**Title**: Database Structure for a Class of Multi-Period Mathematical Programming Models

**Author 1**: Goutam Dutta

Indian Institute of Management, Ahmedabad

Vastrapur, Ahmedabad 380015, India

**Author 2**: Robert Fourer

Department of Industrial Engineering and Management Science

Northwestern University, Evanston, IL 60208, USA

**Abstract**

We describe how a generic multi-period optimization-based decision support system can be used for strategic and operational planning in a company whose processes can be described in terms of five fundamental elements: Materials, Facilities, Activities, Times and Storage-Areas. We discuss the issues of interface design, data reporting and updating, and production and profit planning. We also compare the performances of two different types of database structures with respect to optimization.

**Keywords**: Decision Support System, process industries, optimization, strategic and operational planning.

## 1. Introduction

This work started as a project to design an optimization-based decision support system (DSS) for strategic planning for steel companies in North America. As the project was supported by The AISI (American Iron and Steel Institute), the DSS was generic in concept, but capable of being specifically applied to any particular company's facilities by supplying appropriate data Fourer [9]. Complete data for a steelmaking operation, including values such as yields, capacities, and prices indexed over products and processes could be conveniently supplied in the form of relational database files. The optimization was taken to be over a single planning period, however, and thus difficulties involving the indexing of data and model entities over time were not addressed.

This paper extends the work of Fourer [9] to a multi-period case. We adopt the fundamental elements of Materials, Facilities, and Activities from the previous work, but add two more elements - Times and Storage-Areas. Among the major points we address are the following:

- What are the key features of a multi-period DSS?
- What are the difficulties in implementing a multi-period DSS?
- What are the alternatives for handling multi-period indexing in a database context?
- In what ways can the optimal result be represented in a multi-period DSS?
- Why is an update mode difficult in a multi-period database?
- How do alternative data structures compare with respect to data storage, data retrieval, and support for optimization models?

### 1.1 Literature Review

Database representation of an LP is one of the eight approaches of LP representation comprehended by Murphy [12]. The approach is considered as translation form, as it is used as a bridge between modelers' form, and algorithmic form. Fourer [8] recognized that no single form of LP representation can be developed which can be easily understood by modelers', computers, and industry practitioners simultaneously. Database representation of LP was initially discussed by Fourer [8]. Murphy [12] summarized eight most popular approaches of representing an LP. He summarized the modelers' form, algorithmic form, and translation forms of LP representation.

Most of the industry data remains in database systems, one need to look for a system which can handle the bulk data required for an LP in a systematic manner. Data modeling in context with mathematical programming is discussed in detail (Dominguez, Mitra, and Lucas [5]). Readers interested in database systems, and data modeling are referred to the book by Date [2]. We recognize that not much work is done in representing an LP in translation form in general and database form in particular. We demonstrate (using an LP model for strategic planning in process industry), how an LP can be represented in the form of a database structure. We follow the widely accepted Dolk [3, 4] framework for data, model, and dialogue management.

### 1.2 Outline

In section 2 of this paper, we discuss the design issues raised by a multi-period database. We introduce the various elements of the DSS and discuss possible implementations. We also consider the correspondences of the various files in the DSS with the various variables in the linear program. In section 3, we discuss the various steps of multi-period optimization - constraint and variable generation, coefficient matrix generation, solution of the optimization problem, and reading of the optimal values back into the database. We also indicate how we allow for soft capacities through the use of artificial variables. Section 4 considers how the various features of the DSS can be useful for the strategic and operational planning in a process industry. Section 5 discusses the various features for reporting and updating of the data, and in section 6, we compare two different variations of the database design – one primarily hierarchical and another primarily relational - with respect to optimization. We conclude the paper by outlining the scope for further work.

## 2. Database Design Issues for Multi-Period Models

Our generic multi-period planning model has, as previously noted, five fundamental elements:

***Times*** are the periods of the planning horizon, represented by discrete numbers (1, 2, 3 ...). They can be as short as weeks, though for a planning model they are most likely to range from months to years.

*Materials* are the physical items that figure in some stage of production. They may be inputs, intermediates, or outputs, and sometimes more than one of these.

***Facilities*** are collections of machines that produce some materials from others. For example, a Hot Mill that produces sheets from slabs is a facility.

***Activities*** are productive transformations of materials. Each facility houses one or more activities, which uses and produces materials in certain proportions. Production of hot metal, production of billets, pickling, and galvanizing are examples of steelmaking activities.

***Storage-Areas*** are fields or warehouses where raw materials, intermediate products, or finished products may be stored.

An algebraic formulation for the multi-period model is given in **Appendix.**

### 2.1 The *Times* file

Our database is implemented within 4th Dimension, a relational database management system, Adams [1]. Other database systems such as Access or Oracle could be used just as well. Figure 1 summarizes the structure of the database as expressed within 4th Dimension. The five boxes labeled *Materials, Facilities, Activities, Times,* and *Storage-Areas* correspond to the five major Elements, or files, of the database. Items within each box denote the file's data fields and subfiles, with the subfile entries distinguished by a light-shaded line that runs to the top of a separate box in which the subfile data fields are listed. The smaller, independent database structure in the upper right of the diagram holds a generated linear program as described at the end of this section. We use 4th Dimension's notations for files, subfiles, and field. Further details can be found in the earlier discussion of the one-period model Fourer [9].

The structure of the Times file in the database is very simple, consisting basically of a record per period. A name field can be adjusted according to whether the periods are modeling, say, quarters or years. The complications introduced by the multi-period structure lie mainly in the ways that Times interact with all of the other pieces of the database structure.

***Insert Figure 1***

### 2.2 Materials File

The Materials data (Figures 2 and 3) are stored in a hierarchical way. In Table 1, we show the one-to-one correspondence between the parameters of the LP model and the fields in the [Materials] file. In this file the material name ([Materials]MatName) and material identification string ([Materials]MatTag) are unique. [Materials]MatTag is required for data entry in the sub-files of the [Materials] file. In the [Materials] file, BuyMax, BuyMin, SellMax, SellMin, BuyPrice, SellPrice, BuyOpt, SellOpt, InvMax, InvMin, InvOpt, InvCCost, CostIn and CostOut are the time-dependent subfields in the [Materials]MatTime sub-file. Since the model is multi-period, the dual variables (MatDual) are also considered as a function of time and are put in the MatTime sub-file.

***Insert Table 1***

***Insert Figure 2 and 3***

**2.3 Facilities File**

In the Facilities file (Figures 4 and 5), for time dependent parameters we retain a structure similar to that of the [Materials] file. In Table 2 of Appendices, we show the one to one correspondence between the parameters of the LP model and the fields [Facilities] file. We define [Facilities]FacTime as a subfile where the CapMax, CapMin, CapOPT and CapDual subfields are the time dependent maximum, minimum, and optimal production levels of the facility, and the time dependent dual value of the facility capacity. The VendorCost is the cost of vendoring (outsourcing) an additional unit capacity of the facility at that time.

There are two indexed subfields in [Facility]Inputs, which is a sub-file of the Facilities file. The first one is the input material, which is related to the [Materials] file. The other is [Facility]Inputs'InTime which is the time dependent field of the [Facilities]Input File and is related to the Time file. The subfile [Facilities]Outputs is entirely analogous.

***Insert Figure 4 and 5***

***Insert Table 2***

**2.4 Activities File**

[Activities] is defined (Figure 6) as a separate file. (In STEEL-TIME2, we consider [Activities] as a sub-file of the [Facilities] File). There is a field of [Activities]ActTime which is the indexed field of time in the [Activities] file and related to the [Times] file. In Table 3 we show one to one correspondence between the parameters of the LP model and the [Activities] file. In each [activities] file there is a field; ActFacName that specifies which facility it belongs to. This is required so that the user can search for the activity through the facility. The other important field is ActTag, the unique identification of each activity. The [Activities] file can be indexed over [Activities]Act Name or [Activities]ActTag (identification string). Two activities may have the same ActName (like PRODUCTION OF BILLET), but if they have a different ActTime, they will

have a different ActTag. In other words every record of [Activities] file will be identified by a unique [Activities]ActTag.

While defining the activity inputs (ActInMat) or activity outputs (ActOutMat), we have to consider the fact that ActInMat (or ActOutMat) should have only those materials which are in Facility Inputs (or Outputs) and also at the time where ActTime is equal to [Facilities]Inputs'IntTime ([Facilities]Outputs'OutTime). For example, let us assume that the BLOOM, BILLET and SLAB are available as [Facilities]Inputs at Time =1 in [Facilities]FacName =ROLLING MILL, but BLOOM and SLAB are only available as [Facilities]Inputs at Time =2 in the same facility. In the [Activities]File at Time=1 the possible choices available in the subfield Activities]ActInPuts'ActInMat are BLOOM, BILLET and SLAB, but only SLAB and BLOOM are available as [Activities]ActInputs'ActInMat at Time =2 in the same facility.

***Insert Table 3***

***Insert Figure 6***

**2.5 Storage-Areas File**

In the [Storage-Areas] file we have the name of the Storage-Area and the time at which the materials are stored. In addition, we have the capacity constraint of the storages giving the maximum and the minimum capacities of the storage-areas. The structure of the [Storage-Areas] file is similar to that of the [Activities]File. [Storage_Areas]StoreTag is the field which uniquely identifies the records of the file. In the [Storage-Areas] file, we have a sub-file called [Storage-Areas]StoreMatList which lists all the materials that can be listed.

**2.6 Variables File**

In the [Variables] file we have fields Number, Type (Material Bought, Material Sold, Material Inventoried, Activity at Facility), Identification Number 1 (ID1), Identification Number 2 (ID2), Objective, Upper bound and Lower bound as in the single period model. However, we have also an Identification Number 3 (ID3) field which indicates the time of the variable. [Variables]Optimal refers to the most recent optimal value of

the variable. The variables file has a sub-file known as [Variables]Coeffs which has a subfield called [Variables]Coeffs'Constr and this constraint is related to the [Constraints]Number of the [Constraints] file.

### 2.8 Constraints File

In addition to [Constraints]Number, the [Constraints] file has a field for Type (Material Balance, Facility Input, Facility Output and Facility Capacity, Storage Capacity or Storage Total, which refer to the equation numbers 3-8 respectively in Appendix). The Identification Number 1 ([Constraints]ID1) indicates the Material Name for the Material Balance equation and Facility Name for the other three types of constraints. The Identification Number 2 ([Constraints ID2) refers to the material for the Facility Input and Output respectively. As in the [Variables] file, ID3 refers to the time of the variable. [Constraints]Dual refers to the dual variable corresponding to the most recent optimal solution.

## 3. Optimization

Once the data of the five database files and their respective sub-files are entered, they are validated by a set of diagnostic tests to be explained in the next sub-section.

### 3.1 Optimization Steps

This subsection describes how the subsequent optimization process is carried out. The principal steps (Figure 7) are as follows:

***Insert Figure 7***

1. The data describing the production scenario at different time periods is collected and stored in the database.
2. The constraints associated with the linear program are generated. The constant terms of the constraint equations or inequalities, LoRHS and HiRHS, are extracted from the database and stored in the [Constraints] file.
3. The variables of the associated linear program are determined, along with their coefficients in the constraints. Variables are stored in a separate [Variables] file and coefficients in its [Variables]Coeff subfile. This step gives the user a choice of discounted or undiscounted optimization. If the latter is

chosen, it prompts for an interest rate, and all cost, price, and revenue data are converted to their discounted values in the objective function.

4. The [Constraints] and [Variables] files are scanned and all of the essential information about the linear program is written to an ordinary text file in a compact format. This text file is the input file to our solver.
5. A linear programming solver reads the text file - we used XMP, by Martsen [11] - which solves the indicated linear program and then writes the optimal values of the variables to a second text file.
6. The second text file is read and the optimal values are placed in appropriate fields of the [Materials], [Facilities], [Activities], and [Storage_Areas] files and their sub-files.

### 3.2 Diagnostics Rule

The diagnostic routines are written to ensure that the linear program is complete and free from errors and infeasibilities. We use the various file procedures, layout procedures and global procedures to implement these routines and following rules.

**Rule 1:** For every variable the upper bound should not be less than the lower bound. For every constraint the lower right hand side (LoRHS) should not be more than the higher right hand side (HiRHS).

**Rule 2:** For every variable and every constraint, there should not be more than one non-zero element.

**Rule 3:** For every sub-file indexed over one time subfield, the number of sub-records in the sub-file should be same as the number of records in the [Times] file.

**Rule 4**: For files and sub-files indexed over one time field and one non time field, the number of records (or sub-records) should not be more than the product of the number of records (or sub-records) in the [Times] file and the number of records related to the non-time field.

**Rule 5:** If a record or sub-record is indexed over a time field or sub-field and one non-time field or sub-field, there will be only one record or sub-record containing any particular combination of the time field and non-time field.

## 4. Features of the DSS

We would like to use this DSS for strategic and operational planning. In this subsection, we will discuss various features of this DSS.

### 4.1 Strategic and Operational Planning

In strategic planning, the DSS will be able to answer questions such as:

1. What is the effect of cost or price changes of raw materials and finished products on the product-mix?
2. If the company is planning to diversify into different products, what products should be chosen?

In operational planning the DSS will be able to help the steel company officials with questions like these:

1. In response to a shortage of liquid steel, which results in the partial operation of the finishing mills in the downstream production line, which of the finishing mills should go down?
2. Should external scrap be purchased as a substitute for hot metal and at what price?

For example, in the experience with an Indian steel company (Sinha [14], Dutta [7]) the marginal profit of an extra megawatt of electrical power was found to be several million dollars. This study justified the investment of installing diesel-generating sets. Similar studies can be done using our DSS.

### 4.2 Soft Capacities

If we have infeasibility in the "Facility Capacity" constraint, we can generate a "Soft Capacity" variable, which is similar to an artificial variable. At the end of step 2, the user will have the option to use a procedure which generates this variable. The concept of soft capacities is described in detail by Dutta et a. (2004).

### 4.3 User Friendliness

This is the most important point of this research. We have been able to demonstrate that multi-period, multi-product, multi-facility process industry planning can be done with little or no knowledge of linear programming. All the user has to do is click the appropriate buttons to run the required linear programs.

The DSS can be used in three modes: *Data, Optimal* and *Update*. In the *Data* mode, the user enters data in the five different files. The Optimal mode is for display of optimal values and dual prices. The DSS takes much longer (92 minutes) to generate the [Variables] file and the [Constraints] file than to solve the problem (3 minutes). If there is no addition or deletion of records in the [Materials], [Facilities] and [Activities] file,

any change in the parameters of these files can be reflected in the corresponding changes to the [Variables] and [Constraints] file (without procedures of variable and constraint generation). This is accomplished in the *Update* mode resulting in saving of user time.

### 4.4 Multi-Period Model

The multi-period structure of our DSS has the following advantages:

1. The model can show how the cash flow of the company changes with different interest rates. The user is allowed to enter the interest rate. The user also has the option to optimize over nominal or discounted financial parameters.
2. The importance of inventories is considered in this model. Using this DSS we will be able to make decisions as to whether it is more profitable to produce at the current time period and hold inventory, or to produce in the future.
3. The user can see the effect of changing the parameters in one time period on the optimal decisions for other time periods.

### 4.5 Generality and Flexibility

The model is sufficiently generic so that it can be used by any process industry that transforms materials in different facilities. When the company decides to make any new product, a record can be added to their materials database. Similarly when a new facility is installed the user can enter an appropriate record. For any linear programming model done in AMPL or GAMS or Excel Solver the user does not have the advantage of route flexibility. In this DSS, any route of the product can be added or deleted by addition and deletion of appropriate material, facility and activity. If another industry wants to use this software, they only need to change the relevant data entry files for their company.

## 5. Reporting and Updating the Data

In this section, we consider the different files and discuss the time dependent layouts where the time dependent parameters are entered as subfields.

### 5.1 Layouts with Time as a Subfield

First, let us consider the [Materials] File. In this file, no time dependent parameters are in the file level except for MatInvZero. These fields will be the same in *Data* or *Optimal* layouts. In order to see the optimal value of the material COIL bought at Time = 2, the user has to select the optimal mode in the *Examine* menu of the main menu and select Materials. Then a list of Materials will be displayed. The user has to then select the material COIL and a layout called Materials Optimal (Figure 8) will be displayed. In this layout there will be an included layout that lists the data of all time dependent parameters of the materials COIL. Once the user selects Time=2 a list of parameters is displayed in a layout for Time=2 and one of them is BuyOPT which shows the optimal value of Material bought in Time= 2. Similarly, if the user wants to get the BuyPrice of material called SCRAP at Time =3, he or she has to go through steps similar to all these.

We now discuss two different types of searches. We want to compare the searching process of an activity and an input material in the same [Facilities] file. Let us assume that [Facilities]FacName= BASIC OXYGEN FURNACE. The user selects Facilities and Optimal in the *Examine* menu of the main menu and gets a listing of all facilities and selects the facility = BASIC OXYGEN FURNACE and goes to the Facilities Optimal screen.

***Insert Figure 8***

This is common to both the searches. In the first search, he or she clicks the Activities button and goes to the next page of the Facilities Optimal Screen. This screen layout lists all the activities in this facility as an included layout. If the user wants to find the values of rate for the output material STEEL for the Activity = CRUDE STEEL PRODUCTION at Time=2 of this facility, then he or she looks at the list of activities and searches for Activity = CRUDE STEEL PRODUCTION and Time=2. This leads to an Activity Optimal Screen which lists the output materials. Then this list gives the value of output rate for the output material =STEEL. *In this case, to get a required value, we first search (on the [Activities] file) with a combination of two fields, and then look for a sub-file or subfield.* In the second search, to get the maximum value of input material STEEL SCRAP that can be accommodated in this facility at Time=2, the user looks at the facilities Optimal Screen and looks at the included layout of Inputs. This included

layout lists all the input materials at all times. The user then searches for Material = STEEL SCRAP and Time=2. *In this case the search is performed with two searches at the sub-file level*.

**5.2 Included Layouts and Graphs in the Time File**

Suppose we have a question from a user. At Time =1, what is the optimal value of material sold for SINTER, and HIGH CARBON BILLET? In the Examine menu, the user can select Materials and *Optimal*, and this will lead to a list of Materials. The User can double click at SINTER and this will lead to the Materials Optimal screen of SINTER. In this screen there will be a list of Times and the user can find the optimal value of material sold at Time = 1 in this list. Then he has to return to the list of Materials and double click here again at HIGH CARBON BILLET. Then he gets another Materials Optimal Screen of HIGH CARBON BILLET. Then he can look again at the Time Layout and see the material bought at Time = 1. This is a cumbersome procedure. At Time=1, the user cannot go from one material to another. This can be overcome by making an included layout of the [Materials] file in the [Times] file.

**5.3 Reporting of Optimal Dual values**

In this section, we discuss the difficulties in reporting the optimal dual values in multiple time periods. For a single period model, the display of dual values is simple and straightforward. However, for the multi-period model we have dual values for more than one time period. In addition, the reduced cost for the variable Material Inventoried, any time period is dependent on dual values from more than one time period. This makes our task difficult for displaying the optimal dual values.

**5.4 Optimal Summaries**

In the case of a multi-period model, creation of summaries is difficult and not straightforward like in single period models. In this section, we discuss two different ways the summaries can be displayed: summaries of each time period separately, and grand summaries for all time periods.

We repeat the equation of the objective function **(equations of Appendix)**:

$$Z(t) = \left(\sum_{J \in M} c_{jt}^{sell} x_{jt}^{sell} - \sum_{J \in M} c_{jt}^{buy} x_{jt}^{buy} - \sum_{(j,j') M^{conv}} c_{jj't}^{conv} x_{jj't}^{conv} - \right.$$

$$\left. \sum_{(i,k) \in F^{act}} c_{ikt}^{act} x_{ikt}^{act} - \sum_{j \in M} h_{jt} x_{jt}^{inv} - \sum_{i \in F} C_{i,t}^{vend} x_{i,t}^{vend} \right) \quad \textbf{(1)}$$

$$\mathbf{Z} = \boxed{\sum_{t \in T} Z(t)} \quad \textbf{(2)}$$

We will now break it up into different parts. Typically a user would like to answer "What is the sum total of revenue obtained by selling all materials at one time (say Time = t)?" Let us define it as it R(t), the revenue at time t :

$$R(t) = \sum_{J \in M} s_{jt}^{sell} x_{jt}^{sell} \quad \textbf{(3)}$$

Similarly we can write the corresponding summation terms for the other terms. We define

$Cp(t)$ = Cost of purchase of all materials at time t

$Ca(t)$ = Cost of all activities at time t

$Ci(t)$ = Cost of carrying inventory at time t

$Cc(t)$ = Cost of conversions at time t

$Cv(t)$ = Cost of outsourcing at time t

Once we have calculated all the six quantities we can rewrite the net profit as the following:

$$Z(t) = R(t) - Cp(t) - Ca(t) - Ci(t) - Cc(t) - Cv(t) \quad (4)$$

The terms of equation 5.5 can be displayed in a grand summary over all time periods (Figure 9). Based on the equation 5.8, we can also display the summary for each period (Figure 10).

***Insert Figure 9, and 10***

**6. Comparison of Database Structures**

In this section, we consider the different variations of the [Materials] and [Facilities] files. These files can be organized in several ways and we discuss how the computer times for variable and constraint generation vary

with different variations of the relational and hierarchical databases. We consider two different types of structures: STEEL-TIME1and STEEL-TIME2. The structure of STEEL-TIME1 is similar to STEEL-TIME (which we have discussed in Section 3. Fourer (1997) has studied two different variations of the [Constraints] and [Variables] files, one relational and one hierarchical. We extend his comparison to two different variations of the [Materials] and [Facilities] files. We compare the implementation of STEEL-TIME1 and STEEL-TIME2 according to four different criteria: ease of use, data storage and retrieval, ease of development and efficiency of optimization.

***Insert Figure 11 and 12***

**6.1 Implementation of STEEL-TIME1 vs. STEEL-TIME2**

STEEL-TIME1 is a modified version of STEEL-TIME Fourer (1997). We find that STEEL-TIME2 is faster in generating the variables and constraints than STEEL-TIME1.This is because in STEEL-TIME1, the data for time dependent parameters are stored in a sub-file. So every time a record is written in the [Variables] file, first the record of the [Materials] is searched for, then the sub record of the file is searched for, and finally the record is written in the [Variables] file. However in STEEL-TIME2, fields like BuyMax, BuyMin are at the field level. Therefore to write a record in the [Variables] file, we only have to search the [Materials] and [Facilities] at the file level. Similarly, the disk-space for the data of STEEL-TIME2 is higher than that of STEEL-TIME1. This is because time-independent parameters like MatName, MatTag, MatInvZero, FacName, FacTag, FacType etc. are duplicated in STEEL-TIME2.

The numbers of constraints and variables in STEEL-TIME1 and STEEL-TIME2 are equal. The other similarities and differences of STEEL-TIME1 and STEEL-TIME2 are as follows:

1. In STEEL-TIME1, the time dependent parameters are in subfields of the [Materials] and [Facilities] files. In STEEL-TIME2 these are in the fields of the [Materials] and [Facilities] files.
2. The [Storage-Areas] file of STEEL-TIME is not considered in this comparison. In addition, vendoring or outsourcing is not considered an option. Even if the indexing in the formulation and the way of representing the mathematical model are different, we essentially solve the same optimization problem in STEEL-TIME1 and STEEL-TIME2.

3. STEEL-TIME1 or STEEL-TIME2 cannot be clearly classified as a purely relational or purely hierarchical database. Each has both relational and hierarchical aspects. STEEL-TIME1 is more relational and [Activities] is a separate file. STEEL-TIME2 is more hierarchical, and [Activities] is a sub-file of the [Facilities] file.

**6.2 Ease of Use**

STEEL-TIME1 appears to be more complicated than STEEL-TIME2. Other than the [Times] file there are only two files in STEEL-TIME2, the [Materials] and the [Facilities] file.

Therefore it is easier to use STEEL-TIME2 than STEEL-TIME1. In the [Materials] file, all the purchase, sales and inventory related data about the Materials are kept at the file level. When the materials are displayed on an output layout, in STEEL-TIME2, sorting is possible with respect to the [Materials]MatName as well as [Materials]MatTmeID. However in STEEL-TIME1, [Materials]MatName is at the file level and the [Materials]MatTime'MatTimeID is at the sub-file level. So sorting is not possible at the same level in STEEL-TIME1.

In STEEL-TIME1, there are three files and [Activities] is a separate file related to the [Facilities] file. From a developer's point of view, STEEL-TIME1 is more complicated than STEEL-TIME2. Moreover, most of the searches are performed at the sub-file level. For example, it is possible to list the dual prices and the reduced cost coefficients in the output layout at the file level in STEEL-TIME2, but similar lists are not possible in the STEEL-TIME1. Such a display can be available in STEEL-TIME1 at the sub-record level only. On the other hand, STEEL-TIME1 has a greater flexibility for listing the activities, as [Activities] is a separate output file. Because of the inherent advantages of the relational file, the user will be able to update activities separately. Although we have not implemented this concept in STEEL-TIME1, such an implementation is possible. STEEL-TIME1 will also allow the user to compare two activities of two facilities by listing activities on the output file. So an activity PRODUCTION OF ES1 in three facilities M1, M2, M3 can be listed by performing a search with [Activities]ActName = "PRODUCTION OF ES1". Such searches are not possible with STEEL-TIME2.

**6.3 Data Storage and Retrieval**

STEEL-TIME1 satisfies the conditions of normalization that no piece of information be stored in more than one place. This condition is not satisfied in STEEL-TIME2. We also see that STEEL-TIME2 takes greater storage space than STEEL-TIME1.

In STEEL-TIME2 certain fields are repeated. [Materials]MatName, [Materials]MatType, [Materials]MatInvZero, [Materials]MatUnits, [Facilities]FacName and [Facilities]FacUnits are the fields that are repeated for every record of the [Time]TimeID file. This certainly requires more space for data storage, but does not pose a very serious problem with respect to ease of use. The 4th Dimension software allows a script to be written so that when the user enters the data for [Materials]MatName for one time period, the same [Materials]MatName is also available in other time periods. Therefore, as long as we are not changing [Materials]MatTime, we do not need to enter the data for each time period.

**6.4 Ease of Development**

STEEL-TIME2 is easier to develop than STEEL-TIME1. However, we have decided to opt for STEEL-TIME1 as our main implementation, primarily because the latest version of the 4th Dimension software does not support more than one level of sub-file. Because of the inherent advantage of relational databases, [Activities] was defined as a separate file in STEEL-TIME1, whereas it was a sub-file in the [Facilities] file of STEEL-TIME2.

**6.5 Efficiency**

The times for constraint generation, variable generation and solution, and reading optimal values and the dual values are as shown in Table 4.

***Insert Table 5***

We find that STEEL-TIME2 is faster in generating the variables and constraints than STEEL-TIME1. This is because in STEEL-TIME1, the data for time dependent parameters are stored in a sub-file. So every time a record is written in the [Variables] file, first the record of the [Materials] is searched for, then the sub-record of the file is searched for, and then the record is written in the [Variables] file. However in STEEL-TIME2 fields like BuyMax, BuyMin are at the field level. Therefore to write a record in the [Variables] file, we only have to

search the [Materials] and [Facilities] at the file level. Similarly, the disk-space for the data of STEEL-TIME2 is higher than that of STEEL-TIME1. This is because time-independent parameters like MatName, MatTag, MatInvZero, FacName, FacTag, FacType etc. are duplicated in STEEL-TIME2.

After a careful comparison of these two variations, we find that STEEL-TIME2 is superior to STEEL-TIME1 on an overall basis. However we need to extend the present study so that STEEL-TIME2 is normalized. This can be done by replacing all the sub-files by files so that [Materials]MatTime and [Facilities]FacTime and other sub-files will be normalized with additional indices and key-fields. We will be in a position to recommend STEEL-TIME2 only after that.

**7. Extension and Conclusion**

An extension of the DSS will be non-linearity of the model. Most of the industrial cost curves are non-linear or at best can be represented as having a piece-wise linear behavior. It will be interesting to study how to represent these non-linearities while retaining the model's user-friendliness.

A second extension of the model will be to have multiple objective linear programs and represent them in the database. This can be done by changing the model management system. For example, the current model can be changed to cost minimization, revenue maximization, maximization of marketable products (revenue or production), maximization of the utilization of the facilities etc. It is possible to have a menu driven program in this DSS which optimizes over different objectives.

An interesting extension will be to study the paradigm neutrality Geoffrion [10] of this data structure for the multi-period model. Although the model is designed for the mathematical programming paradigm, we can extend it for inventory control and also for scheduling, vehicle routing and queuing applications. We have parameters for all materials at all times. We can determine the ordering and holding cost for all material and hence try to find optimal order quantities. However, the batch size will be decided by practical considerations like the **heat size** of the steel making shop, the capacity of the vehicle carrying the products and the capacity of the loading and unloading facility. Given that we have the batch size and lead-time of all materials produced, the present model can be extended to a scheduling model of each product in each time.

**Tables**

**Table 1**

Correspondence of [Materials]File and the LP Model

| Sr. No. | Parameter of the LP | Fields of the Tables of the Database |
|---|---|---|
| 1 | $l_{jt}^{buy}$ | [Materials]MatTime'BuyMin |
| 2 | $u_{jt}^{buy}$ | [Materials]MatTime'BuyMax |
| 3 | $c_{jt}^{buy}$ | [Materials]MatTime'BuyPrice |
| 4 | $l_{jt}^{sell}$ | [Materials]MatTime'SellMin |
| 5 | $u_{jt}^{sell}$ | [Materials]MatTime'SellMax |
| 6 | $c_{jt}^{sell}$ | [Materials]MatTime'SellPrice |
| 7 | $l_{jt}^{inv}$ | [Materials]MatTime'InvMin |
| 8 | $u_{jt}^{inv}$ | [Materials]MatTime'InvMax |
| 9 | $h_{jt}$ | [Materials]Mattime'MatInvCCOST |
| 10 | $x_{j0}^{inv}$ | [Materials]MatInvZero |
| 11 | $a_{jj't}^{conv}$ | [Materials]Conversion'ConvYield |
| 12 | $c_{jj't}^{conv}$ | [Materials]Conversion'ConvCost |

**Table 2**

Correspondence of [Facilities] File and the LP Model

| Sr. No. | Parameter of the LP | Fields of the Tables of the Database |
|---|---|---|
| 1 | $l_{ijt}^{in}$ | [Facilities]Inputs'InMin |
| 2 | $u_{ijt}^{in}$ | [Facilities]Inputs'InMax |
| 3 | $l_{ijt}^{out}$ | [Facilities]Outputs'OutMin |
| 4 | $u_{ijt}^{out}$ | [Facilities]Outputs'OutMax |
| 5 | $C_{it}^{vend}$ | [Facilities]FacTime'Vendor_Cost |
| 6 | $l_{it}^{cap}$ | [Facilities]FacTime'CapMin |

| | | |
|---|---|---|
| 7 | $u_{it}^{cap}$ | [Facilities]FacTime'CapMax |

**Table 3**

Correspondence of [Activities] File and the LP Model

| Sr. No. | Parameter of the LP | Fields of the Tables of the Database |
|---|---|---|
| 1 | $l_{ikt}^{act}$ | [Activities]ActMin |
| 2 | $u_{ikt}^{act}$ | [Activities]ActMax |
| 3 | $u_{ikt}^{act}$ | [Activities]ActCost |
| 4 | $r_{ikt}^{act}$ | [Activities]ActCapUsed |
| 5 | $a_{ijkt}^{in}$ | [Activities]ActInputs'ActInMat |
| 6 | $a_{ijkt}^{out}$ | [Activities]ActOutPuts'ActOutMat |

**Table 4**

Comparison of Steel-Time 1 and Steel-Time2

| **Computer** | **Macintosh** | |
|---|---|---|
| **Database** | **STEEL-TIME** | **STEEL-TIME2T-2** |
| **Records in Files** | | |
| **Materials** | **19** | **57** |
| **Facilities** | **21** | **21** |
| **Activities** | **24** | **24 (Sub-file)** |
| **Times** | **3** | **3** |
| **Constraints** | **141** | **141** |
| **Vairables** | **266** | **266** |
| **Disk Space (Model)** | **688** | **336** |
| **Disk Space (Data)** | **472** | **484** |
| **Cons. Generation Time** | **12** | **12** |
| **Var.Generation Time** | **109** | **45** |
| **Writing Constraint Time** | **7** | **7** |

| | | |
|---|---|---|
| **Writing Variable Time** | **22** | **21** |
| **Solving** | **8** | **8** |
| **Reading Optimal Value Tin** | **21** | **21** |
| **Reading Dual Value Time** | **8** | **8** |

**Acknowledgement**

This work has been supported in part by grants from the American Iron and Steel Institute and its members, by grant DDM-8908818 from the U.S. National Science Foundation, and by the Research and Publication Committee of the Indian Institute of Management, Ahmedabad.

## Appendix

## Model Formulation

We first define the data, in five parts: times, materials, facilities, activities, and storage-areas. The notation for the decision variables is then presented. Finally the objective and constraints are described, in both words and formulae.

All quantities of materials are taken to be in the same units, such as kilograms.

### Time data

$T$= {1,……,$T$} is the set of time periods in the planning horizon, indexed by t

$\rho$ is the interest rate per period, taken as zero if there is no discounting

### Materials data

$M$ is the set of all materials

$l_{jt}^{buy}$ = lower limit on purchases of material $j$, for each $j \in M$ and $t \in T$

$u_{jt}^{buy}$ = upper limit on purchases of material $j$, for each $j \in M$ and $t \in T$

$c_{jt}^{buy}$ = cost per unit of material $j$ purchased, for each $j \in M$ and $t \in T$

$l_{jt}^{sell}$ = lower limit on sales of material $j$, for each $j \in M$ and $t \in T$

$u_{jt}^{sell}$ = upper limit on sales of material $j$, for each $j \in M$ and $t \in T$

$c_{jt}^{sell}$ = revenue per unit of material $j$, for each $j \in M$ and $t \in T$

$l_{jt}^{inv}$ = lower limit on inventory of material $j$, for each $j \in M$ and $t \in T$

$u_{jt}^{inv}$ = upper limit on inventory of material $j$, for each $j \in M$ and $t \in T$

$v_{j0}^{inv}$ = initial inventory of material $j$, for each $j \in M$

$c_{jt}^{inv}$ = holding cost per unit of material $j$, for each $j \in M$ and $t \in T$

$M^{conv} \subseteq \{j \in M, j' \in M : j \neq j'\}$ is the set of conversions:

$(j,j') \in M^{conv}$ means that material $j$ can be converted to material $j'$

$\alpha_{jj't}^{conv}$ = number of units of material $j'$ that result from converting one unit of material $j$, for each $(j,j') \in M^{conv}$, $t \in T$

$c_{jj't}^{conv}$ = cost per unit of material $j$ of the conversion from $j$ to $j'$, for each $(j,j') \in M^{conv}$, $t \in T$

**Facilities data**

***F*** is the set of facilities

$l_{it}^{cap}$ = the minimum amount of the capacity of facility $i$ that must be used, for each $i \in F$ and $t \in T$

$u_{it}^{cap}$ = the capacity of facility $i$, for each $i \in F$ and $t \in T$

$c_{it}^{cap}$ = the cost of vendoring (outsourcing) a unit of capacity at facility $i$, for each $i \in F$ and $t \in T$

$F^{in} \subseteq FxM$ is the set of facility $i$nputs:

$(i,j) \in F^{in}$ means that material $j$ is used as an input at facility $i$

$l_{ijt}^{in}$ = the minimum amount of material $j$ that must be used as input to facility $i$, for each $(i,j) \in F^{in}$, $t \in T$

$u_{ijt}^{in}$ = the maximum amount of material $j$ that must be used as input to facility $i$, for each $(i,j) \in F^{in}$, $t \in T$

$F^{out} \subseteq FxM$ is the set of facility outputs:

$(i,j) \in F^{out}$ means that material $j$ is produced as an output at facility $i$

$l_{ijt}^{out}$ = the minimum amount of material $j$ that must be produced as output at facility $i$, for each $(i,j) \in F^{out}$, $t \in T$

$u_{ijt}^{out}$ = the maximum amount of material $j$ that must be produced as output at facility $i$, for each $(i,j) \in F^{out}$, $t \in T$

**Activities data**

$F^{act} \subseteq \{(i,k) : i \in F\}$ is the set of activities:

$(i,k) \in F^{act}$ means that $k$ is an activity available at facility $i$

$l_{ikt}^{act}$ = the minimum number of units of activity $k$ that may be run at facility $i$, for each $(i,k) \in F^{act}$, $t \in T$

$u_{ikt}^{act}$ = the maximum number of units of activity $k$ that may be run at facility $i$, for each $(i,k) \in F^{act}$, $t \in T$

$c_{ikt}^{act}$ = the cost per unit of running activity $k$ at facility $i$, for each $(i,k) \in F^{act}$, $t \in T$

$r_{ikt}^{act}$ = the number of units of activity that can be accommodated in one unit of capacity of facility *i*, for each $(i,k) \in F^{act}$, $t \in T$

$A^{in} \subseteq \{(i,j,k,t) : (i,j) \in F^{in} (i,k) \in F^{act}, t \in T\}$ is the set of activity inputs:

$(i,j,k,t) \in A^{in}$ means that input material *j* is used by activity *k* at facility *i* during time period *t*

$\alpha_{ijkt}^{in}$ = units of input material *j* required by one unit of activity *k* at facility *i* in time period *t*, for each $(i,j,k,t) \in A^{in}$

$A^{out} \subseteq \{(i,j,k,t) : (i,j) \in F^{out} (i,k) \in F^{act}, t \in T\}$ is the set of activity outputs:

$(i,j,k,t) \in A^{out}$ means that output material *j* is produced by activity *k* at facility *i* during time period *t*

$\alpha_{ijkt}^{out}$ = units of output material *j* produced by one unit of activity *k* at facility *i* in time period *t*, for each $(i,j,k,t) \in A^{out}$

**Storage-areas data**

*S* is the set of storage areas

$l_{st}^{stor}$ = lower limit on total material in storage area s, for each $s \in S$, $t \in T$

$u_{st}^{stor}$ = upper limit on total material in storage area s, for each $s \in S$, $t \in T$

**Variables**

$x_{jt}^{buy}$ = units of material *j* bought, for each $j \in M$, $t \in T$

$x_{jt}^{sell}$ = units of material *j* sold, for each $j \in M$, $t \in T$

$x_{jst}^{stor}$ = units of material *j* in storage area *s*, for each $j \in M$, $s \in S$, $t \in T$

$x_{jt}^{inv}$ = total units of material $j$ in inventory (storage), for each $j \in M$, $t \in T$

$x_{j0}^{inv}$ = initial inventory of material $j$, for each $j \in M$

$x_{jj't}^{conv}$ = units of material $j$ converted to material $j'$, for each $(j,j') \in M^{conv}$, $t \in T$

$x_{ijt}^{in}$ = units of material $j$ used as input by facility $i$, for each $(i,j) \in F^{in}$, $t \in T$

$x_{ijt}^{out}$ = units of material $j$ produced as output by facility $i$, for each $(i,j) \in F^{out}$, $t \in T$

$x_{ikt}^{act}$ = units of activity $k$ operated at facility $i$, for each $(i,k) \in F^{act}$, $t \in T$

$x_{it}^{cap}$ = units of capacity vendored at facility $i$, for each i$\in F$, $t \in T$

**Objective**

Maximize the sum, over all time periods, of revenues from sales less costs of purchasing, holding inventories, converting, operating activities at facilities and vendoring:

$$\sum_{t \in T} (1+\rho)^{-t} Z(t)$$

Where,

$$Z(t) = \sum_{j \in M} c_{jt}^{sell} x_{jt}^{sell} - \sum_{j \in M} c_{jt}^{buy} x_{jt}^{buy} - \sum_{j \in M} c_{jt}^{inv} x_{jt}^{inv} - \sum_{(j,j') \in M^{conv}} c_{jj't}^{conv} x_{jj't}^{conv} - \sum_{(i,k) \in F^{act}} c_{ikt}^{act} x_{ikt}^{act}$$

$$- \sum_{i \in F} c_{it}^{cap} x_{it}^{cap}$$

**Constraints**

For each $j \in M$, r$\in R$ and $t \in T$, the amount of material $j$ made available by purchases, production, conversions and beginning inventory must equal the amount used for sales, production, conversions and ending inventory:

$$x_{jt}^{sell} + \sum_{(i,j)\in F^{out}} x_{ijt}^{out} + \sum_{(j',j)\in M^{conv}} \alpha_{j'jt}^{conv} x_{j'jt}^{conv} + x_{jt-1}^{inv} = x_{jt}^{sell} + \sum_{(i,j)\in F^{in}} x_{ijt}^{in} + \sum_{(j,j')\in M^{conv}} x_{jj't}^{conv} + x_{jt}^{inv}$$

For each $(i,j)\in F^{in}$ and $t\in T$, the amount of input $j$ used at facility $i$ must equal the total consumption by all the activities at facility $i$:

$$x_{ijt}^{in} = \sum_{(i,j,k,t)\in A^{in}} \alpha_{ijkt}^{in} x_{ikt}^{act}$$

For each $(i,j)\in F^{out}$ and $t\in T$, the amount of output $j$ produced at facility $i$ must equal the total production by all the activities at facility $i$:

$$x_{ijt}^{out} = \sum_{(i,j,k,t)\in A^{out}} \alpha_{ijkt}^{out} x_{ikt}^{act}$$

For each $i\in F$ and $t\in T$, the capacity used by all activities at facility $i$ must be within the range given by the lower limit and the upper limit plus the amount of capacity vendored:

$$l_{it}^{cap} \le \sum_{(i,k)\in F^{act}} x_{ikt}^{act} / r_{ikt}^{act} \le u_{it}^{cap} + x_{it}^{cap}$$

For each $j\in M$, the amount of material inventoried in the plant before the first time period is defined to equal the specified initial inventory:

$$x_{j0}^{inv} = v_{j0}$$

For each $j\in M$ and $t\in T$, the total amount of material $j$ inventoried is defined as the sum of the inventories over all storage areas:

$$\sum_{s\in S} x_{jst}^{stor} = x_{jt}^{inv}$$

For each $s\in S$ and $t\in T$, the total of all materials inventoried in storage area $s$ must be within the specified limits:

$$l_{st}^{stor} \leq \sum_{j \in M} x_{jst}^{stor} \leq u_{st}^{stor}$$

All variables must lie within the relevant limits defined by the data:

$l_{jt}^{buy} \leq x_{jt}^{buy} \leq u_{jt}^{buy}$, for each $j \in M$ and $t \in T$

$l_{jt}^{sell} \leq x_{jt}^{sell} \leq u_{jt}^{sell}$, for each $j \in M$ and $t \in T$

$l_{jt}^{inv} \leq x_{jt}^{inv} \leq u_{jt}^{inv}$, for each $j \in M$ and $t \in T$

$0 \leq x_{jj't}^{conv}$, for each $(j,j') \in M^{conv}$ and $t \in T$

$0 \leq x_{it}^{cap}$, for each i$\in F$ and $t \in T$

$0 \leq x_{jst}^{stor}$, for each s$\in S$, $j \in M$ and $t \in T$

$l_{ij}^{in} \leq x_{ij}^{in} \leq u_{ij}^{in}$, for each $(i,j) \in F^{in}$ and $t \in T$

$l_{ij}^{out} \leq x_{ij}^{out} \leq u_{ij}^{out}$, for each $(i,j) \in F^{out}$ and $t \in T$

$l_{ik}^{act} \leq x_{ik}^{act} \leq u_{ik}^{act}$, for each $(i,j) \in F^{act}$ and $t \in T$

**References**


[1] Adams, D., Beckett, D., 1999. Programming in 4th Dimension, the Ultimate Guide. Automated Solutions Group: Huntington Beach, CA, USA.

[2] Date, C. J. (1981), Introduction to database systems, 3rd Edition, *Addison-Wesley Publishing Company*

[3] Dolk, D. R. (1986), Data as Models: An Approach to Implementing Model Management, *Decision Support Systems*, 2 (1)

[4] Dolk, D. R. (1986a), A Generalized Model Management System for Mathematical Programming, *ACM Transactions on Mathematical Software*, 12 (2), pp 92 -126

[5] Dominguez – Ballesteros, B., Mitra, G., Lucas, C., & Koutsoukis, N- S. (2002), Modeling and Solving Environments for Mathematical Programming (MP): A Status Review and New Directions, *Journal of Operational Research Society*, 53, pp1072 – 1092

[6] Dutta G., & Fourer R., A survey of mathematical programming applications in an integrated steel plant, Manufacturing & Service Operations Management, 3(4) (2001), 387-400.

[7] Dutta, G., & Fourer, R. (2004), An Optimization-Based Decision Support System for Strategic and Operational Planning In Process Industries, *Optimization and Engineering*, 5, pp 295 – 314

[7] Dutta G., & Sinha G. P, & Roy P.N, and Mitter N., A linear programming model for distribution of electrical energy in a steel plant, International Transactions in Operational Research, 1(1) (1994) 17-29.

[8] Fourer, R. (1983), Modeling Language Versus Matrix generator for Linear Programming, *ACM Transactions on Mathematical Software*, 9 (2), pp 143 – 183

[9] Fourer R., Database structure for mathematical programming models, Decision Support Systems, 20 (1997) 317-344.

[10] Geoffrion A. M., Computer-based modeling environments, European Journal in Operations Research, 41 (1989) 33-43.

[11] Marsten R.E., The design of XMP linear programming library, ACM Transactions on Mathematical Software, 7 (1981) 481- 497.

[12] Murphy, F. H., Storh, E. A., & Asthana, A (1992), Representation Scheme for Linear Programming Models, *Management Science*, 38 (7), pp 964 – 991

[13] SCH Leung, Y. Wu., & Lai K. K. (2006), A Stochastic Programming Approach for Multi-Site Aggregate Production Planning, *Journal of Operational Research Society*, 57, 123-132

[14] Sinha G. P., Chandrashekaran B.S., Mitter N., Dutta G., Singh S.B., Roy P.N., & Roy Choudhary A., Strategic and operations management with optimization at Tata Steel, Interfaces, 25 (1) (1995) 6-19.

**Figures**

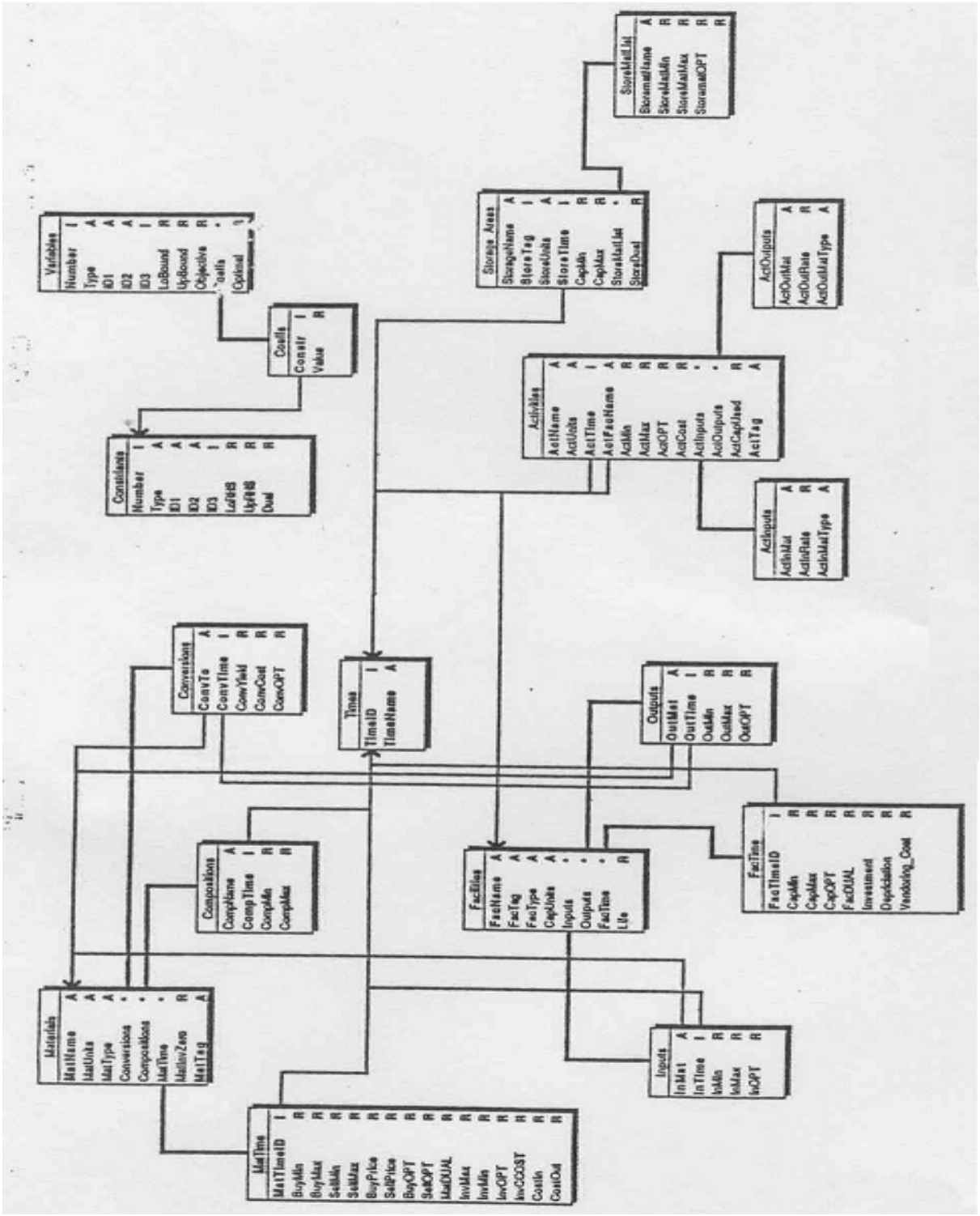


Figure 1: Database structure for STEEL-TIME

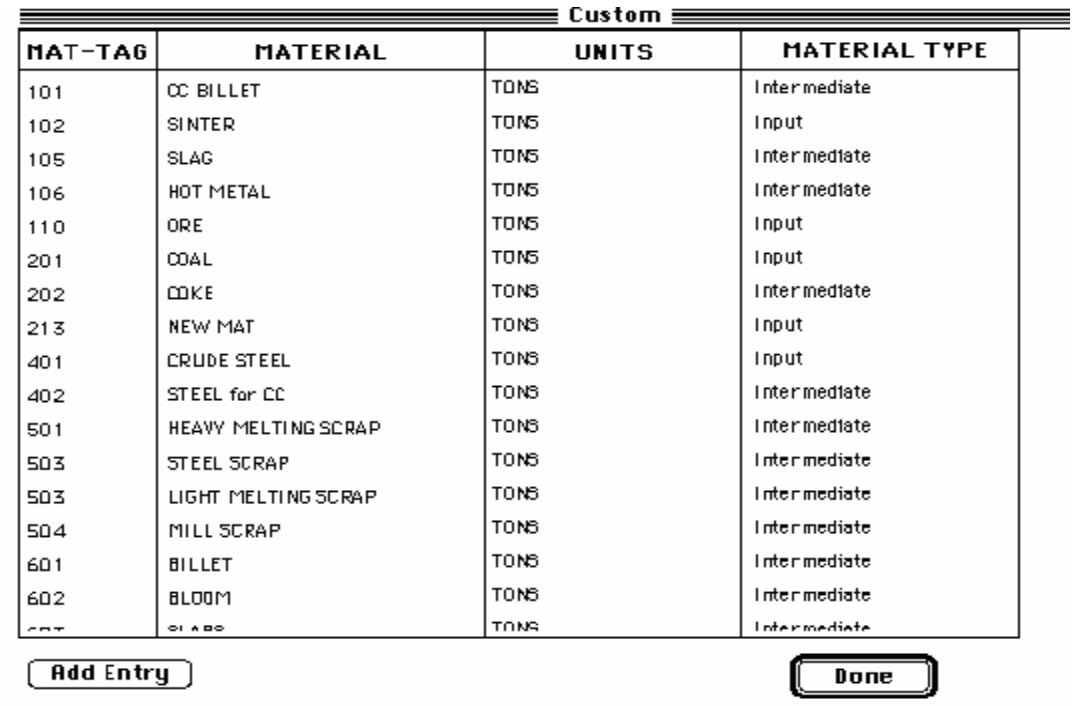


Figure 2: Output Layout of Materials File

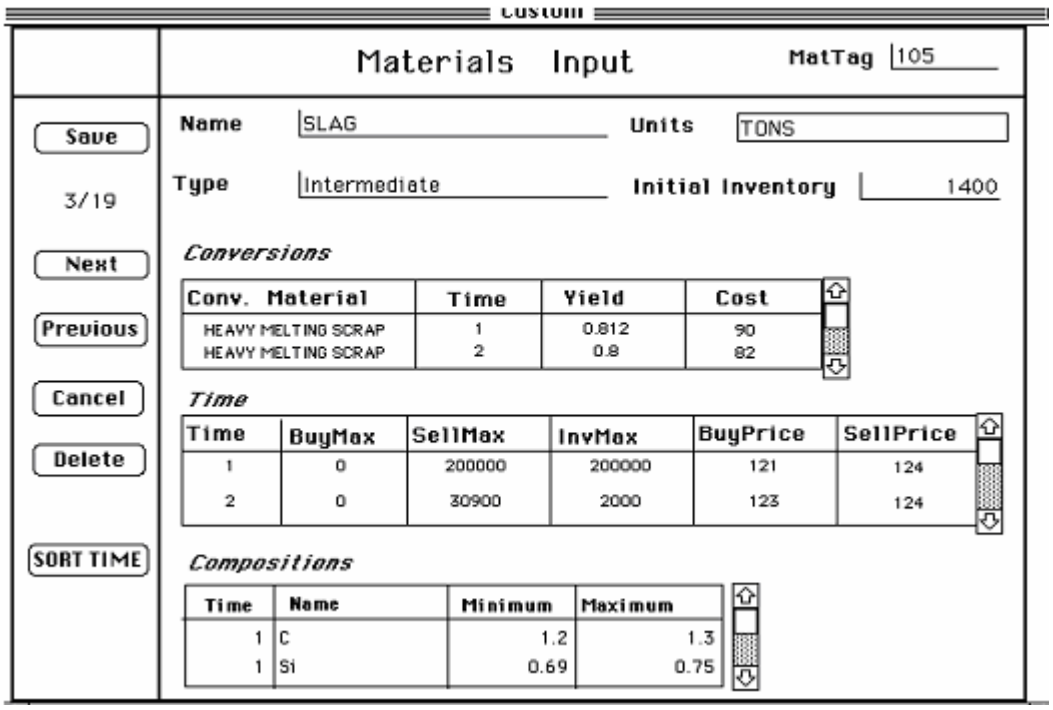


Figure 3: Input Layout of Materials File

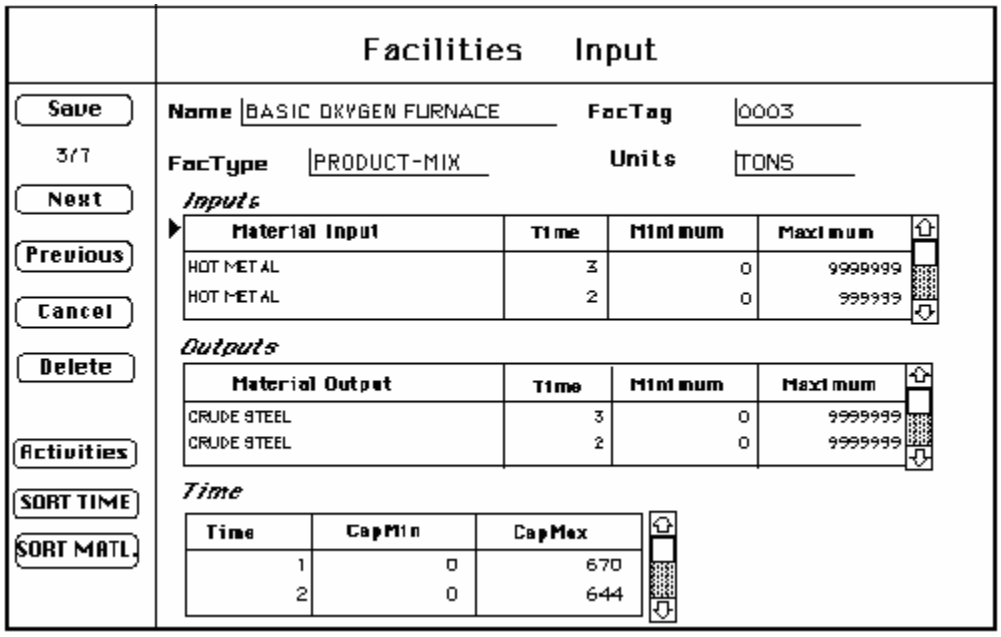


Figure 4: Input Layout of Facilities File

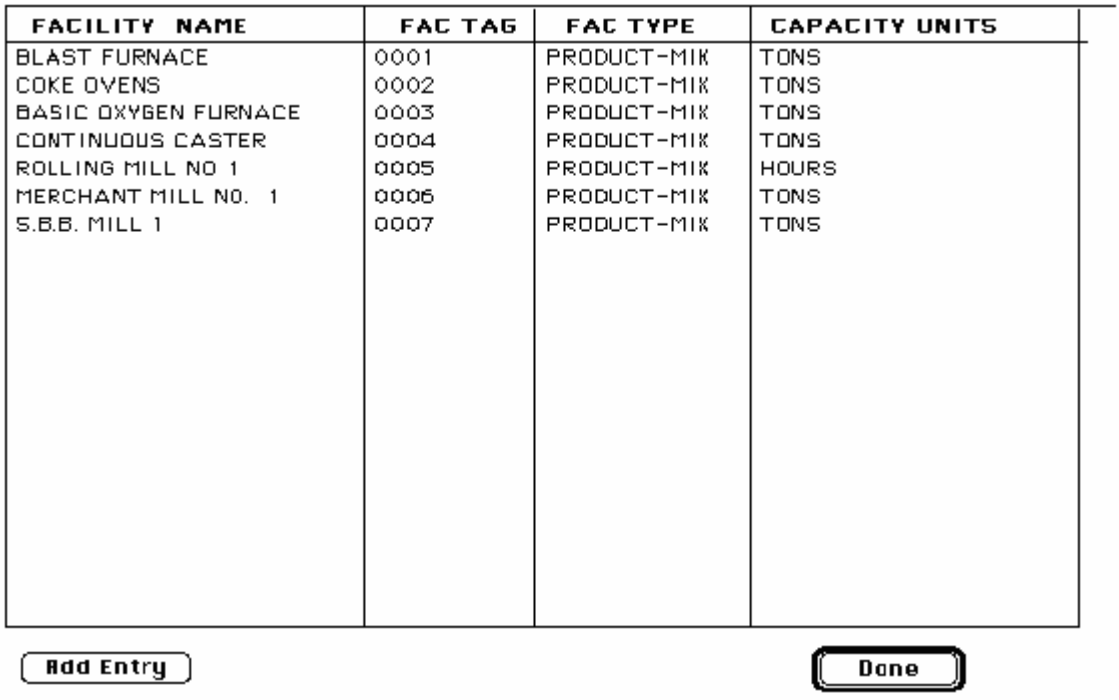


Figure 5: Output Layout of Facilities File

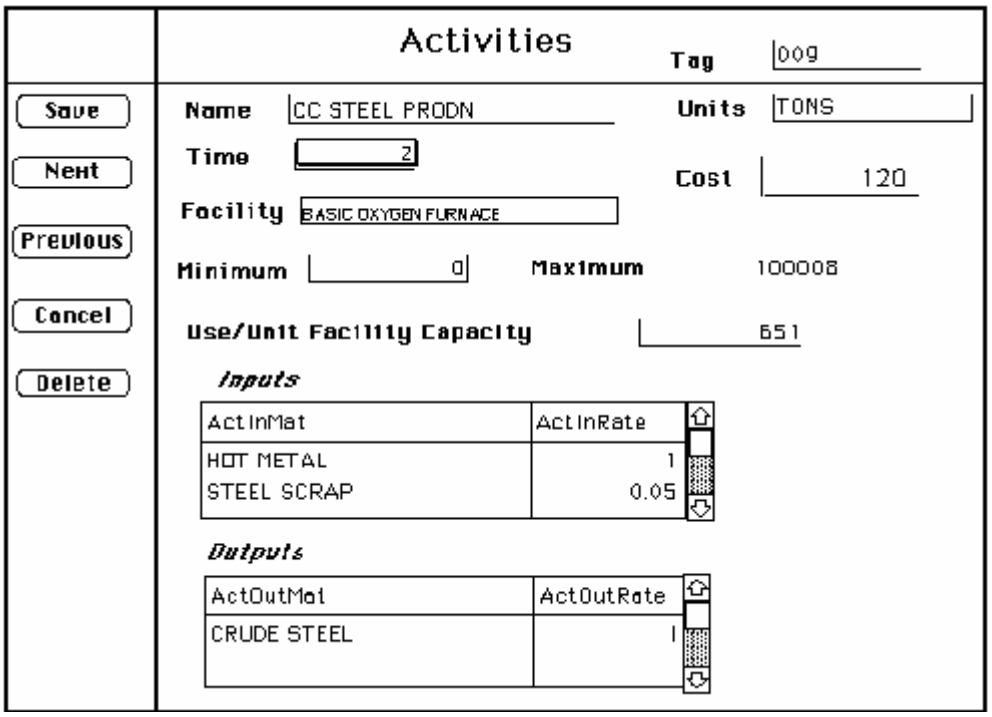


Figure 6: Input Layout of Activities File

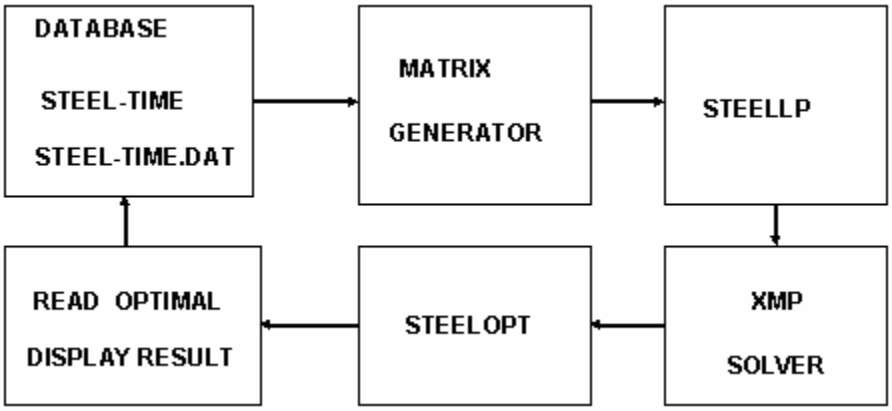


Figure 7: Optimization Steps

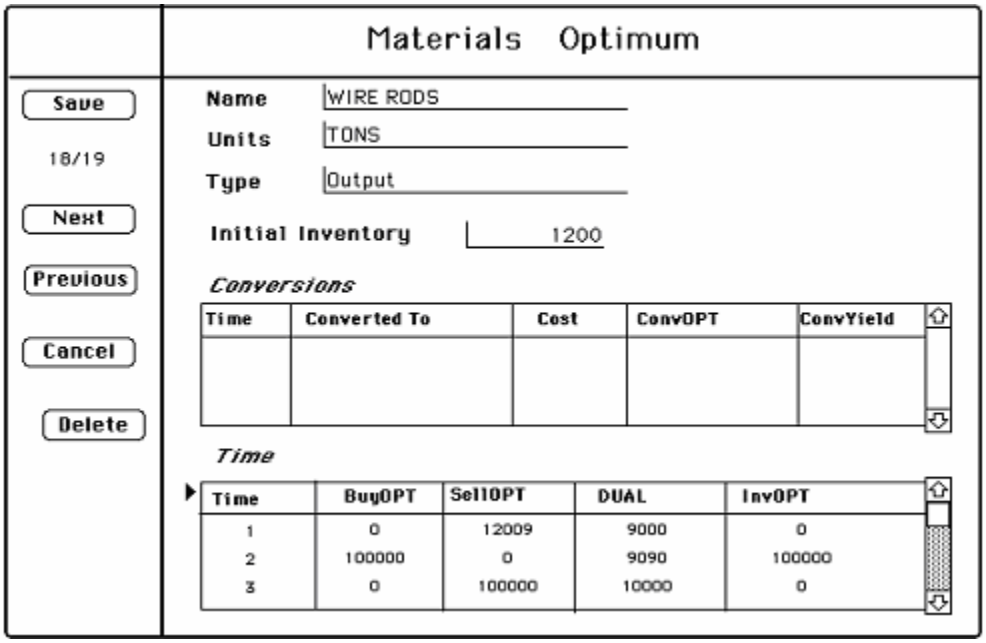


Figure 8: Materials Optimal

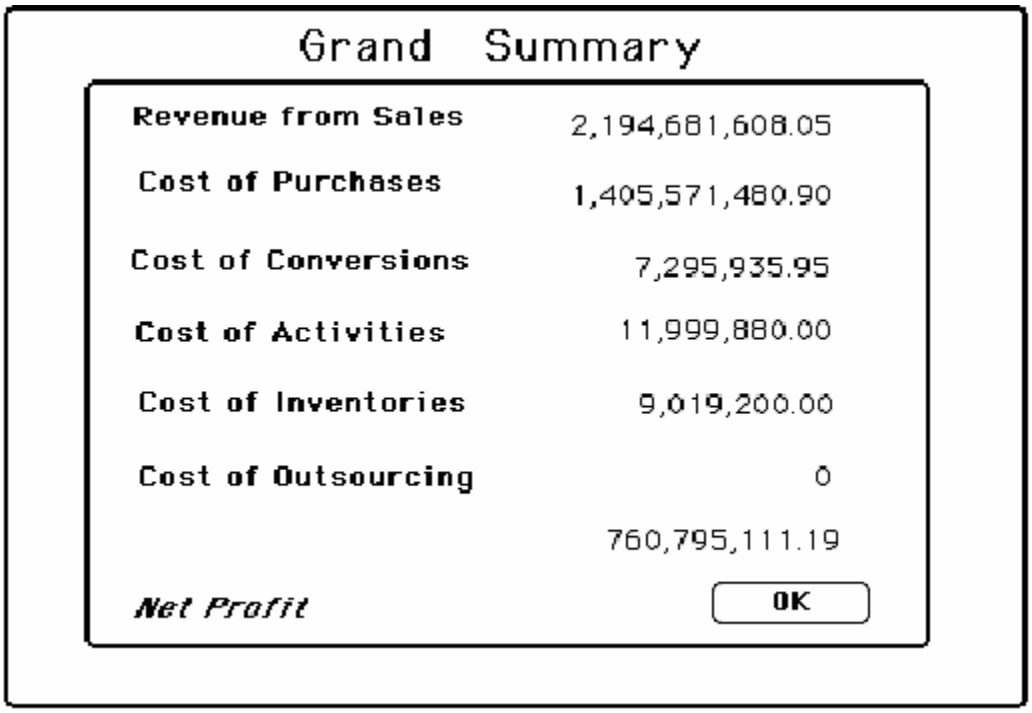


Figure 9: Grand Summary

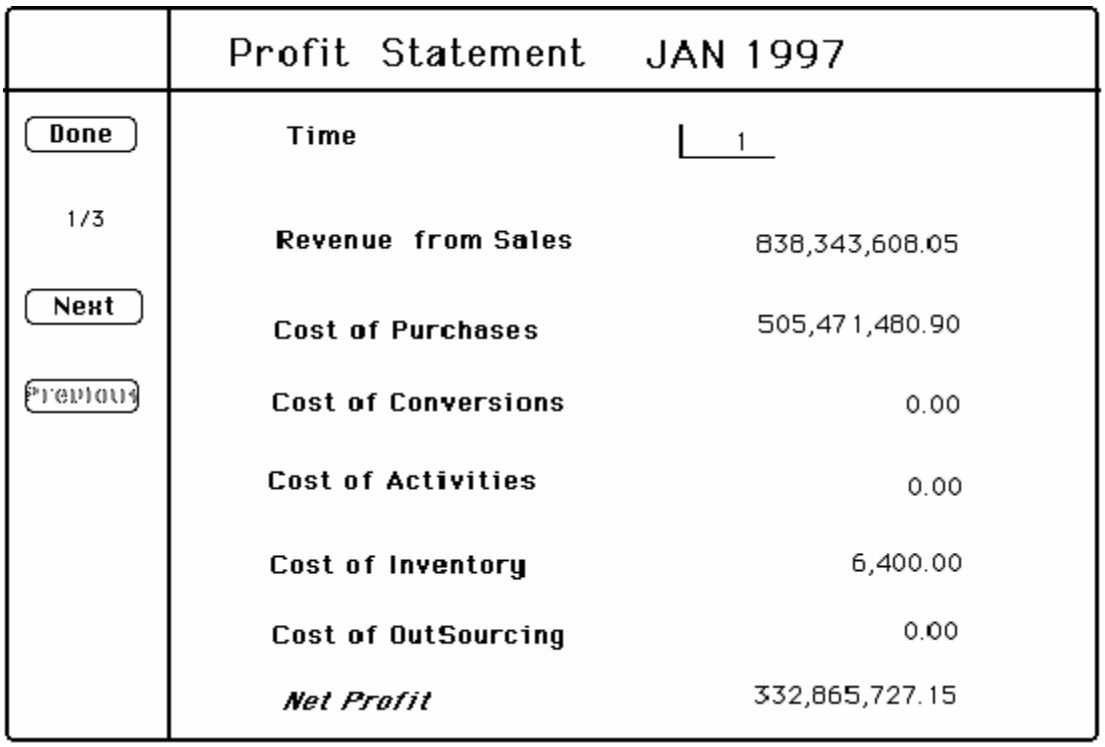


Figure 10: Profit Statement of One Time Period

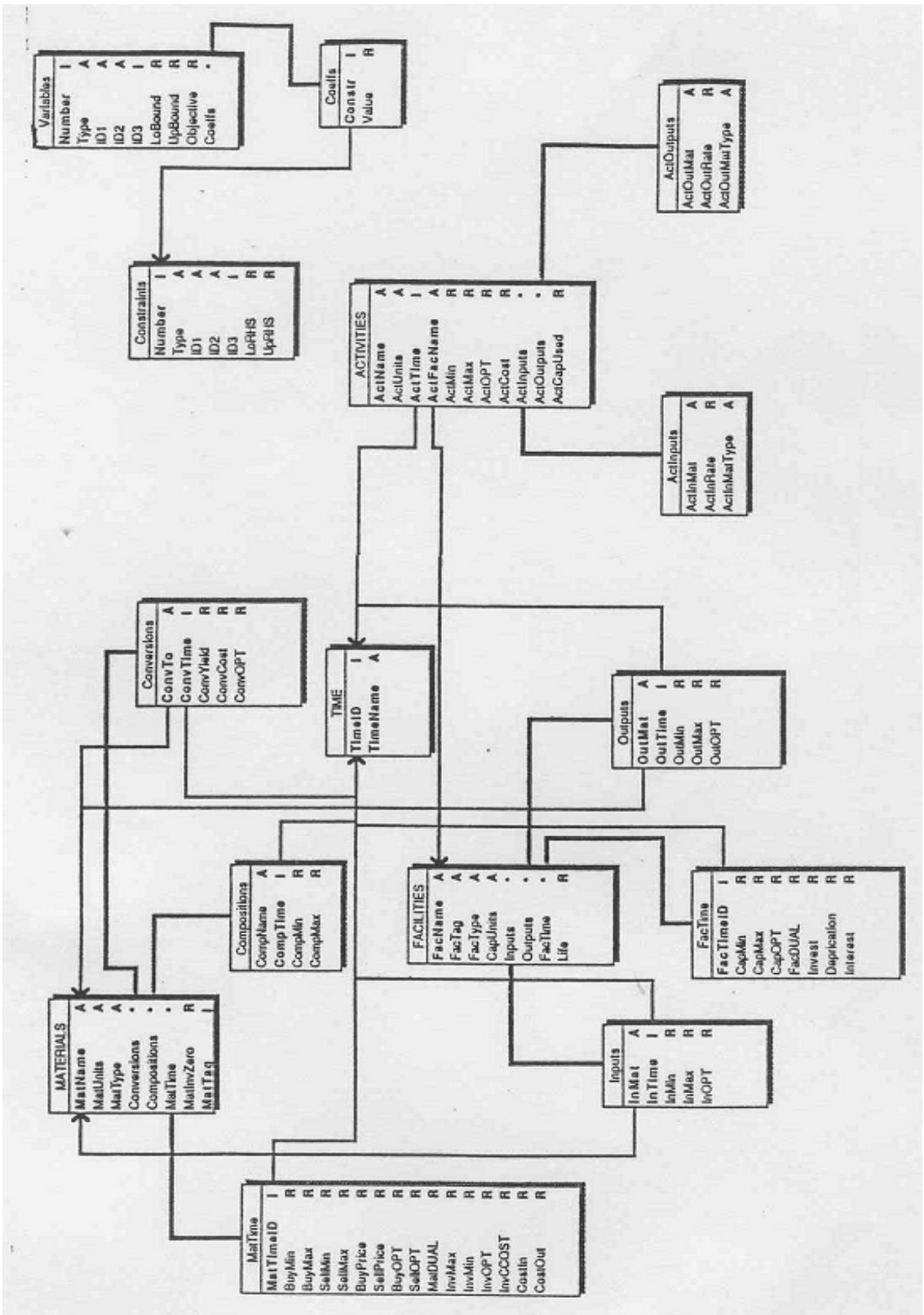


Figure 11: Database Structure of STEEL-TIME1

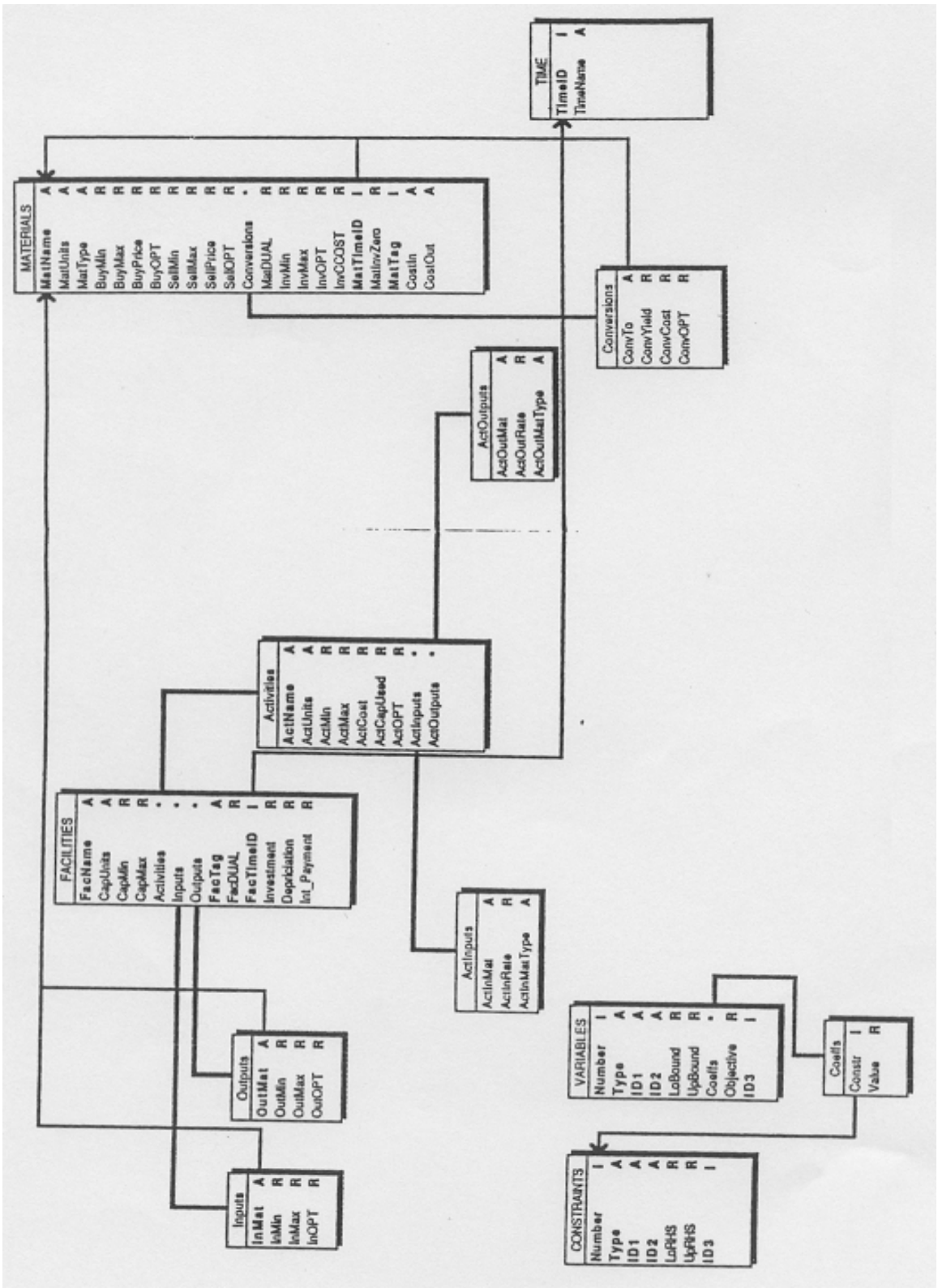


Figure 12: Database Structure of STEEL-TIME2