y is the emission level (of NO_x or VOCs) and c(y) is the cost of reducing a corresponding emission to that level. Each point corresponds to a vertex of a PWL function.

Alternative definitions of cost functions by smooth functions are possible and are currently used, as is described later.

2.3 Multi-Criteria Character of the Optimization Problem

As already mentioned, ozone exposure is undesirable; thus, the ozone concentration should be kept low. This implies that, in general, we should try to decrease the pollutant emissions. It is also clear that we should aspire to reduce the costs of emission abatement. However, achieving this goal implies an increase in ozone concentrations, which is contradictory to our first goal. The resulting optimization problem can be defined as a *multi-criteria problem*. If we change the values of decision variables in order to improve one goal, we might cause a deterioration of another. In single-criterion optimization problems, all potential decisions are assigned a single scalar value (namely, the corresponding value of the goal function). Therefore, choosing the optimal decision (optimal solution) is a well-defined task: the optimal solution is the solution that gives the maximum value of the goal function. In multi-criteria problems each decision is assigned several values, because several *criteria* (goal functions) are defined. In such a case, however, ranking potential decisions becomes a nontrivial task. Several solutions to the problem can exist, that, without any additional assuptions, are not comparable.[1]

There are several techniques for dealing with multi-criteria optimization problems. Multi-criteria optimization must help the user choose between solutions that are not comparable.

In one of such methods, the *aspiration based approach*, the user specifies so-called aspiration and reservation levels for each criterion. The user tries to prevent the criteria values from deteriorating to below the corresponding reservation levels and does not necessarily want to improve them beyond the corresponding aspiration levels.

However, these constraints can be treated only as so-called soft constraints; in other words, they may be violated. This approach also allows for uniform treatment of all criteria by defining so-called criterion achievement functions with respect to the defined aspiration and reservation levels. This technique is described by Wierzbicki [7].

2.4 Preliminary Goal Formulation

In the current approach, it was decided to apply a single-criterion optimization. Although multi-criteria approach should be considered plausible for the next stage of tool

[1]For example, one solution can have higher total costs, but lower ozone concentration(s), whereas for the second solution the situation is reversed. These solutions are not comparable in the sense of partial Pareto order.

development, it seemed reasonable to begin with a precise investigation of the numerical properties of the single-criterion problem. The large number of potential criteria (e.g. 600 variables o_j) also requires that a choice concerning the criteria aggregation scheme be made.

There is sufficient basis to establish upper limits for ozone concentrations in particular receptors. The upper limits can be obtained from an analysis of damage caused by exposure to ozone in certain areas, for example in forests.

Hence, we can try to find minimum-cost emission reductions that will result in ozone concentrations below specified levels. Mathematically, we could request a reduction in emission abatement costs subject to the limits on ozone concentration.

The total abatement cost was chosen for the goal function (to be minimized). The total abatement cost means the sum over all emiters and over both pollutants. Hence, the goal function takes the following form:

$$\sum_{i \in I} (cn_i(n_i) + cv_i(v_i)). \tag{4}$$

The goal function should be minimized.

An additional set of constraints for ozone concentrations is defined by

$$o_j \leq o_j^{max}, \tag{5}$$

where o_j^{max} is a given maximum ozone concentration at the jth receptor.

3 Optimization-Directed Analysis of the Problem

This section provides a discussion of the features of the problem that might be of interest to an optimizer. It also presents the method of solving the problem that was chosen for this project. The few necessary changes made to the problem to improve certain characteristics are also described.

3.1 Basic Features

There are 38 emitters in the current formulation of the model, which is more or less equal to the number of countries, and the model comprises 598 receptors. Therefore, we deal with a problem with 1,271 variables and 1,793 constraints. Whereas a linear model of these dimensions would be considered average-sized, a nonlinear model of this size is large. Because the model was obtained through statistical methods, it was possible to make its structure sparse. The problem was recalculated, with the additional demand that some small coefficients be equal to zero. For example, only about 16% of all coefficients a_{ij} are non-zero (some additional sparsity is certainly implied by the problem structure).

There was no simple way to judge the conditioning of the problem in advance, as the model was nonlinear. Experience shows that nonlinear models of this size are almost certainly ill-conditioned.

3.2 Basic Idea of Solving

Concern about the numerical properties of the problem made it necessary to undertake some special actions to ensure the software robustness and the software's ability to solve the problem for various sets of data. Initially, it was assumed that the user would be

able to specify many different sets of input data. The tool was intended to be used in policy negotiations and thus the need to consider different scenarios could be expected. For example, values of upper limits on ozone concentrations could be changed frequently. Of utmost importance was developing a robust tool capable of providing reliable results.

It was decided to apply a number of different available solver programs to the same optimization task. By using several solvers, we increased the chance that the problem could be solved at all (by at least one of them) in a reasonably short time. Any one of the solvers could prove relevant for a particular task, thought it is difficult to predict which one.

Furthermore, solutions obtained from different solvers could be compared to prove their correctness.

The following section gives an overview of the solvers used.

3.3 Solvers

Three solvers were applied : CFSQP (C Code for Feasible Sequential Quadratic Programming, University of Maryland, MD, USA), Conopt (ARKI Consulting and Development, Denmark), and MINOS (Stanford University, CA, USA). It was planned to also apply a fourth nonlinear solver, DIDASN++, developed at the Institute of Automatic Control at the Warsaw University of Technology. Though it possesses some interesting and promising properties (the shifting of the penalty functions, a special model structure representation), its use was skipped for the time being due to secondary technical reasons.[2] Users interested in a more detailed solvers description can consult Lawrence et al. [5] - for a CFSQP description, Drud[2] -for a Conopt description, Murtagh and Saunders [6] - for a MINOS description, and Wierzbicki et al. [3] for the DIDASN++ algorithm.

The solvers applied use different algorithms, but share one important feature: they all take advantage of the existence of a large linear part of the problem. All three solvers are based on sequential solving of linearizations of the problem (or quadratic approximations - CFSQP).

MINOS and Conopt base on the projected gradient technique. In this technique, a projection of the objective function gradient on the subset of active constraints is used to determine the search direction. As some constraints are not linear, they may be violated after the directional search. Conopt uses special corrections to follow the constraints exactly. MINOS uses the augmented Lagrangian function to preclude too large a constraint violation. DIDASN++ uses the augmented lagrangian to apply the shifted penalty technique. CFSQP uses the sequential quadratic programming technique which belongs to the class of trust region methods.

All the solvers require user-written subroutines that calculate values of the goal function and the left-hand-sides of constraints, and the first derivatives of these values. DIDASN++ is capable of computing the the values and the derivatives automatically, based on a model description in a special format.

3.4 Regularization

As mentioned above, the tool was designed to perform a series of experiments. For different reduction policy options, different optimization runs are to be performed with

[2] DIDASN++ can handle nested dependencies in a special way. It allows for the elimination of intermediate model variables and thus some linear constraints. This advantage, however, made the programming interface of DIDASN++ completely different from the interfaces of the other solvers.

different sets of parameters. However, experience shows that even small changes in an optimization problem may result in completely different solutions (and very different values for the decision variables). Even if a problem is linear and is not varied, we can find several solutions with the same value for the goal function. Although such behavior can be explained mathematically, it usually is not acceptable to a user. In particular, it is undesirable in a negotiation-support situation. It makes influencing the decision by changing parameters difficult. To avoid these phenomena, we could implement a method that keeps each new solution close to the previous one.

The optimization practice asserts a good way to keep the solution near a certain point (called the *reference point*): regularization. Regularization usually consists in adding a penalty term to the goal function. This penalty term is equal to to the distance between the current point and the reference point. The distance is usually expressed in the space of selected variables (in our case, preferably the decision variables n_i and v_i)

Another reason for using regularization is following: In a rough description of the problem we can assume the increasing dependency of any ozone concentration on any pollution. Let us, however, consider a hypothetical one-dimensional case with only one emitter and one receptor. In the case of a particularly high NO_x concentration, the number of OH radicals decreases together with NO_x concentration growth. Because there are fewer OH radicals to react with VOCs, ozone formation becomes less intensive.

This is a fact of notable importance. In some situations we can achieve a reduction in the ozone concentration by increasing the NO_x emission. Clearly, this action leads to an abatement of NO_x reduction costs, because the emission increases.

For the one-dimensional case we could find a simple remedy: we might establish a suitable upper bound for the NO_x emission. However, this method would be more difficult to use in for the multi-dimensional case. In reality, each ozone concentration o_j is a complex function of vectors,

$$o_j(n, v), \tag{6}$$

created by combining equations (1),(2) and (3). (n and v denote vectors consisting of all n_i and v_i.)

Because this phenomenon appears only if the NO_x concentration is high enough and the search for a solution follows a specific path, it does not necessarily have to appear in real optimization runs. In fact, during several experiments, the solutions of the same problems given by different solvers were the same; indicating that a proper solution was probably obtained.

However, a simple method of preventing the likely difficulties may consists in selecting a suitable starting point. In the case of multiple local minima, the solution largely depends on the choice of the starting point. The user can set different starting points. The minimal attainable emissions point is a good candidate for the starting point. Let us define the starting point in the space of decision variables n_i and v_i. The minimal emissions point is composed of the lower bounds of n_i and v_i. (The point corresponds to the minimal NO_x emission in our one-dimensional case.) With such a starting point, we decrease the probability of jumping to the previously described "degenerated" region.[3] If the steps taken during an optimization run are large, reaching the degenerated region can happen relatively easily. Thus, another regularization term should be added to the goal function to prevent points from the subsequent iterates from differing from each other too much.

[3]This scenario corresponds to jumping to another side of the diagram of the function $o(n)$ in the one-dimensional case described above.

In light of the problems outlined above, the goal function must be modified by adding additional penalty terms:

$$\sum_{i \in I}(cn_i(n_i) + cv_i(v_i)) + \epsilon(\|n - \bar{n}\| + \|v - \bar{v}\|) + \eta(\|n - n^{k-1}\| + \|v - v^{k-1}\|), \qquad (7)$$

where ϵ is a given small positive number and η is a positive number. The first term corresponds to the cost of emission reductions and is the original goal function. The term $\epsilon(\|n-\bar{n}\|+\|v-\bar{v}\|)$ is the regularizing term introduced to keep the optimal solution close to the point defined by given reference vectors $\bar{n}$ and $\bar{v}$. The term $\eta(\|n - n^{k-1}\| + \|v - v^{k-1}\|)$ (where k is the iteration index) will be added in the future as a regularizing term introduced to find a local minimum that is close to a given starting point.

Another benefit of introducing regularization consists in improving numerical properties of the problem. Penalty terms are frequently good-shaped quadratic functions. Thus, adding penalty terms to the goal function can improve the shape of the goal function diagram (it can lead to a reduction of the disproportion between the lowest and the highest eigenvalue of the goal function hessian.)

3.5 Cost Functions

Originally, the functions were defined as PWL functions. Each PWL function was defined by a set of ordered pairs (y, c(y)), interpreted as points; y represents the emission level (of NO_x or VOCs) and c(y) represents the cost of reducing the corresponding emission to that level. Each point corresponds to a vertex of a PWL function.

There are two main drawbacks to using such functions. First, applying PWL functions necessitates the generation of additional variables and constraints and causes the problem size to grow. There are standard methods for converting convex PWL functions (which are components of the goal function) to a set of dummy variables and constraints. The goal function becomes a sum of a certain number of dummy variables (the number of dummy variables equals the number of original cost functions, i.e., the number of goal function components). For each dummy variable, as many linear inequality constraints are defined as many pieces there were in the original PWL function. Because the PWL functions consisted of a rather large number of pieces (about 20), this approach described had the potential to increase the problem dimensionality significantly.

The second drawback is related to the slopes of the pieces of the PWL functions. In the data provided, some neighboring pieces of the PWL functions had very similar slopes. This was a potential source of numerical problems. Normally, to avoid singularities, one should avoid generating constraints that are almost linearly dependent.

The new idea was to represent goal functions based on smooth nonlinear approximation. The following form was chosen for representing cost functions:

$$cx(x) = \frac{a + bx}{1 + cx + dx^2} + e, \qquad (8)$$

where cx denotes either one of the cn_i or one of the cv_i functions and x denotes the corresponding n_i or v_i. Additionally, a lower bound x^{min} and an upper bound x^{max} are defined for the argument x of each function $cx(x)$. They imply the domain of each smooth function. This information is used to derive bounds for NO_x and VOC emissions.

3.6 Scaling

There are no clear rules for scaling of nonlinear problems. In most cases, the scaling must be done intuitively. Fortunately, it was possible to assess the ranges inside which variables could vary. The rough idea applied in this approach can be called the "0.01-100" rule: variables of the rescaled problem should vary in the approximate range of $[0.01-100]$. This demarcation is mainly motivated by the values of various stopping parameters of solvers, which should be several orders of magnitude less than values of variables. The scaling consists in multiplying the variables, coefficients, goal functions, and right-hand-sides of the problem by appropriate scaling coefficients. For a consistently scaled problem, after an optimization of the scaled problem has been finished, it should be possible to derive needed values by dividing the values of variables (or other objects) by the same scaling coefficients. For some variables, the rule is not applied, because of the the negativity of the variables' lower bounds or because of ranges that were too large. The cost functions can be scaled so that the "0.01-100" rule is (almost) satisfied by either the entire goal function or by each of its components. The second approach was motivated by the willingness to keep the jacobian elements values in reasonable ranges. (The jacobian values depend on the scaling of particular components rather then scaling of the entire goal function). A user can influence scaling by adding additional parameters in the specification file (see Appendix C).

A more precise description of scaling can be found in Appendices A and B.

Some solvers turned out to be very sensitive to changes in the scaling coefficients. For example, a tenfold decrease in the scaling coeficients for cost functions caused the computation time to grow from 296 seconds to 8734 seconds of CPU time (for a full-sized data set). MINOS had some problems calculating the inverse Hessian matrix, because its elements were too large.

The experience gained during scaling and reformulating of cost curves, among some other activities, illustrates the importance of data preprocessing in optimization. Preprocessing is often critical if the optimizer is to achieve satisfactory results.

4 Software Overview

The development of the software was a difficult task from the software-engineering point of view. It was decided to write a C++ program using the three selected solvers.[4] The Unix operating system (Solaris) was chosen as the working environment with the SPARC C++ compiler.

All of the applied solvers require a problem definition from the user. The model structure must be provided at the beginning of an optimization run. It consists of the character of the variables and constraints (nonlinear, linear, equality, inequality), some coefficients defining the linear part of the model, and some specific control parameters. During an optimization run the solvers demand information about the goal function value, constraint (left-hand-side) values, and the corresponding derivative information. The values of first derivatives of the goal function and constraints must be provided.

It should be obvious that the interfaces necessary for exchanging the information are completely different for each solver. A few main differences can be listed:

[4]The idea of using existing modeling languages was rejected in order to gain more freedom in both handling data defining the model and in introducing enhancements. Applying a multi-criteria approach would require the development and linking of a special graphical user interface.

- Different sequences of variables (linear, nonlinear,[5] equality, inequality)/constraints
- Different calling/argument-passing conventions
- Different representations of the Jacobian matrix (dense/sparse storage, storage by rows/by columns)
- Treating of linear part of the model (some solvers exclude the linear term from constraint/goal values and derivative values)
- Existence of the MPS format file defining the linear part of the model
- Different programming languages of the solvers (C, Fortran)

Users interested in the details of the solver interfaces should see Lawrence et al. [5] for a description of CFSQP, Drud[2] for a description of Conopt, and Murtagh and Sounders [6] - for a description of MINOS.

Despite the differences in interfaces, as many parts of the software as possible remain common (solver-independent); this applies, for example, to data structures storing the model. The object-oriented programming technique proved helpful in this design.

The input data originally came from different sources and had different formats. The different input files were replaced by one file capable of including all model coefficients; thus, difficulties during the data reading phase and data inconsistencies were avoided. The Hierarchical Data format (HDF) was used for creating such a file. HDF, a portable, public domain scientific data format, is well documented and supported by National Center for Supercomputing Applications (NCSA) at the University of Illinois at Champaign-Urbana, USA. The HDF library permits the storage and retrieval of various types of objects (scalars, vectors, sparse vectors, matrixes, etc.). The objects are identified by unique character strings.

A C++ library has been created to serve as an interface between HDF and the main program data structures.

An actual optimization run is performed by one program, which is linked with one of the solver libraries (solvers are compiled into the library form). The structure of such a program consists of a number of modules, listed here approximately according to the flow of information between a solver and data files:

- The direct interface routines, written in the C language;
- The C++ interface class O3 (and derived classes O3_conopt, O3_cfsqp, O3_minos);
- The calculating module, which performs the calculations of goal/constraint values and derivatives (i.e., implements the mathematical formulas). Classes: O3_goal, O3_jac. The calculations of the cost function values are performed in a separate module (class O3_cost and derived O3_smooth and O3_pwl);
- The problem container (class O3_nlp), which has a reference to the model holder and defines some additional objects (bounds, initial point, etc.);
- The model container (class O3_data), which stores the model coefficients, reads actual data, and generates test data.

Instead of using actual data, a small sample problem can be generated to test the software.

[5] The term "nonlinear variable" denotes a variable involved in at least one nonlinear constraint or in a nonlinear goal function

5 Usage Instructions

5.1 Data Preparation

The main programs and the main files are installed in one directory, currently $(HOME)/Ozone/prog.

The HDF file "data.hdf" should be prepared, containing data with the object names listed in Section A.6. Currently, the program "reader" is invoked from the main directory, which generates the HDF file based on information read from ASCII files. Eventually, however, a more direct way to generate the HDF file will be possible. The program "reader" takes no parameters:

```
reader
```

This program can be produced simply by typing

```
make reader
```

5.2 Specification File

The specification file "_spc.o3" contains some control parameters. Each line of this file contains one parameter specification. The parameters are listed in Appendix C. The specification file can be edited by any text editor.

5.3 Main Program

The main program reads the HDF file and the specification file, performs an optimization run, produces some debug information, and stores results. The main program uses only one of three solvers at any given time. Thus, it can have names cfsqp, conopt or minos, depending on the solver library with which it has been linked.

Let us consider a Conopt version of the main file. To produce it, we invoke the following command in the main directory:

```
make conopt
```

The program takes no arguments:

```
conopt
```

The remaining solvers can be generated and used in a similar way.

5.4 Main Program Output

The main program produces a report from all the stages of the work. The report form can be controlled using settings in the specification file. The report is printed to the standard output (screen) and has a descriptive form that is easy to understand. The report can contain

- A listing of control options (e.g. read from the specification file);
- A dump of model data read from an HDF file;

- Scaling information;
- A dump of some values calculated during the problem-defining phase, such as bounds, starting point, and reference point. Ozone concentrations for the initial point are also given;
- Trace of calculating goal/constraint values and derivatives;
- A dump of different values (goal value, variables, constraint activities, etc.) for the solution;
- Information on the program execution path (chosen execution option);
- Status and error messages.

The solution is also stored in the file "_solution" (the name of this file can be changed in the specification file).

It should be kept in mind that each of the solvers used produces its own output. This output can be assigned to standard output or can be directed to files. Details concerning output can be found in the solvers' documentation.

6 Results, Conclusions, and Suggestions

In the tests performed, the software tool proved to be able to solve the problems described in this paper. The author's goal was to provide a tool capable of finding the optimal strategy for reducing the ozone concentration in Europe. However, it has not yet been used in negotiations. The problems inputing to the tool were designed to test the software rather than to provide any real advice regarding pollution abatement strategies. Thus, though most of the data were real, some parameters were artificially set (e.g., upper limits on ozone concentrations). This fact can disturb some real-world interpretations of the results, but does not change the numerical properties of the problem.

The following satisfactory results can be summarized:

- Two solvers (Conopt and MINOS) were able to solve the problem given to them.
- The results given by the two solvers that solved the full-sized problem were the same.
- All three solvers were able to solve a small artificial testing problem; they gave the same results.
- The full-sized problem was solved within a few minutes by two of the solvers.

The full-sized problem used in the experiment consisted of 1,271 variables and 1,793 constraints (38 emitters and 598 receptors). The problem was solved on a Sun SPARCserver-1000. Default settings were used in the experiment. The solving times in seconds are given in Table 1.

The CFSQP solver seemed to converge, since some indicators (e.g., the Kuhn-Tucker norm) were significantly decreasing. However, this solver did not finish the experiment in a reasonable time (after 10 days the solution was not found). Nonetheless, the fact, that the solver works properly on the small test problem is promising. Intensive disk swapping was also observed during the CFSQP work, which could slow the work down

Solver	*Computation time*
Conopt	159
MINOS	368
CFSQP	(converging)

Table 1: Optimization times

significantly. A possible solution wold be to try to run this solver on a computer system with extensive real memory (a supercomputer system). The behavior of CFSQP is still being investigated.

In general, the results can be described as promising. At least two solvers were able to give a solution and the solutions passed several tests for errors. The optimization time is short; thus, the tool may be used in a multivariant analysis of the problem (optimization runs for different problems may be performed during policy negotiations).

Several observations and suggestions for the further development of the tool can be formulated:

- The technique basing on solving the same problem using different methods has shown some advantages. It helped in validating results and perhaps also helped to avoid a defeat, which might have been the judgement had only CFSQP been applied.
- The success of an optimizer greatly depends on data preprocessing.
- Compared with a single-criterion approach, a multi-criteria technique can provide a better analysis of the problem, which clearly contains more than one criterion.

7 Acknowledgments

The work described in this paper was carried out in collaboration with the Methodology of Analysis (MDA) and the Transboundary Air Pollution (TAP) Projects at IIASA. The author was a staff member of the TAP Project for one month and a participant of the YSSP Program at IIASA during the Summer of 1996. A number of institutions and persons helped make this research possible.

The author would like to thank the Polish Committee for IIASA for providing funding for his participation in the 1996 YSSP.

The simplified ozone model is being developed by the TAP Project. Colleagues from the TAP Project helped familiarize author the model and its background. All data used for the model were prepared by colleagues from the TAP Project, who also provided cost functions in the form of smooth functions. Therefore, the author would like to thank Markus Amann, Imrich Bertok, Janusz Cofala, Serguei Chibaev, Frantisek Gyarfas, Christopher Heyes, Zbigniew Klimont, and Wolfgang Schöpp for their help and cooperation. Thanks also go to IJsbrand Haagsma, a YSSP participant, for his introduction to the HDF library and for his collaboration in the development of software for data handling.

Two solvers used for this study were provided free of charge by their developers:

- CFSQP - C Code for Feasible Sequential Quadratic Programming, from the University of Maryland, MD, USA; and
- Conopt - a System for Large Scale Nonlinear Optimization, developed by ARKI Consulting and Development Company, Denmark.

The author would like to thank Dr. Arne S. Drud of ARKI, not only for making the Conopt library available for the research, but also for his visit to IIASA and for consultation concerning the formulation of the optimization problem.

The prototype version of the optimization software was developed in 1995 by Piotr L. Zawicki of the Warsaw University of Technology. The results of that study were very helpful for the the work reported here.

Finally, the author would like to thank Marek Makowski, who developed the formulation of the optimization problem, and designed and implemented a large part of the software. The author thanks him for many discussions and the advice concerning various scientific, engineering, and formal problems.

References

[1] M. Amann, C. Heyes, and W. Schöpp. A simplified model to describe the ozone formation process in Europe. Technical report, International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria, 1995.

[2] A. S. Drud. Conopt: A system for large scale nonlinear optimization. Reference Manual. ARKI Consulting and Development A/S, Bagsvaerdvej 246 A 2880 Bagsvaerd, Denmark, 1995.

[3] J. Granat, T. Kreglewski, and A. P. Wierzbicki. IAC-DIDAS-N: A dynamic interactive decision analysis and support system for multicriteria analysis of nonlinear models v. 4.0. Collaborative Paper CP-91-010, International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria, 1991.

[4] C. Heyes and W. Schöpp. Towards a simplified model to describe ozone formation in Europe. Working Paper WP-95-34, International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria, 1995.

[5] C. Lawrence, J. L. Zhou, and A. L. Tits. Users's guide for CFSQP version 2.2. Technical Report TR-94-16r1, University of Maryland, College Park, MD 20742, USA, 1994.

[6] B. A. Murtagh and M. A. Saunders. MINOS 5.1 user's guide. Technical Report SOL 83-20R, Stanford University, Stanford, CA 94305-4022, USA, 1987.

[7] A.P. Wierzbicki. Multi-objective modeling for engineering applications in decision support. In *Proceedings of the Twelth International Conference on Multiple Criteria Decision Making Lecture Notes in Economics and Mathematical Systems.* Springer Verlag, Berlin, 1996.

[8] P. L. Zawicki. Software tools for generation, simulation and optimization of The Simplified Ozone Model. Working Paper WP-95-107, International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria, 1995.

A Model Definition

This section gives the description of the model used by the software tool. It also summarizes some demands regarding the data.

A.1 Notation

Indices $i \in I$, $j \in J$ correspond to emitters and receptors, respectively. The numbers of elements in I and J correspond to the number of countries (about 40) and the number of grids (about 600).

A.2 Decision Variables

n_i - annual emission of NO_x

v_i - annual emission of VOCs

Each of the decision variables defined for $i \in I$ is implicitly bounded by a corresponding domain of the cost function that defines costs associated with a reduction of emissions (see Section A.6).

A.3 State Variables

en_j - effective emissions of NO_x experienced at the jth receptor;

ev_j - a representation of another nonlinear term (alternative formulation to using effective emissions of VOCs experienced) at the jth receptor;

o_j - resulting ozone concentrations.

State variables are defined for each receptor ($j \in J$). The o_j variables are defined only in the model description (they will not appear in the model actually generated for optimization). However, a function for computing o_j is implemented.

A.4 Constraints

The mean effective emissions (of, respectively, NO_x and VOCs) experienced at the jth receptor are given by

$$en_j = \sum_{i \in I} e_{ij} n_i + enn_j, \tag{9}$$

$$ev_j = \sum_{i \in I} d_{ij} v_i, \tag{10}$$

where enn_j are the given effective natural emissions of NO_x.

The ozone concentration is defined by

$$o_j = k_j + \sum_{i \in I} (a_{ij} v_i + b_{ij} n_i + \gamma_{ij} n_i^2) + \alpha_j en_j^2 + \beta_j en_j ev_j \tag{11}$$

and it is constrained at each receptor by:

$$o_j \leq o_j^{max}, \tag{12}$$

where o_j^{max} is a given maximum ozone concentration at the jth receptor.[6]

A.5 Goal Function

See Section 3.4.

A.6 Parameters

The following parameters are needed for the problem definition:

- Transfer coefficients: a_{ij}, b_{ij}, e_{ij}, d_{ij};
- Effective natural emissions of NO_x: enn_j;
- Background concentration in equation (11): k_j;
- Parameters of nonlinear terms: α_j, β_j, γ_{ij};
- Points defining PWL cost functions: $cn_i(n_i), cv_i(v_i)$;
- Parameters defining smooth cost functions: $cn_i(n_i), cv_i(v_i)$: five parameters of each function plus values of minimum and maximum emissions (that define the domain of each function); see also Section 3.5;
- Maximum ozone concentrations: o_j^{max};
- Reference point for decision variables: $\bar{n}$, $\bar{v}$;
- Starting point: n_0, v_0;
- Values of emissions (en, n, v) for 1990 (denoted by en_{90}, n_{90}, v_{90}). Values of ev_{90} are not available;
- String identifiers for emitters and receptors, em_id_i, rec_id_j;
- Feasibility tolerance for ozone, denoted by o^{feas}. The tolerance can be set by a directive in the specification file, its default value is 1. It is used for receptors, for which minimum concentrations would result in a violation of the original constraint; see definition (20);
- Regularization parameters: ϵ and parameters defining a sequence η_k.

Additionally, values of o_{90} are provided (they will be used for checking the computations of o_{90} values, provided that enn are also provided for 1990).

[6] β_j was added to the model formulation in order to allow for scaling. Therefore, it is assumed that (before scaling) all $\beta_j = 1$.

A.7 Assumptions and Requirements

The following implementation assumptions are assumed for the prototype version developed during this research:

- The optimization will be run on Sun in a batch mode.
- Selection of options will be defined in the specification file.

A.7.1 Assumptions about the data

The data provided for the problem generation should fulfill the following requirements. If any of the requirements is not met the preprocessing will terminate with a fatal error (and thus the application will be terminated):

- All data listed in Section A.6 must be provided in the same units. Currently, data are split into several files but for a final implementation all data should be stored in one file, preferably using HDF.
- All "nonsignificant" transport coefficients should be filtered out. Additionally, any coefficient in the data file with a value smaller than 10^{-8} will be reset to zero during the solver's preprocessing.
- Cost functions are smooth, strictly convex and strictly decreasing. Definitions of these functions must contain minimum and maximum emissions (that define the domain of each function). The units of emissions should be the same as those used for calculations of other coefficients (currently, emissions are normalized using 1990 emissions). The osts units should be the same for both NO_x and VOCs, and should be scaled in such a way that the maximum cost will be around 10.
- There are no empty variables/constraints (i.e., each emitter has at least one nonzero coefficient to one receptor, and there is at least one such coefficient for each receptor). This implies that in each row of matrices e and d there is at least one positive coefficient.
- All components of e and enn are nonnegative.
- In the current set of data, some elements of d are negative.

The following data are *optional*:

- Reference point for emissions (minimum emissions will be assumed).
- Starting point for emissions (minimum emissions will be assumed).

A.7.2 Preprocessing by functions linked to solvers

There are a number of requirements (listed below) that should be met in order to avoid numerical problems and obtain sensible solutions. These requirements will be met during the generation of the optimization problem defined above and will be redefined as a mathematical programming problem in Appendix B.

- Unfortunately nonlinear solvers do not provide scaling. Therefore, the problem should be scaled so that the absolute values of variables will be in the range [0.01, 100]. The outline of the scaling to be implemented is given in Appendix B.5.
- Bounds for all variables are defined in the way specified in Section B.1.
- To avoid an infeasible solution, there is an option to relax the constraints for the ozone concentrations in receptors for which the corresponding constraints would be violated for minimum emissions.

B Mathematical Programming Problem

This sections gives a description of the mathematical programming problem actually solved by the solvers as well as some internal data preprocessing leading to this problem.

The following formulation of the Mathematical Programming Problem (MPP) is used for the MPP definition common to all solvers.

In the following specification, the same notation is used as in Appendix A. Namely, indices $i \in I$, $j \in J$ correspond to emitters and receptors, respectively. The numbers of elements in I and J correspond to numbers of countries (about 40) and number of grids (600-700), respectively. However, in the MPP formulation, the indices will be assigned for consecutive numbers (starting from 0). Therefore, for the sake of interpreting a solution, identifiers for both emitters and receptors are necessary.

B.1 Variables

All variables are packed into one vector x, which is composed of vectors corresponding to variables en_j, ev_j, n_i, v_i. The composition of the vector x is the same for all solvers, namely

$$x = \{en, ev, n, v\}. \tag{13}$$

The lower and upper bounds for variables are denoted by superscripts l and u, respectively, and are defined during the problem generation as described in Appendices B.1.1 and B.1.2.

B.1.1 Nonlinear variables

en_j - effective emissions of NO_x experienced at the jth receptor:

$$en_j^l \le en_j \le en_j^u, \tag{14}$$

where en_j^l and en_j^u are defined by equation (9) for n_i^l and n_i^u, respectively.

ev_j - the representation of the second nonlinear term (in the previous formulation it was the effective emissions of VOCs experienced at the jth receptor):

$$ev_j^l \leq ev_j \leq ev_j^u, \tag{15}$$

where ev_j^l and ev_j^u are defined by equation (10) using v_i^l and v_i^u, depending on, whether the corresponding coefficient d_{ij} is positive or negative.

n_i - annual emission of NO_x

$$n_i^l \leq n_i \leq n_i^u \tag{16}$$

where n_i^l and n_i^u are defined by the domains of cost curves for NO_x.

B.1.2 Linear variables

v_i - annual emission of VOCs

$$v_i^l \leq v_i \leq v_i^u, \tag{17}$$

where v_i^l and v_i^u are defined by the domains of cost curves for VOCs.

Additional linear variables may be generated, if required, by converting of the nonlinear cost functions in a specific way.

B.2 Goal Function

The goal function is implemented as

$$\sum_{i \in I}(cn_i(n_i) + cv_i(v_i)) + \epsilon \sum_{i \in I}((n_i - \bar{n_i})^2 + (v_i - \bar{v_i})^2). \tag{18}$$

B.3 Constraints

For the sake of brevity, all summations in the following specification of constraints are done for $i \in I$. However, in the actual implementation, all matrices are stored as containers of sparse vectors; therefore, the summations are done for nonzero elements only.

B.3.1 Nonlinear constraints

Constraints (11) and (12) are combined as follows:

$$\sum_{i \in I}(a_{ij}v_i + b_{ij}n_i + \gamma_{ij}n_i^2) + (\alpha_j en_j + \beta_j ev_j)en_j \leq o_j^{max} - k_j + s_j, \tag{19}$$

where the surplus terms s_j are generated only if allowed by a corresponding option and are defined by

$$s_j = \max(o_j^{min} - o_j^{max} + o^{feas}, 0), \tag{20}$$

where o_j^{min} is defined by equation (11) for values of v_i, n_i, en_j, ev_j set to $v_i^l, n_i^l, en_j^l, ev_j^l$, respectively, and o^{feas} is a given feasibility tolerance for the constraint (12).

B.3.2 Linear constraints

Equation (9) is converted into

$$en_j - \sum_{i \in I} e_{ij} n_i = enn_j. \tag{21}$$

Equation (10) is converted into

$$ev_j - \sum_{i \in I} d_{ij} v_i = 0. \tag{22}$$

Additional linear constraints may be generated, if required by converting nonlinear cost functions in a specific way.

B.4 Names of Variables and Constraints

Because of the typical limitations for length of names (maximum of eight characters) a dual system of names is implemented, namely, shorter names for solvers and longer user names.

B.4.1 Names for solvers

Names for solvers are composed of a character and a number. Characters r and c are used for rows (constraints) and columns (variables), respectively. A number is a sequence number, starting from zero, used for numbering variables [in the sequence defined by (13)] and rows [in the sequence defined by (19), (21), and (22), possibly followed by additional constraints resulting from a conversion of PWL functions).

B.4.2 User names

A user name is composed of a root and an identifier (denoted by id), separated by the _ character. An identifier id is used instead of a number of variable/constraint. Such an identifier id is defined by an identifier for an emitter or a receptor, whichever is appropriate. Shorter names are extended by trailing dots in order to make all names of equal length.

Root names for variables are defined by a variable name. Root names for constraints are defined as follows:

o3 - O_3 concentrations (19);

en - effective NOx (21);

ev - effective VOCs (22).

B.5 Scaling

- The values of costs will be scaled so that the sum of costs will be of range of hundreds (hence cost for each emitter and type of pollution should be about 10).
- Emissions (n_i and v_i) will be scaled by the factor $100/n_{i90}$ and $100/v_{i90}$, respectively, where n_{i90} and $v_{i90}$are corresponding components of vectors n_{90} and v_{90} (the emissions in 1990). Solutions for such a scaling are expected to be in the range [25 – 150].

- Constraints (19) will not be scaled (expected values of O_j are about $40 \div 70$).
- Constraints (21) will be scaled by the factor 100 (expected range before scaling: $[0.01, 0.7]$).
- Constraints (22) will not be scaled (expected range $[-2.0, 60.0]$).

B.6 Dimensions of the Optimization Problem

Let E be a number of emitters and R a number of receptors. E is about 40, R about 600 (in the current data E=38, R=598). E may increase to about 700.

The dimensions of the optimization problem are as follows:

- $1 + \mathrm{E} + 2 \times \mathrm{R}$ non-linear variables $(cost, n_i, en_j, ev_j)$;
- E linear variables (v_i); and
- R nonlinear constraints defined by (19).
- $2 \times \mathrm{R}$ linear constraints: (21) and (22).

C Specification File

Each line of the file contains one option. The option contains of a key word and, for some options, an option value.

The options are given in Table 2.

Option	*Meaning*	*Default*
data_file	HDF file	data.hdf
Solution_file	solution file	_solution
no_scale	Do not scale model	-
no_check_empty	Do not check for unused variables and constraints involving no variables	-
test	Use test data instead of reading HDF file	-
gen_files	Generate ASCII files containing test data	-
test_with_rfp	Test data contains non-default reference point	-
test_with_start	Test data contains non-default starting point	-
wise_bounds	Relax bounds to avoid singularities	-
list_data	Dump model	-
cost_pwl	Use PWL representation of cost functions instead of smooth one	-
epsilon	Regularization coefficient	1E-4

Table 2: Specification file

Switching on the "wise_bounds" option relaxes the admissible solutions set. Normally, the bounds on en_j and ev_j are derived from n_i and v_j , using equations (21) and (22) as

described in Appendix B. However, the bounds for en_j, ev_j can be reached simultanously with the bounds for n_i and v_i. To avoid any singularities the bounds for en_j and ev_j are moved in order to enlarge the admissible solutions set. The moving of bounds consists in multiplying the bound by a number slightly greater than one and adding a small number to the bound.