INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 1349134

Optimization of the packaging of the Mars oxygen manufacturing plant

Santhanam, Venkatesan, M.S.

The University of Arizona, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

OPTIMIZATION OF THE PACKAGING OF THE MARS OXYGEN MANUFACTURING PLANT

by

Venkatesan Santhanam

A Thesis Submitted to the Faculty of the

DEPARTMENT OF AEROSPACE AND MECHANICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN MECHANICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1992

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: S. Vanderbeam

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

A. Arabyan 7/10/92

Ara Arabyan
Associate Professor of Aerospace and
Mechanical Engineering

Date

ACKNOWLEDGMENT

I thank Professor Ara Arabyan for his patience in seeing this project through to the end. Without his help I would never have done this work. I would like to extend my thanks to Professor K.R. Sridhar and Professor P.E. Nikravesh for their time and patience in reviewing this report. My special thanks goes to people in the C.C.I.T. - Research support, particularly Barry Richards, Sanjay and John Lee for their help and suggestions without which I would have found it very difficult to complete this work. I extend my thanks to the N.A.S.A. - Space Engineering Research Center for sponsoring this project. Finally, I thank my parents, other family members and my relatives in the United States especially my brother Sampath Kumar Santhanam for their blessings and encouragement which took me through this ordeal. Above all I thank Lord Sri Venkateswara for helping me throughout my career.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

ABSTRACT

A complete analysis of the various energy losses involved in the Mars oxygen manufacturing plant is performed. The various losses considered are the pressure losses and radiation losses in connections between components, radiative heat losses from and between different components of the plant. These, together with the cost and mass of a library of components are used to construct an objective function to optimize the packaging of the plant. A software package has been created to determine component locations in the package such that this objective function is minimized. The package reads input data from a component database system and performs various iterations to arrive at a configuration having the minimum of the energy losses. The steepest descent method is used for minimizing the nonlinear objective functions describing the energy losses. The minimization of the cost and mass factors are performed by a brute search method. The results of the optimization are presented graphically.

CHAPTER 1

INTRODUCTION

The design of complex mechanical systems which are required to operate reliably and autonomously in alien environments is an intricate and iterative process which requires extensive testing in a wide range of hypothetical scenarios. Given the impossibility or the high cost of recreating alien environments on earth, such scenarios have to be generated by computers and applied to simulated versions of the actual mechanical systems. As a result, computer simulation has become an essential and integral part of designing and building mechanical systems to be used in space or extraterrestrial environments. This specific project was initiated with the aim of providing the essential simulation capability for the oxygen manufacturing plant that the Space Engineering Research Center at the University of Arizona has undertaken to build on Mars.

Recent advances in computer hardware and software techniques offer opportunities for the creation of large-scale engineering design systems that were once thought to be impossible or impractical. However, creation of such systems is usually a large and complex project. Furthermore, these systems require frequent modifications and extensions because of the continuing developments in engineering theories and practice.

Confronted with such a massive and complex project, software developers have been attempting to devise systematic approaches that would complete the software system while maintaining its understandability, modifiability, reliability, and efficiency. Considerable effort has been made toward achieving these goals through database management and software design techniques.

1.1 Project outline

A database system has been developed which feeds the data to a common working reservoir which in turn is used by various subprograms to perform specific operations related to the project. The project can also interact with various sub projects in various other fields (e.g. a sub project could be the project which involves the area of artificial intelligence for the autonomous production of oxygen on Mars).

The overall objective function (to be minimized) depends on the location of the components, mass and cost of models of such components (i.e. a model is a particular kind of a component) and the size and material of models of connection between these components. Inorder to obtain an optimal solution (i.e. an optimal plant configuration) of this objective function, a search should be adopted such that all component models are placed at all possible locations in the plant with all available component and connection models combinations. In other words, every component and connection model must be used with every other component and connection model at all possible locations in the plant. The number of iterations required to find the optimal solution in this way will rise in an exponential way. This will result in an NP-hard problem which cannot be solved in finite time. Thus a mechanism is adopted to breakdown the current problem into subproblems (whose variables take discrete and continuous values) which are optimized to reach a suboptimal solution.

A novel approach has been used for placing the components at discrete regions instead of placing the component at all possible locations. The resulting plant configuration is checked so that it satisfies certain requirements. The plant configuration should not have components that are physically interfering with each other. A new idea has been adopted which would separate the interfering component configuration into an noninterfering one.

1.2 Review of past work on design database optimization

Evans (1991)[1] succeeded in developing an integrated design database system for the Architecture and Engineering Series (AES). The software provided three dimensional graphics for modelling and drafting. The software package was written in the C language. Vassilev *et al* (1991)[2], developed an user-friendly interface system which could optimize constrained and unconstrained nonlinear problems. However, the developed system did not use database files as its data source. Chiu *et al* (1990)[3] devised a model for the optimal database allocation of the resources in certain computing systems. The objective of this method was to optimize the total operation cost. Hirano *et al* (1991)[4] introduced a new method which combines knowledge engineering technology with mathematical programming procedures. The method performs heuristic controls over optimization algorithms. The key concept involved in the heuristic control is the qualitative classification of design variables. Sadeghi (1984)[5] also used heuristic optimization with interactive graphics approach in the design of control systems. A data storage and retrieval system has been designed by Mulleneers *et al* (1989)[6] to optimize information for a biological analysis. The database system was built using a database programming language called Clipper. The system could store and retrieve data from a unique data source and was not integrated with other systems.

1.3 Thesis overview

The various chapters in this thesis deal with the different features of the development of a package for the Mars oxygen manufacturing plant and the optimization of the developed package. Each of these chapters are discussed in detail below.

Chapter 2 discusses the basic features of the Mars oxygen manufacturing plant. Various proposed schemes are presented for the production of oxygen in Mars. The proposed schemes

are compared on the basis of various criteria and their merits and demerits are then discussed. The need for computer simulation is emphasized in order to compare the various schemes in an exhaustive manner.

Chapter 3 deals with the design of a uniform database system which acts as a data reservoir for all the subprograms that are linked to it. A general theory on the management of databases is presented. The various capabilities of the database system are then discussed. The process of adding, editing or deleting data for a component or a connection model is also analyzed. This chapter also explains the procedures involved in building a computer representation of a user-defined oxygen manufacturing plant.

Chapter 4 discusses the optimization of various objective functions related to the plant. The integration of the optimization process with the database system is also discussed. A general theory on optimization and its classification is presented. Different numerical methods for nonlinear programming are then discussed. The mathematical formulation of the various objective functions related to the plant is then presented. The determination of the optimal value of these objective functions is then analyzed. In addition a novel method is presented for dividing the plant into n regions.

Chapter 5 discusses the results obtained after the optimization of the various objective functions. The results are graphed for a 5-component system and a 9-component system. The graphs show the way in which values of the various objective functions decrease as the number of iterations increase. The execution time taken for various number of component and connection models is also graphed. Then conclusions and directions for future work are discussed.

CHAPTER 2

THE MARS OXYGEN MANUFACTURING PLANT

Cost has been recognized as a significant determining factor in space exploration. The single largest cost factor cost is associated with the initial launch. Despite valiant efforts by NASA, ESA and Arianspace and the promise of cheaper services from other countries, realistic projections show a launch cost not less than $500-$1,000/lb. While this improvement is praiseworthy, it is still not sufficient for frequent space explorations. More than 80% of a spacecraft's mass is due to the propellant. Therefore the integral part of the cost factor is the propellant mass. All the propellants are presently carried from the Earth as there is no means for propellant production elsewhere. Hence the production of propellants at locations remote from the Earth is essential for frequent and extended space explorations. This requires in-situ resource utilization (ISRU) whereby only a small fraction of the key hardware and consumables are launched from Earth and the rest are extracted/processed/built/assembled from *local* resources extraterrestrially.

Scientific missions to Mars (Viking I and II) have sent back intriguing data which reveal the possibility of manufacturing propellants from available resources on Mars. This was one of the basic motivations for a mission to Mars.

2.1 Propellant processing

Ramohalli *et al* (1991)[7] found out that a large fraction of any propellant is the oxidizer. They obtained this result by examining various propellant combinations. Since oxygen is needed

for a range of other purposes (e.g., life support), the production of oxygen can have a significant impact on launch mass (cost) reductions.

Oxygen can be extracted from any compound of oxygen provided sufficient energy (of the right type) is available. However, because of the critical importance of launch weight, it is imperative to minimize the plant size and therefore maximize the rate of oxygen production and yield efficiency. As a result of these considerations, various different schemes have been proposed for extraterrestrial oxygen production.

2.2 Various schemes for oxygen production

The schemes that have been proposed for extraterrestrial oxygen production primarily differ in their oxygen sources. These sources include carbon dioxide found in the atmosphere, water from ice caps, water-ice at the poles, water from the soil, water collected en route, Martian permafrost and various oxides on the moon or extraterrestrial resources. Production of oxygen from the Martian atmosphere is the present area of interest because of the following main reasons. Autonomous equipment can be tested in detail in simulated Martian conditions quite economically on Earth because Mars is one of the few planets whose conditions are simulated with relative ease. Various schemes have been proposed with choices in oxygen resources and overall system design. Different options have been suggested for the production of oxygen from the Martian atmosphere for spacecraft propulsion and/or life-support. One attractive option for oxygen production from Martian atmosphere is the use of electrochemical separation cell technology. The oxygen cell technology separates oxygen from other gases by means of a solid electrolyte maintained at approximately 1000 °C. The oxygen cell uses zirconia as the cell material.

Three major options have been proposed by Ramohalli *et al* (1989)[8], for the production of oxygen on Mars, using the oxygen cell technology. They are called as Option I, Option II and Option III as shown in Figure 2.1, Figure 2.2 and Figure 2.3.

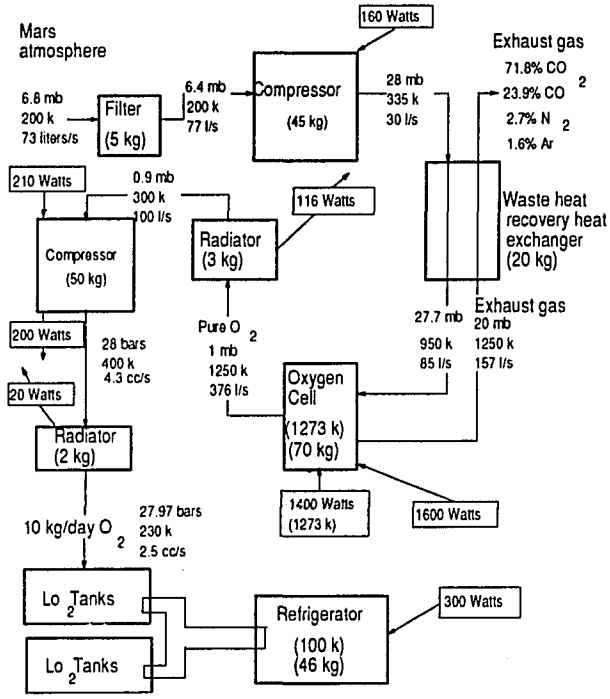

Figure 2.1 Schematic presentation of the oxygen manufacturing plant - Option I

All these options have the common feature that they take Martian atmospheric gas, compress it to a higher pressure, heat it to a temperature at which oxygen can be separated, and use zirconia membrane technology. The oxygen is then compressed (usually liquified) and stored for use in the propulsion module. Beyond this common design, the details are different for the three options. A description of the generic features in the Mars oxygen manufacturing plant is as follows:

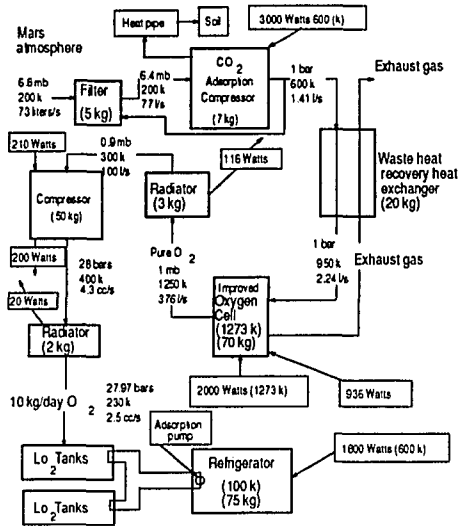

Figure 2.2 Schematic presentation of the oxygen manufacturing plant - Option II

2.3 Preparatory sub-system

The Martian atmosphere contains carbon dioxide, dust particles and other extraneous gases. This atmospheric gas requires purification before oxygen can be separated from it. The filter, compressor and heat exchanger form the *carbon dioxide preparatory sub-system*, which transforms the Martian atmosphere to a gas which contains carbon dioxide as its primary constituent. Each of these components of the preparatory sub-system is explained in greater detail below.

Dust and other extraneous particles present in the Martian atmosphere are removed by a filter. The intake of the filter is the Martian atmosphere at a pressure of 6.8 mb and 200 °k. The outlet of the filter is at 6.4 mb and 200 °k. The outlet of the filter is compressed to 28 mb by a compressor. The pressurized Martian atmosphere is then passed to a waste heat recovery heat

exchanger to gain energy from the gases exhausting from a zirconia cell (ZrO_2). Thermal decomposition of carbon dioxide then occurs in the heat exchanger. The products of the thermal decomposition are carbon dioxide, carbon monoxide and oxygen. For more information on the preparatory sub-system please refer Venkatesh *et al* [9].

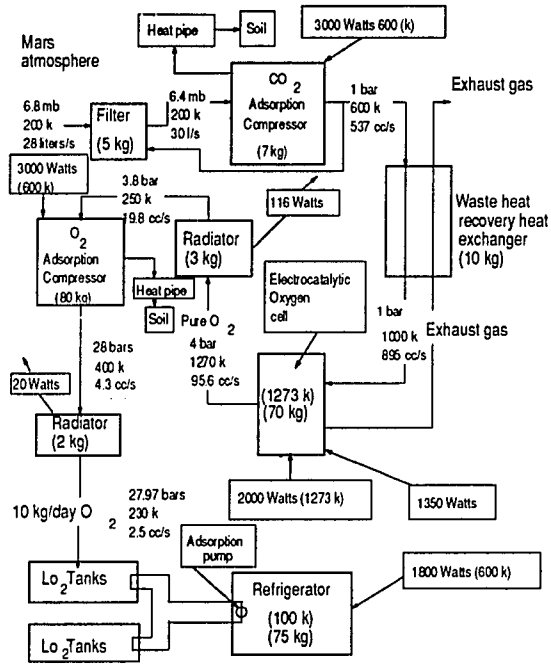

Figure 2.3 Schematic presentation of the oxygen manufacturing plant - Option III

2.3.1 Oxygen cell

The heart of the plant is the oxygen cell that pumps (electrochemically separates) oxygen from carbon dioxide by means of a solid electrolyte maintained at approximately 1,000 °C. Zirconia is one of the materials on which data is available. This was the primary reason which led to the selection of zirconia as the cell material.

Through experimental analysis it has been found that carbon deposition will not be a problem as would be expected theoretically from the relative bond strength of the compounds. For more information on oxygen cell please refer Ramohalli *et al* (1991)[10] and Ramohalli *et al* (1991)[7], .

2.3.2 Radiator

The function of the radiator is to bring the temperature of the hot oxygen from the oxygen cell down to a point where it can be efficiently adsorbed by the oxygen adsorption compressor. The radiator brings down the temperature of the oxygen from 1250 °K to 300 °K .

2.3.3 Oxygen compressor, tanks and refrigerator

The oxygen exiting the radiator is compressed from 0.9 mb to 28 bar by an oxygen compressor. Adsorption compressors are also used to increase the efficiency of the compressor. This kind of compressor is still in its development stage. For more information on adsorption compressors please refer Ramohalli *et al* (1989)[8].

The oxygen-radiator radiates heat to the Martian atmosphere and cools the oxygen gas from 400 °K to 230 °K . The liquified oxygen is stored in tanks. The tanks are further cooled by a refrigerator which cools the gas from 230 K to 100 K.

2.4 Study of the various proposed plant options

Each option has its own advantages and disadvantages. The filter in Option I can be clogged especially during frequent dust storms. A cleaning or replacement method is required for these kind of filters. The filter proposed in Option II can be cleaned by periodic blasts of carbon

dioxide from the adsorption compressor. This filter requires proper design and testing. The filter for Option III is the same as the one for Option II. All the options have a filter which weighs about 5 kg.

Option I has a compressor (carbon dioxide) which is designed to run unattended for 4800+ hrs in dust storms. This compressor requires a special design. The compressor in Option II is a nitrogen adsorption compressor developed at the (Jet Propulsion Laboratory J.P.L.). This compressor can pump carbon dioxide easily using Zeolite 13X. Option III has a compressor similar to the one in Option II. The compressor of Option I has a weight of 45 kg and the compressors of Option II and Option III each weigh 7 kg. The energy input for the compressor in Option I is 160 watts whereas it is 3000 watts for the one in Option II and 1200 watts for the one in Option III.

The heat exchanger in Option I is still in its development stage. The heat exchanger in Option II is a smaller and lighter heat exchanger. Option III has a heat exchanger which is similar to the one in Option II. The Option I heat exchanger weighs 20 kg whereas the heat exchanger in Option II and Option III each weigh 10 kg.

The oxygen cell proposed in Option I is still in research stage, with development needed to demonstrate durability of seals and membranes, suitable electrodes, and adequate current density. Option II has an oxygen cell with improved electrodes which reduce the voltage drop leading to greater efficiency. Higher current densities allow more compact design with greater redundancy. An electrocatalytic oxygen cell is proposed in Option III (same as Option II) but can operate at slightly higher voltages to reduce carbon dioxide electrocatalytically to oxygen with a higher yield. All options have oxygen cells which weigh 70 kg.

The first oxygen-radiators in Option I and Option II are essentially the same and they are still in their development stage. Option III has an oxygen-radiator whose size is different from the other options. All options have radiators which weighs 3 kg.

The oxygen-compressor in Option I is a multistage compressor which takes oxygen at about 1 mb and compresses it to 28 bar. Option II has a compressor which is essentially the same as in Option I. Option III has a compressor which is of the adsorption type. It is similar to the carbon dioxide adsorption compressor except for the adsorption material. The compressors in Option I and Option II weigh 50 kg each. Option III has an compressor which weighs 80 kg.

The design of the second oxygen cell and the Lithium oxide tank do not change in any of the options. The molecular adsorption cryocooler (MACC) weighs 46 kg in Option I whereas the one in Option II and Option III weigh 75 kg each. For more information on the proposed plant options please refer Ramohalli *et al* (1989)[8].

2.5 Performance risk factors

Every component in these options has a performance risk factor associated with it. These risk factors indicate the advantage of one option over the other. The development risk factors proposed by Ramohalli *et al* (1989)[8], for all the components in all the considered options are shown in Table 2.1, Table 2.2 and Table 2.3. Option III has the advantage of eliminating all rotary machinery. This decreases the overall risk factor for Option III.

Component	Risk factor
Filter	3
Compressor (CO_2)	4
Heat exchanger	1
Oxygen cell	1
Radiator	1
Compressor (O_2)	4
Radiator	1
LOX tanks	1
Refrigerator (Simple)	3
MACC Refrigerator	3

Table 2.1 Development risk factors for components of Option I

Component	Risk factor
Filter	2
Adsorption Compressor (CO_2)	3
Heat exchanger	1
Oxygen cell	4
Radiator	1
Compressor (O_2)	4
Radiator	1
LOX tanks	1
MACC	3

Table 2.2 Development risk factors for components of Option II

Component	Risk factor
Filter	2
Adsorption Compressor (CO_2)	3
Heat exchanger	1
Oxygen cell	4
Radiator	1
Adsorption Compressor (O_2)	3
Radiator	1
LOX tanks	1
MACC Refrigerator	3

Table 2.3 Development risk factors for components of Option III

CHAPTER 3

DESIGN OF A UNIFORM DATABASE SYSTEM

An integrated database system for the proposed Mars oxygen manufacturing plant has been developed. The database system provides a uniform user interface to enter different types of data about components in the plant and their connections. It also exports and imports data to and from other programs.

The principal components of the Mars oxygen manufacturing plant include a filter, high pressure compressor, heat exchanger, oxygen cell, low pressure compressor, high temperature heat radiator, low temperature heat radiator and refrigerator. Each of these components is connected to other components by different connections such as mechanical, fluid, electrical or any other connections. The plant can be described by specifying each component and the way in which they are connected with each other.

3.1 Database and database management

A database is essentially a collection of related data. To use a database, software can be written that can perform the tasks of accessing information, sorting them out and modifying them if necessary. A software of this type which acts as an interface between the user and the data is called a *Database Management System.*

3.1.1 Databases

A database is a table of data consisting of rows and columns. A row, also called a *Record* consists of many different pieces of information about a particular item. A column also called a *Field* consists of a class of data corresponding to each one of the records.

3.1.2 Management of databases

Foxpro 1.02 is a program developed for data management - the cataloging, tracking and processing of information. Foxpro provides speed, dBASE IV compatibility, and an outstanding environment for the development of business applications. Foxpro has capabilities to provide multiple windows, pull-down menus and mouse support.

Data storage and data retrieval are simple but tedious operations. Thus they constitute ideal tasks for a computer. Typically the process of data retrieval and modification involves the following essential operations:

1. Open a database file and make it available.

2. Close databases and make them inactive. No data can be stored or retrieved from a closed database.

3. Select a database to activate an already open database file.

4. Index a database: An index file can be opened to define the order in which the records occur. This however does not affect the physical order of the records in the database. A database can be indexed in an alphabetical order or in a numerical order.

5. Skip: The Skip command moves the pointer from the current record position to a specified number of records.

6. Go: Go command is used to move the record pointer to a specified record number.

7. Seek: The Seek command is used to search a selected indexed database for the record which has the same index key expression as the given expression. For example, if a database is indexed on a field in a numerical order, one can seek the record corresponding to a particular number in that field. This command is also called as Search command.

8. Append: The Append command adds blank records to the currently selected database.

9. Replace: The Replace command replaces a value in a database with a specified value.

10. Delete: The Delete command marks the selected record for deletion but does not actually delete the record.

11. Pack: The Pack command removes records that have been marked for deletion by the Delete command. For more information please refer Jones (1990) [11].

3.2 Generic integrated system

An integrated system is required which would work on a generic database and would compare the various proposed oxygen manufacturing schemes on common grounds. Because of the multidisciplinary nature of the plant's design process, this integrated software system, illustrated in Figure 3.1, is designed to provide data storage, classification, access and visualization services for all the different groups working on the design and testing of the plant.

The database system is integrated with an animation program which, in different modes, provides visual feedback to designers and researchers about the location of and temperature distribution among components as well as heat, mass and data flow through the plant as it operates in different scenarios; a control program to investigate the stability and response of the system under different disturbance conditions; an optimization program to maximize or minimize various criteria as the system evolves into its final design; a graphics program which portrays the

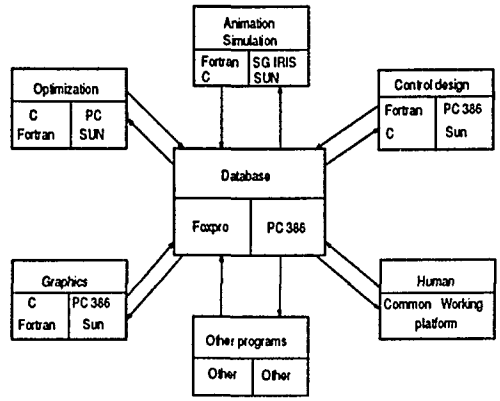

Figure 3.1 Generic integrated system

various aspects of the design process; and a common platform for the user to activate the various modules of the overall system.

3.3 Uniform database system

This system provides a uniform data entry system which can act as a server of information to other components of the project in the appropriate format after conducting basic checks against physical constraints. In addition, the system serves as rudimentary expert system which can monitor spatial, thermal, fluid, electrical and data compatibilities among different system components. Ultimately, this database will act as the common user interface for entering every piece of information about the system as the system evolves into its final design.

The Foxpro 1.02 database package has been selected to serve as the basic platform for building the desired database system. This package can permit users from any background to

operate the system with ease. The package provides a customized data entry and organization system which is specific to the needs of the overall project. The primary objectives of this design was to build a standard, portable database structure, to provide a uniform graphical user interface and to facilitate data access from other software components in the project.

3.4 Component description

The database system stores the general information pertaining to every component as well as the specific information of every model of such components. Each component in the database system is described in two ways: First, general characteristics of a component, (e.g. the quantity of the fluid flow through the component, the absolute pressure of the fluid flow, the absolute temperature of the fluid or the location of the connections in the component), second, characteristics specific to a particular model of a component (e.g. the mass, the cost and the dimensions). These component characteristics are also termed as *library component characteristics* for the sequel.

All general characteristics of components are stored in a *component database file*. The component database file has fields to store every general characteristic of all components. Each record in this database file contains the general information of a particular component. All library component characteristics are stored in database files with each record bearing the specific name of the component model. These database files contain different fields for storing different library component characteristics. For example, all library component characteristics of all the different filters are stored under a separate database file. Each record in this database file contains the specific characteristic information of a particular model of filter, such as `Filter 1, Filter 2, Filter 3,` etc.

3.5 Connection description

Impedance matching is the consideration that is given in the initial design of components to assure that the impedance properties of a particular component are compatible with the associated transmission system into which it is to be incorporated. A component that is properly matched will produce efficient transmission of power whereas a poorly matched component results in an unnecessary loss of energy. Therefore impedance matching is required between two connected components. For more information on impedance matching please refer Robert 1976[12].

Three categories of connections are defined for the plant; mechanical, electrical and fluid connections. Every connection in the plant is specified by some unique properties. A mechanical connection is specified by its cross section, mode of connection and the material of the connection. For example, a mechanical connection can be an universal coupling made of steel with a cross sectional area of 24 square units. An electrical connection is specified by the thickness of its diameter, type of insulation and the its material. For example, an electrical connection can be a copper wire having a diameter of 15 units with a `Type B` insulation. A fluid connection is specified by the thickness of the connection, size of the connection and the material of the connection. For example, a fluid connection can be a stainless steel channel with a cross-sectional area of 32 square units and a thickness of 3 units. All mechanical connections in the database system are stored in a separate database file. Similarly all electrical and fluid connections in the database system are stored in separate files. Properties associated with a particular connection are stored under various fields in these database files. For example, the database file for mechanical connections has fields for storing information like the description of the mechanical connection, cross section, mode of connection and the material of the connection. Every record in these files contains information pertaining to a particular connection.

For example, in an electrical connection database, all information pertaining to a particular electrical connection, say `Electrical 1`, is stored in a record. The information pertaining to a connection is accessed by seeking the name of the connection in the database file which contains that particular connection. For example, if the information pertaining to a particular connection, say `Mechanical 1` is required, it is obtained by seeking the record in the mechanical connection database, which has `Mechanical 1` as its connection description.

3.6 Data entry, storage and modification

Foxpro has capabilities to create windows, pop-up screens and menu systems which can be operated by both the keyboard and the mouse. These utilities help the user to access the relevant data. Figure 3.2 shows the main menu. It has options to edit, add and delete data present in the database. The main menu also has options which allow the user to build his own oxygen manufacturing plant, options to perform specific optimization operations (by using the

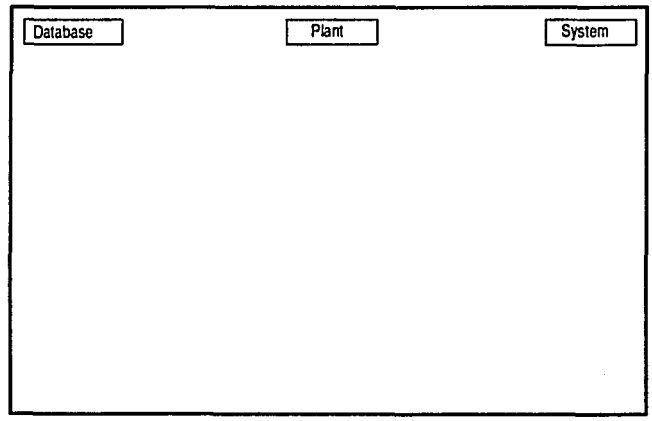

Figure 3.2 Main menu

Optimize option in the Plant menu) or execute system commands. The System option of the main menu allows the user either to return to Foxpro or DOS.

3.6.1 Add operations

Selecting the Add option present in the Database menu, brings up a add pop-up menu as shown in Figure 3.3. When the user selects the, Components to the database option from the Add pop-up menu, a pop-up window appears which prompts the user for the name of the component. After the user enters the name of the component, an algorithm searches for the existence of the named component in the component database file. If a component with the entered name exists in the component database file then the algorithm prompts the user whether he/she wants to add the component into the library of such components. Please refer Figure 3.4.

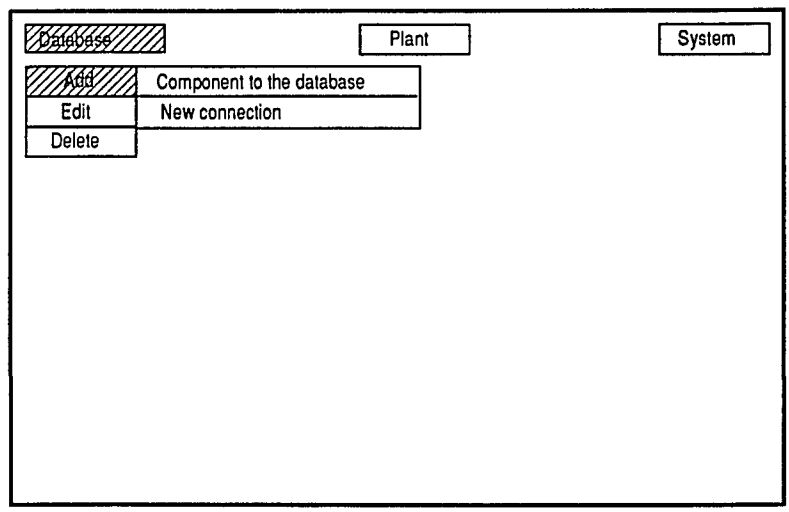

Figure 3.3 Main menu with add pop-up

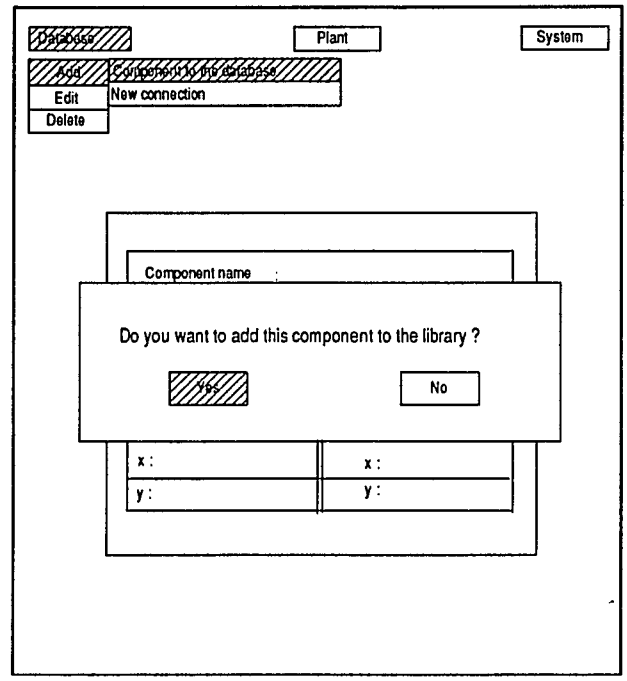

Figure 3.4 A window prompting the user to add a component to the library

If the entered component name does not match the name of other components in the component database file, then the user is allowed to proceed with the entry of general properties of that component. A window requesting the entry of general component properties appears as shown in Figure 3.5 The data is then stored in the component database file after the user's confirmation. When the user enters the component name which already exists in the component database, a pop-up window appears which prompts the user for entry of data pertaining to a model of the selected component.

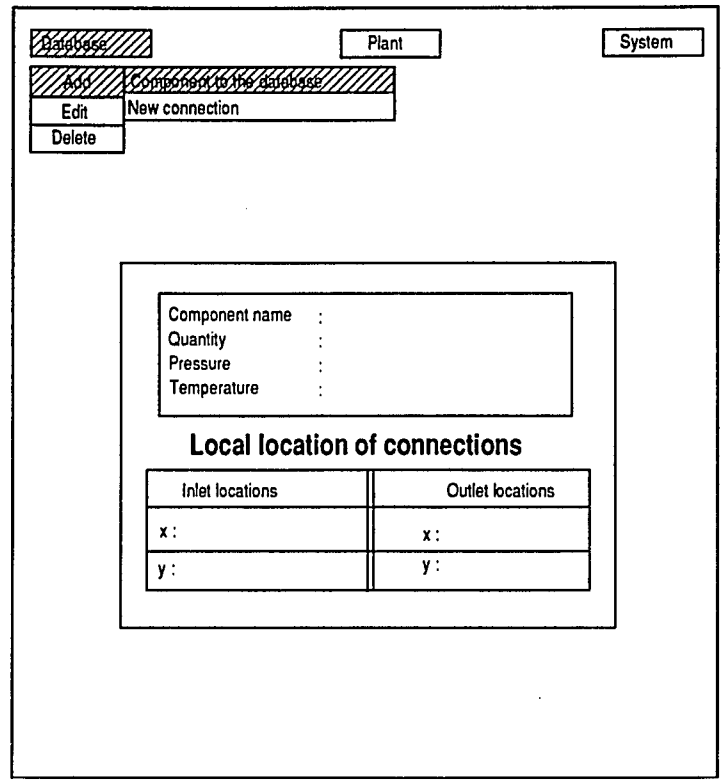

Figure 3.5 A window requesting the entry of general component properties

The data entered is then stored after confirmation. The sequence of these operations is shown in Figure 3.6. If the user selects the option called `New connection` from the add pop-up, then the general connections in the database.

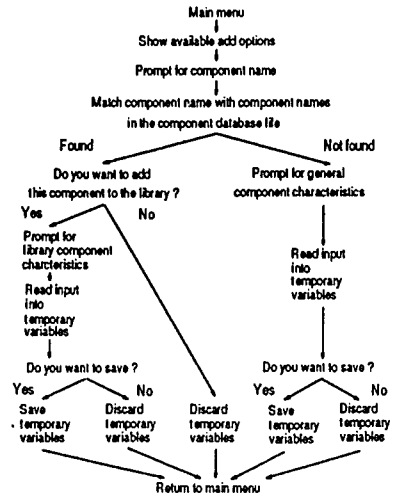

Figure 3.6 Sequence of operations to add a component to the database

system pops up. They are `Mechanical`, `Electrical` and `Fluid` connections, as previously explained. When the user picks up a particular connection say `Mechanical connection`, then a window requesting for the input of the properties of mechanical connections pops up, as shown in Figure 3.7.

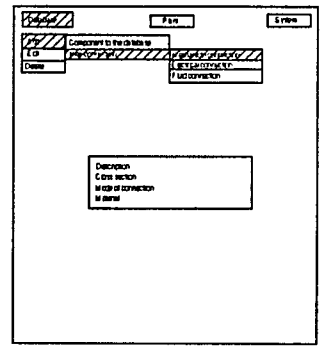

Figure 3.7 A window requesting the properties of a mechanical connection

Similarly windows for electrical and fluid connections appear, if the user picks up the respective option as shown in Figure 3.8 and Figure 3.9.

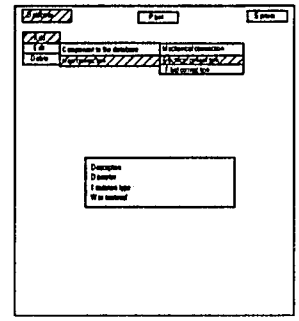

Figure 3.8 A window requesting the properties of an electrical connection

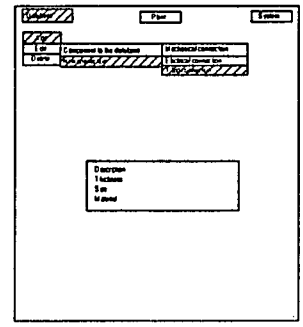

Figure 3.9 A window requesting the properties of a fluid connection

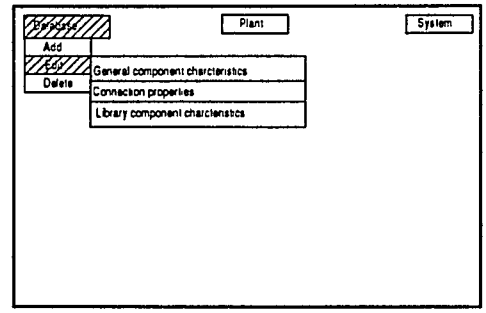

Figure 3.10 Main menu with edit pop-up

3.6.2 Edit operations

Selecting the button called `Edit` from the `Database` menu, brings up a edit pop-up menu as shown in Figure 3.10.

When the user selects the `General component characteristics` option from the edit pop-up menu, a pop-up consisting of the list of components available in the database appears. If the user now selects a particular component in the pop-up, the data relevant to the chosen component is displayed as shown in Figure 3.11. The user is now allowed to make appropriate changes in the data relevant to the chosen component. The modified data is stored into temporary variables. After the completion of the data modification process, a pop-up window appears requesting user's confirmation, as shown in Figure 3.12. If the user selects the `Yes option`,

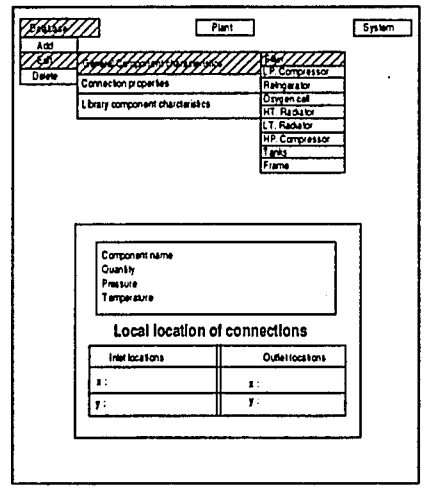

Figure 3.11 Main menu with component edit pop-up

the temporary variables are converted into permanent variables, and thus the modified data is stored in the database file for permanent use.

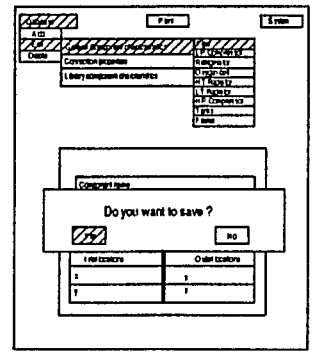

Figure 3.12 Component edit pop-up with window requesting user's confirmation

If the user selects the `No` option, the modified data is not copied into the database file. The sequence of these operations are shown in Figure 3.13. If the user now selects the second option in the edit pop-up (please refer Figure 3.14) i.e. `Connection properties`, a pop-up appears with the list of general connections in the database system. They are `Mechanical`

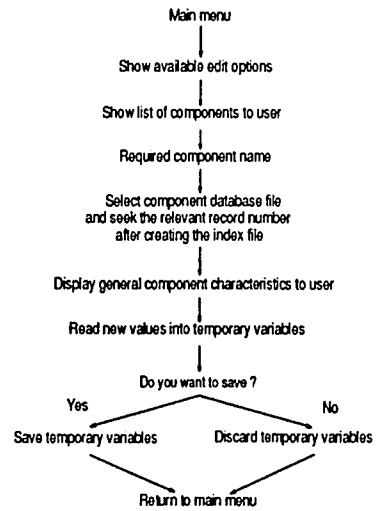

Figure 3.13 Sequence of operations to edit general component data

connection, Electrical connection and Fluid connection. When the user picks up a particular option, say a Mechanical connection, a list of mechanical connections present in the database system pops up. On the selection of a particular mechanical connection, say Mechanical 1, the properties associated with this mechanical connection pops up, as shown in Figure 3.14. As explained above, the modified data is copied into the database file, if the user confirms the request for saving the modifications. In a similar way, the list of electrical connections present in the database file is displayed, if the user picks up the Electrical connection option. The properties associated with the chosen electrical connection, say Electrical 2, pops up for modification. The changes are then recorded or discarded on the basis of the user's action. In a similar manner, all fluid connections can also be modified and stored. The sequence of these operations is shown in Figure 3.15.

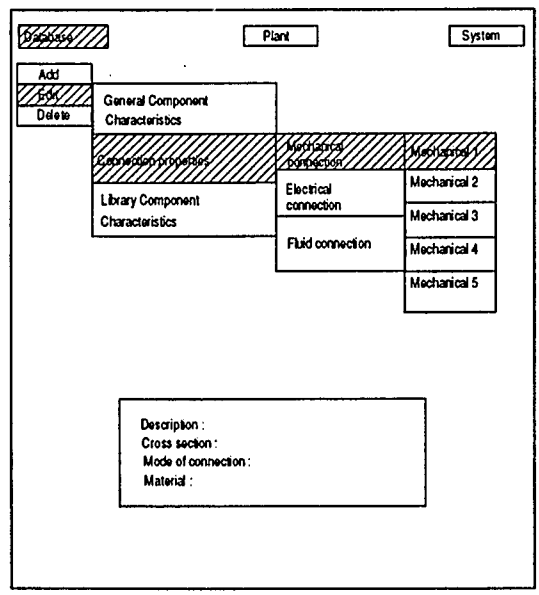

Figure 3.14 Main menu with connection edit pop-up

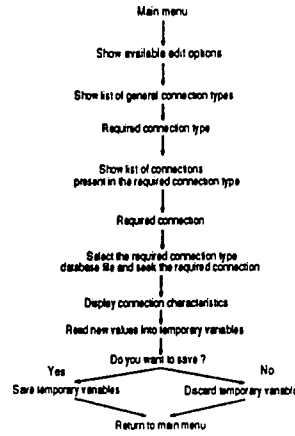

Figure 3.15 Sequence of operations to edit connection properties

When the user picks the third option of the edit pop-up (please refer Figure 3.16) i.e. `Library component characteristics`, the list of components available in the database file pops up. When the user selects a particular component, say `Filter 1`, the list of filters present in the

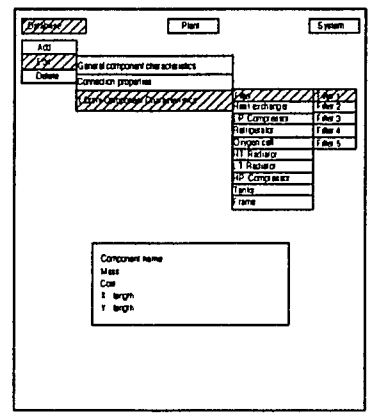

Figure 3.16 Main menu with library edit pop-up

database file for filters, pops up. When a particular filter is chosen say `Filter 1`, the properties associated with this filter pops up for modification (please refer Figure 3.16). The modifications are either saved or discarded as previously explained. The properties associated with any particular model of any component are thus modified in the same way. The sequence of these operations is shown in Figure 3.17.

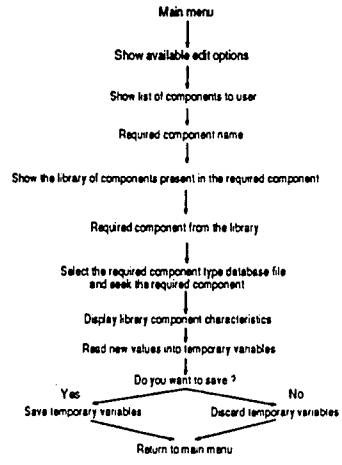

Figure 3.17 Sequence of operations to edit library component data

3.7 Delete operations

The `Delete` option of the `Database` menu allows the user to remove any component or remove any kind of connection or remove any model of any component from the database. The flow of control in performing these operations is the same as that of edit operations.

3.7.1 Removing a component from the database

When the option, `A component from the plant` is chosen from the delete menu, a list of all the components present in the component database file pops up for deletion. Once a

component is selected, the particular record which contains the selected component is located by the `Search` command. `Delete` and `Pack` database commands are then used to remove all the data relevant to the chosen component in the component database file.

3.7.2 Removing a connection from the database

Any connection can be removed from the database by selecting the option, `A new connection` from the delete menu. Once this option is activated, all the general connections in the database pops up, for selection. When a particular connection is chosen, all models of the chosen connection present in the database for such connections, pops up for deletion. For example, when a `Fluid connection` is selected, a list of all the models of fluid connections present in the fluid database file pops up for deletion. When a particular fluid connection is chosen, the `Search` command is selected to search the chosen connection in the fluid database file. Once this connection is chosen, `Delete` and `Pack` commands are used to remove all the information pertaining to the chosen model. Any model of any kind of connection can thus be removed from the database.

3.7.3 Removing a model of any component from the database

Any model of any component can be removed from the database by selecting the option called `Remove a component from the library`. Once this option is activated, all components present in the component database, pops up for selection. Once a component is selected, a list of all the models of the selected component pops up for deletion. The selected model is then located in the database file (if it contains the selected model) by the `Search` command.

3.8 Build operations

The database system has capabilities which allow the user to build a test-bed Mars oxygen manufacturing plant using the components and connections chosen by the user. Menu systems help the user to pick any component model and place it at any desired location in the plant. When the user picks up the `Plant` option of the main menu, a list of options appear as shown in Figure 3.18.

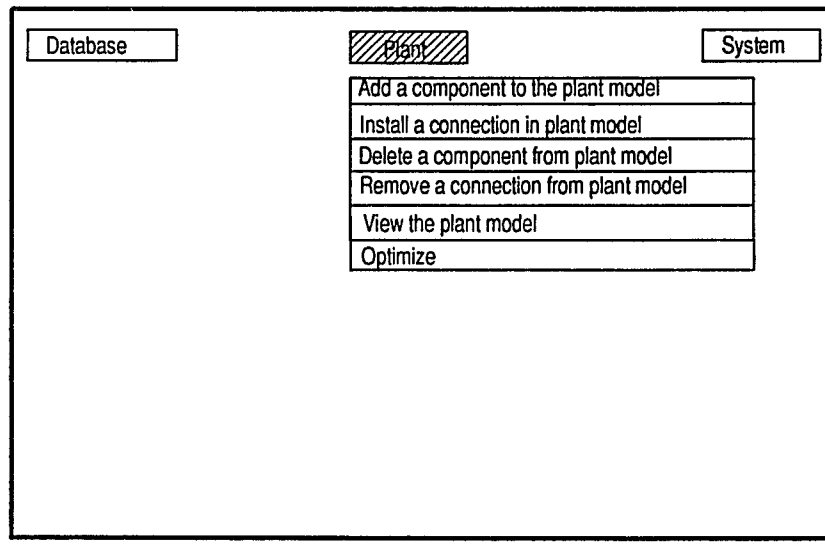

Figure 3.18 Main menu with plant pop-up

3.8.1 Adding a component to the plant model

If the `Add a component to the plant model` option is selected, the list of components present in the component database pops up. On the selection of any one of the components in the database, the corresponding list of models of the selected component appears, as shown in the Figure 3.19 . When a particular model is selected, its dimensions are transferred to temporary variables.

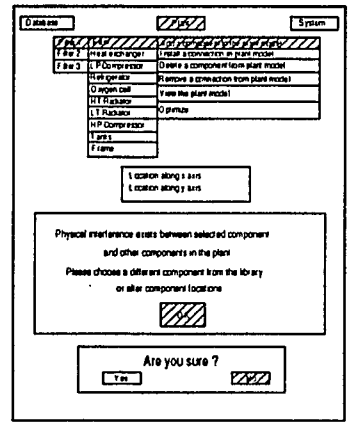

Figure 3.19 Adding a component to the plant model

A window then appears requesting the entry of *x,y* coordinates of the left-most corner of the selected model, in the plant model. These coordinates which have the plant layout as their reference are termed as *global* coordinates. Once the *x,y* coordinates are entered, the exact location of the component in the plant is computed. The suggested location is then processed by a subroutine which checks for any *physical interference* of the selected component with other components in the plant model. When a component is placed over one or more components, in

other words if the area occupied by a component is occupied by one or more components (whether the whole area or a part is occupied), then such a state is termed as physical interference. In case of physical interference the user is warned as shown in the Figure 3.19. The *x,y* coordinates of the connections which are local to the selected component are termed as *local coordinates*. The *local* coordinates of connections in the plant model are read from the component database. The *global* coordinates of these connections in the plant model are then calculated, since the *global* coordinates of the components are known. The values of these *global* connection locations are then stored in temporary variables.

The user is asked to confirm the entry as shown in the Figure 3.19. If the user confirms the request, the values of the temporary variables are transferred to a separate database file, called the **Build** file, which stores all the information relevant to the test-bed oxygen manufacturing plant. Part of the data is transferred to the record containing the chosen model in a database file which contains all the different models of the selected component. The information in these database files are utilized by the different optimization routines.

3.8.2 Installing a connection in the plant model

Any kind of connection can be made between any number of components by selecting the second option of the `Plant` menu (i.e. `Install a connection in plant model`). When this option is selected a sub menu pops up, listing the general connections available in the database system. They are the mechanical connection, electrical connection and the fluid connection. When any one of these general connections is chosen, a list containing all the models of the chosen connection appears, as shown in the Figure 3.20.

Once a particular model of any one of the general connections is chosen, the required specific properties of the chosen model are transferred to temporary variables. Sub menus like

`From` and `To` then appear to aid the builder in connecting the appropriate components in the plant model. A confirmation is then requested from the user, as shown in the Figure 3.20.

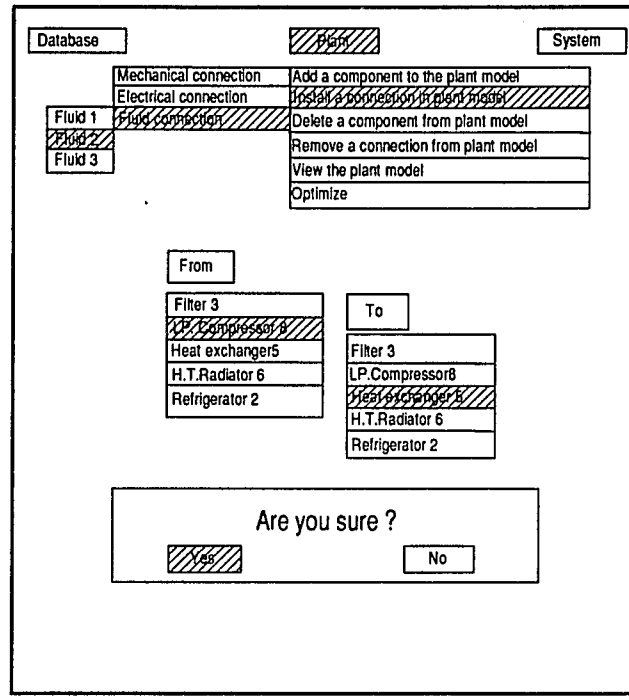

Figure 3.20 Installing a connection in the plant model

For example, when a `Fluid connection` is selected, all models of the selected fluid connection pops up, as shown in the Figure 3.20. Specific properties of the chosen fluid connection (viz. the size of the fluid connection, absolute surface temperature of the chosen fluid connection and the material of the connection) are transferred to temporary variables. All components present in the **`Build`** database file pops up, under the `From` sub heading. Once a component is chosen, the `Search` command is used to locate the record corresponding to the

component. The required global connections of the searched component are then transferred to temporary variables. The same procedure is repeated, when the `To` field is chosen.

On confirmation by the user, all temporary variables are transferred to the connection database file. Part of the data (viz. the specific properties of the chosen fluid connection) is transferred to the **`Build`** database file. Thus the database files are updated and empty fields in the database files are filled.

3.8.3 Removing a component from the plant model

The remove option allows any component to be removed from the plant model. When this option is activated, a menu showing all components present in the **`Build`** database file pops up. When a particular component is chosen, the `Search` command is used to locate the record corresponding to the component. Once the component is searched, the `Delete` and `Pack`

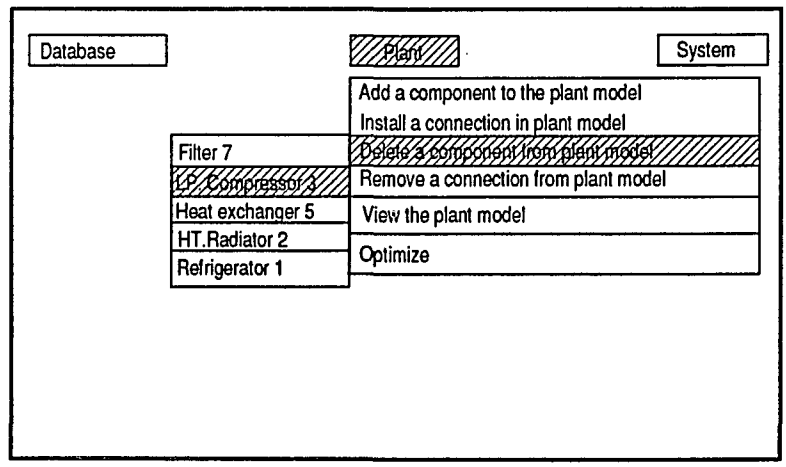

Figure 3.21 An option to delete a component from plant model

database commands are used to remove the record containing all the data relevant to the component. Thus any component can be removed from the plant model as shown in Figure 3.21.

3.8.4 Removing a connection from the plant model

The same procedure is used to remove any connection in the plant model. In this case, all connections present in the plant model pops up, when this option is activated. The chosen connection is searched and deleted from the connection database file. This option allows the deletion of any kind of connection from the plant model (please refer Figure 3.22).

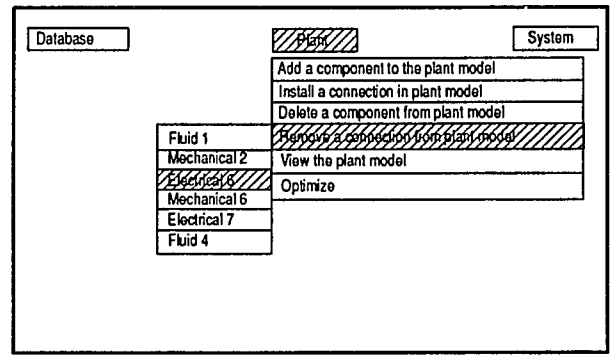

Figure 3.22 The option to delete a connection from the plant model

3.8.5 Building the plant model

Once the specific models of components and connections are selected, a sub routine transfers the dimensions of the components and the global connections in the plant model to text files. The dimensions of the components in the plant model are read from the `Build` database file. The global connections in the plant model are read from the `Connection` database file.

The components in the plant are modelled as rectangular blocks which are assumed to represent the actual component configuration. Figure 3.23 shows this kind of modelling. A particular type of connection is represented in Figure 3.23. Different types of connections can represented in different ways.

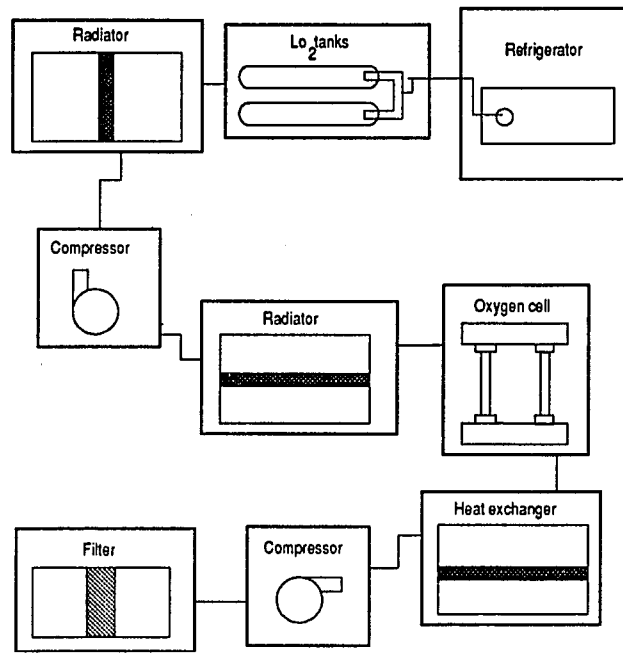

Figure 3.23 Rectangular modelling of the plant

The exported data in text files are read by a graphics routine written in C. The data imported by the graphics routine is utilized by C graphics commands to draw the plant model. Any user defined plant configuration could be obtained. An example of such a configuration is shown in Figure 3.24.

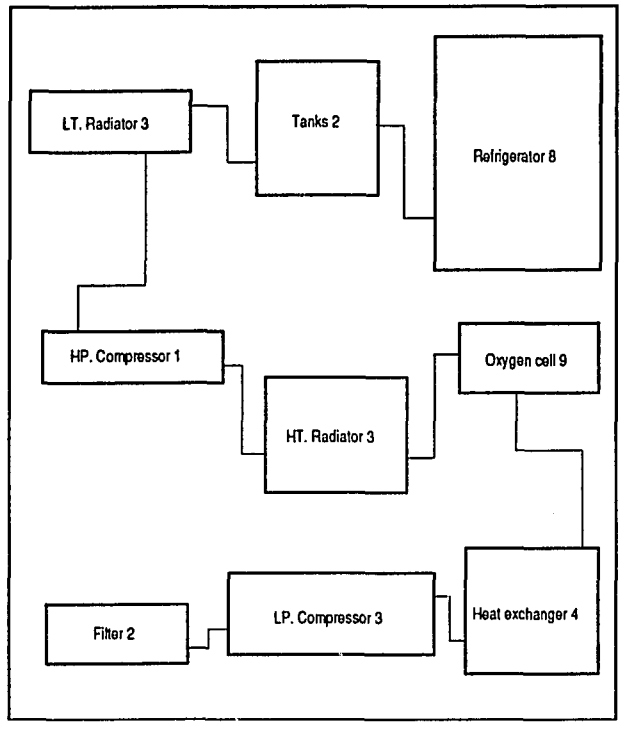

Figure 3.24 Initial plan of the plant

CHAPTER 4

IMPLEMENTATION OF THE OPTIMIZATION PROCESS AND ITS INTEGRATION WITH THE DATABASE SYSTEM

4.1 Introduction to optimization

The objective of general engineering design procedures is to find an acceptable solution which satisfies the functional requirements and physical constraints of a problem. Since in general, there will be more than one acceptable solution, the purpose of optimization is to choose the best one of the many acceptable solutions. Thus a criterion has to be chosen for comparing the different alternate acceptable solutions and for selecting the best one. The criterion with respect to which the design is optimized, when expressed as a function of certain variables is known as *criterion* or *merit* or *objective function*.

The ultimate goal of optimization is to either minimize the effort required or maximize the desired result. Since the effort required or the benefit desired in any practical situation can be expressed as a function of certain decision variables, optimization can be defined as the process of finding the conditions that give the maximum or minimum value of a function. Without loss of generality, optimization can be taken to mean minimization since the maximum of a function can be found by seeking the minimum of the negative of the same function. There is no single method available for solving all optimization problems efficiently. Hence a number of optimization methods have been developed for solving different types of optimization problems.

The methods adopted to find the optimal solution of an objective function are also known as mathematical programming techniques. The mathematical programming techniques are useful in finding the minimum of a function of several variables under a prescribed set of constraints.

4.2 Statement of an optimization problem

An optimization or a mathematical programming problem can be stated as follows:

$$\text{Find} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \quad \text{which minimizes } f(\boldsymbol{X}) \tag{4.1}$$

subject to the constraints

$$g_j(\mathbf{X}) \leq 0, \quad j = 1,2,.....,m$$

and

$$l_j(\mathbf{X}) = 0, \quad j = 1,2,.....,p$$

where $\boldsymbol{X}$ is an n-dimensional vector called the design vector, $f(\boldsymbol{X})$ is called the objective function and $g_j(\boldsymbol{X})$ and $l_j(\boldsymbol{X})$ are, respectively, the inequality and the equality constraints, n denotes the number of variables, m denotes the number of inequality constraints and p denotes the number of equality constraints. The problem stated in Equation 4.1 is called a constrained optimization problem. Some optimization do not involve any constraints and can be stated as:

$$\text{Find} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \quad \text{which minimizes } f(\boldsymbol{X}) \tag{4.2}$$

Such problems are called unconstrained optimization problems.

4.3 Classification of optimization problems

Optimization problems can be classified in several ways. Some of the important classifications are as follows:

4.3.1 Classification based on the existence of constraints

Any optimization problem can be classified as a *constrained* or an *unconstrained* one depending upon whether the constraints exist or not in the problem.

4.3.2 Classification based on the nature of equations involved

Another important classification of optimization problems is based on the nature of expressions for the objective function and the constraints. According to this classification, optimization problems can be classified as *linear, nonlinear, geometric* and *quadratic* programming problems.

If any of the functions among the objective and constraint functions in an optimization problem is nonlinear, the problem is called a Nonlinear programming problem. A function h($\boldsymbol{X}$) is called a *posynomial* if h can be expressed as the sum of power terms each of the form:

$$c_i \; x_1^{a_{i1}} \; x_2^{a_{i2}} \ldots\ldots x_n^{a_{in}}$$

where c_i and a_i are constants with $c_i > 0$ and $x_j > 0$. Thus a posynomial can be expressed as:

$$h(\mathbf{X}) \;=\; c_1 x_1^{a_{11}} x_2^{a_{12}} \ldots\ldots x_n^{a_{1n}} + \ldots\ldots + c_n x_1^{a_{N1}} x_2^{a_{N2}} \ldots\ldots x_n^{a_{Nn}} \qquad (4.3)$$

A geometric programming problem is one in which the objective function and constraints are expressed as posynomials in $\boldsymbol{X}$. A quadratic programming problem is a nonlinear programming problem with a quadratic objective function and linear constraints. A quadratic problem can be stated as:

Find $\boldsymbol{X}$ which minimizes,

$$F(\mathbf{X}) = c + \sum_{i=1}^{n} q_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij} x_i x_j \qquad (4.4)$$

subject to

$$\sum_{i=1}^{n} a_{ij} x_i = b_j, \quad j = 1,2,.....m$$

$$x_i \geq 0, \quad i = 1,2,.....n$$

where c, q_i, Q_{ij}, a_{ij} and b_j are constants. If the objective function and all the constraints are linear functions of the design variables, the mathematical programming problem is called a linear programming problem. A standard form of a linear programming problem is as follows:

Find $$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \qquad (4.5)$$

which minimizes

$$f(\mathbf{X}) = \sum_{i=1}^{n} c_i x_i$$

subject to the constraints

$$\sum_{k=1}^{n} a_{jk} x_k = b_j, \quad j = 1,2,...,m$$

and

$$x_i \geq 0, \quad i = 1,2,....n$$

where c_i, a_{jk} and b_j are constants.

4.3.3 Classification based on the separability of the functions

Optimization problems can be classified as separable and nonseparable programming problems based on the separability of the objective and constraint functions. A function f($\boldsymbol{X}$) is

separable if it can be expressed as the sum of *n* single variable functions $f_1(X)$, $f_2(X)$,....,$f_n(X)$, that is,

$$f(\mathbf{X}) = \sum_{i=1}^{n} f_i(x_i)$$

A separable programming problem is one in which the objective function and the constraints are separable and can be expressed in the standard form as

Find X which minimizes

$$f(\mathbf{X}) = \sum_{i=1}^{n} f_i(x_i) \tag{4.6}$$

subject to

$$g_j(\mathbf{X}) = \sum_{i=1}^{n} g_{ji}(x_i) \leq b_j, \quad j = 1,2,.....,m$$

where b_j is a constant.

For more information on optimization and its classification please refer Rao (1984)[13].

4.4 Programming requirement

If the objective function and the constraints are fairly simple in terms of decision variables, analytical methods of optimization can be used to solve the problem. On the other hand, if the optimization problem involves the objective function which is too complicated to manipulate, it is not easy to solve it by using analytical methods. Thus numerical methods are used to solve such problems.

4.5 Numerical methods of nonlinear programming

A nonlinear programming problem can have an objective function with a single or a number of variables. An objective function with a number of variables is called as *multivariable* objective function. A multivariable objective function can be classified into two types: first, a

multivariable objective function without constraints, second, a multivariable objective function with equality and inequality constraints. There are number of algorithms developed to solve these kinds of nonlinear programming problems. Some of them are dealt as follows:

4.5.1 Line searching

One of the simple algorithms for finding the minimum of a function of one variable is the *bisection line search.* The idea of this method is to look for a value of x at which the derivative $df(x)$ is zero. Inorder to search the interval [a^o, b^o] for a minimum of the function, the midpoint of the interval is found ($x^o = \frac{1}{2}\left(a^o+b^o\right)$) and the derivative of the function is evaluated at this point.

The slope of a function's graph is negative to the left of a minimum and positive to the right. Thus if the derivative is positive then the minimum for a convex function would lie to the left of the considered point. In that case the right half of the original interval can be discarded to obtain a new problem whose interval size is half of the previous one. In the other case when the slope at the midpoint is negative, the minimum will be at the right half of the interval and so the left half is discarded. Repeating this process reduces the interval of uncertainty until the location of the minimum is known to be within any prescribed tolerance. The algorithm is shown in Figure 4.1.

The line search algorithm can converge to a local minima when the function has them instead of to a global minimum. The algorithm can fail because of round off errors or irregular behavior (such as nondifferentiability) of the function. A speedy convergence is obtained only at the price of decreased robustness. These are the draw backs in the line searching technique.

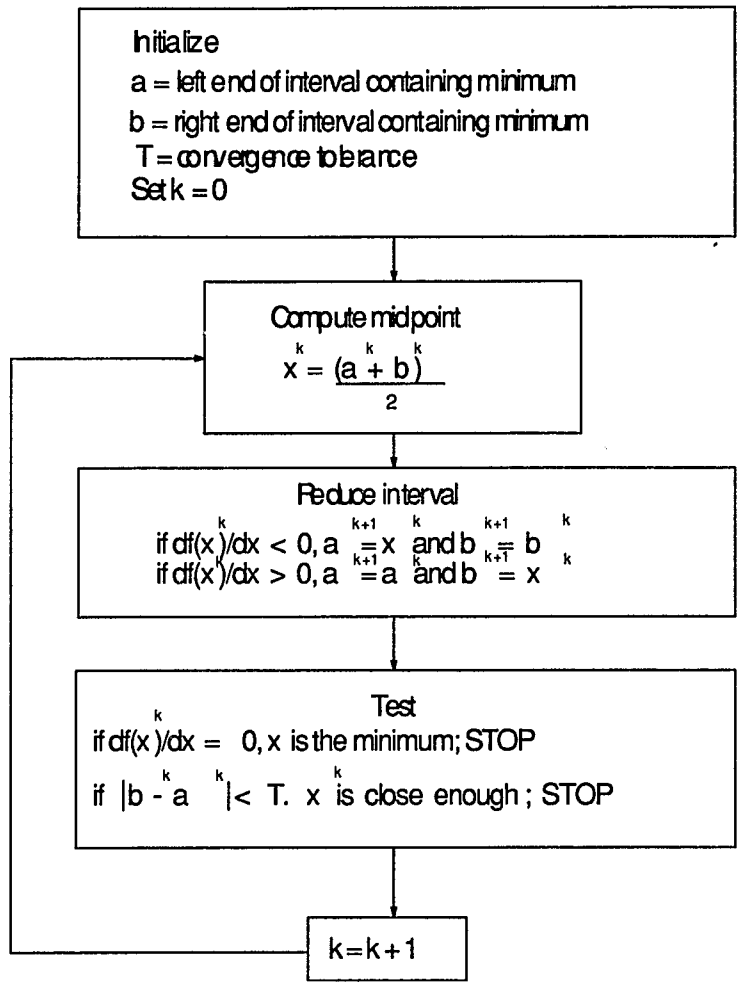

Figure 4.1 Bisection line search

4.5.2 The method of steepest descent

The numerical method for finding the constrained minimum of a continuous, non-linear objective function is based on the idea of going down hill on the graph of the function. The

gradient of a function always points in the direction of the fastest increase so the way to go downhill is to move in the direction opposite to the gradient vector.

The search direction is found by computing the negative of the gradient. For steepest descent in a constrained space with n variables,

$$0 \leq x_j \leq U_j$$
$$j = 1,2,....n$$

where U_j is a vector containing the maximum value of the variables. The search direction is given as

$$d^k = \begin{bmatrix} -\frac{\delta O_I}{\delta x_j} \\ 0 \end{bmatrix} \tag{4.7}$$

where d^k is the search direction and is equal to $-\frac{\delta O_I}{\delta x_j}$ if any one of the following conditions are satisfied.

$$0 < x_j < U_j$$

$$x_j = 0 \text{ and } \frac{\delta O_I}{\delta x_j} < 0$$

$$x_j = U_j \text{ and } \frac{\delta O_I}{\delta x_j} > 0$$

If these conditions are not satisfied then d^k is made zero, as indicated in Equation 4.7. The bisection line search is then performed in the direction d^k. The line search is performed on the line,

$$x^{k+1} = x^k + ad^k$$

where x is the line passing through the points x^k and having direction d^k. The scalar parameter a measures the distance on the line from the point. Having completed the line search in the direction d and obtained a^* it is possible to update the vector x^k such that the value of the objective function is a minimum.

$$x^{k+1} = x^k + a^* d^k$$

The process is now repeated by starting from this point, by calculating the search direction, conducting a line search in that direction, and so forth. The iterations continue until successive values of x^k are close enough together such that

$$\left\| x^{k+1} - x^k \right\| < T$$

where T is the tolerance permitted. The algorithm is summarized as shown in Figure 4.2.

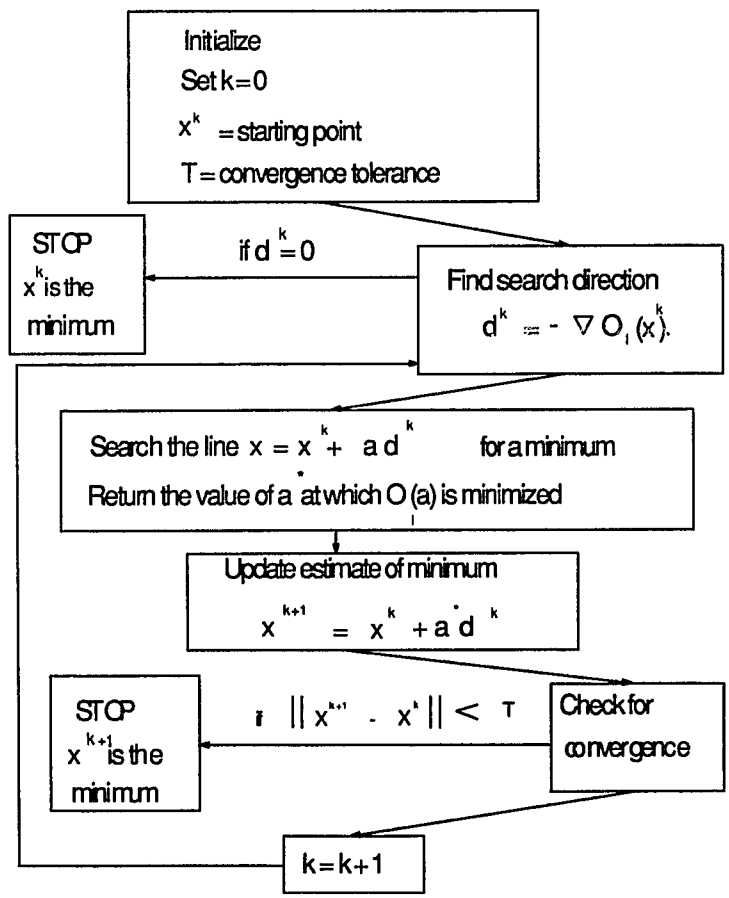

Figure 4.2 Steepest descent algorithm

The algorithm follows a long, narrow valley in the graph of the function, taking tiny steps and requiring many iterations to get near the optimal point . Thus phenomenon, known as *zigzagging*, causes slow convergence and is thus a serious drawback to the method of steepest descent.

4.5.3 The generalized reduced-gradient method (GRG)

This method is used to determine the minimal value of a nonlinear objective function with linear or nonlinear constraints. The minimization of a nonlinear objective function with linear constraints is a part of the method adopted to minimize a nonlinear objective function with nonlinear constraints. The minimal value of a nonlinear objective function with nonlinear constraints is found by linearizing the constraint equations. The idea of the generalized reduced gradient algorithm is to solve a sequence of subproblems, each of which uses a linear approximation of the constraints. The linearization of the constraint equations is performed by expanding each constraint function in a Taylor series and neglecting terms beyond the linear one.

In each iteration of the algorithm, the constraint linearization is recalculated at the point found from the previous iteration. Typically, eventhough the constraints are only approximated, the subproblems yield points that are progressively closer to the optimal point. As the optimal point is approached, the linearized constraints of the subproblems become progressively better approximations to the original nonlinear constraints in the neighborhood of the optimal point.

The first step in applying GRG is to pick a starting point X^0, at which to perform the first linearization. The approximation formulas are then utilized to linearize the constraint equations at the point X^0 and form the first approximate problem. The equality constraints of the approximate

problem is solved by expressing *m* variables in terms of other variables. These *m* variables are called as *basic* variables and the other variables are called as *nonbasic* variables.

The basic variables are substituted in the objective function to yield the reduced problem. The resulting objective function is an unconstrained one. The minimal value for the unconstrained objective function is determined by setting the partial derivatives and equating them to zero. The basic variable values are then computed. The nonbasic variables are then found since the basic variables are known. Thus the first iteration of the GRG algorithm produces a new point x^1 .

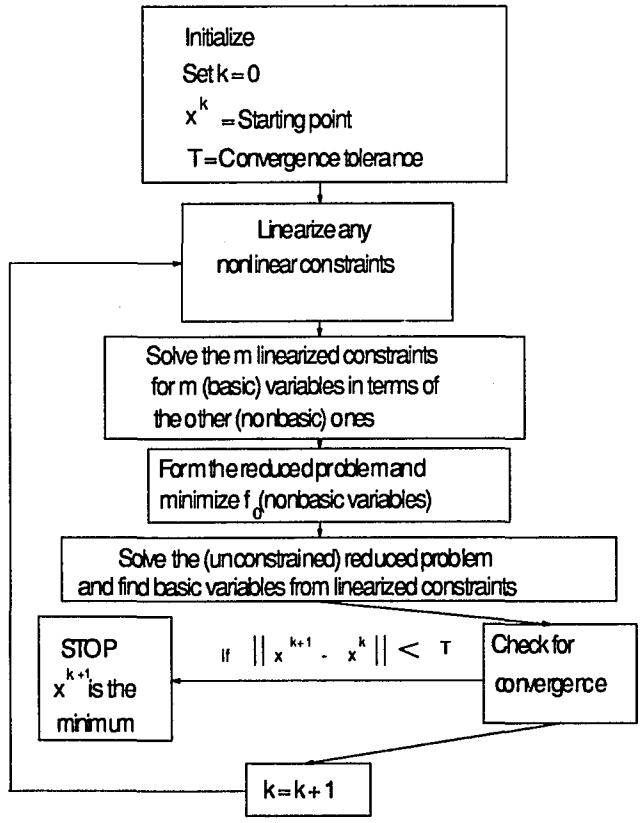

Figure 4.3 Generalized reduced-gradient algorithm

The solution process is continued by relinearizing the constraint functions at the new point, use the resulting system of linear equations to express m basic variables in terms of the other nonbasic variables, substitute into the objective function to obtain a new reduced problem, solve the reduced problem for x^2, and so forth. The flow chart explaining the GRG algorithm is shown in Figure 4.3.

The algorithm exhibits a varying rate of convergence. The GRG algorithm is quite complicated when compared with some other nonlinear minimization methods. The performance of the GRG algorithm is difficult to predict on theoretical grounds. These are the difficulties experienced with the GRG algorithm.

4.5.4 The ellipsoid algorithm

A simple algorithm for inequality-constrained nonlinear programming problem is the *ellipsoid algorithm*. The basic idea of the ellipsoid algorithm is to generate a sequence of successively smaller ellipsoids each containing the optimal point. An ellipse is initially constructed to contain the optimal point. Since the optimal point is not known, the ellipse is so constructed that it contains the entire feasible region S. Because the optimal point must be feasible, an ellipse that contains the entire feasible region is sure to contain the optimal point. The optimal point is localized within regions of diminishing volume as the iterations of the algorithm progress, and under the right conditions the ellipsoid centers approach the optimal point. Each ellipsoid (after the first one) is constructed by cutting the previous ellipsoid in half and then enclosing the half that is known to contain the optimal point .

There are many ways to cut the initial ellipsoid in half so as to have S (which contains the optimal point) contained entirely in one of the halves, but a particularly easy cut is suggested by inspection of an inequality constraint contour which is an violated constraint. The line drawn

tangent to the contour $f_i(x) = f_i(x^0)$ (where *i* is the index of a violated constraint) is called the *hyperplane*. The algorithm which illustrates this is shown in Figure 4.4.

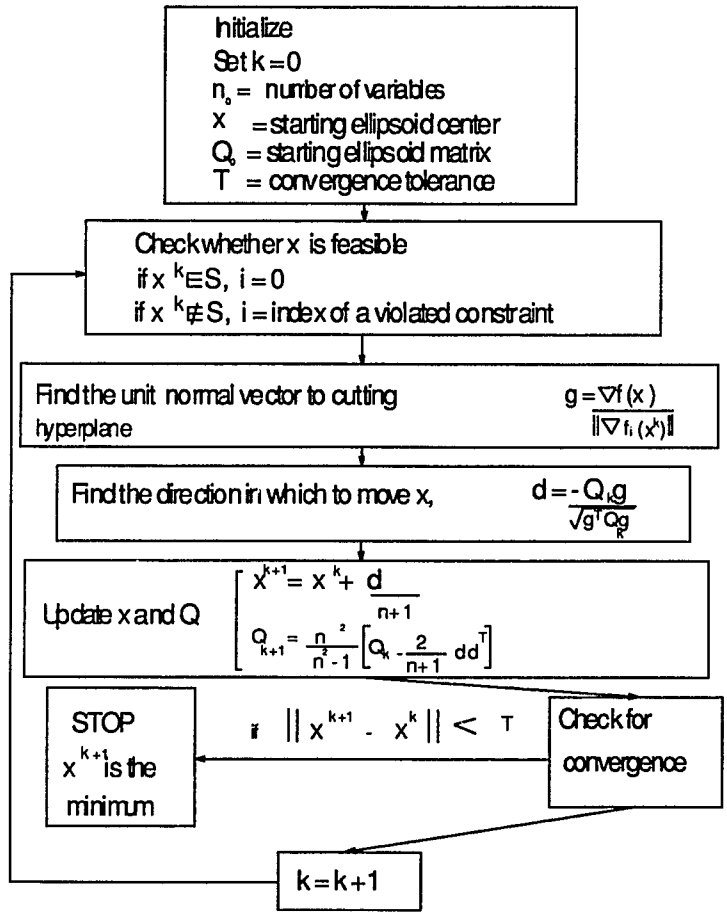

Figure 4.4 Ellipsoid algorithm

The ellipsoid algorithm sways in its results from one iteration to the other, particularly near the beginning of the solution process. A feasible point with a low objective value may be produced early in the iterations, only to be followed by many iterations in which the feasible points

all have higher objective values. This is a serious draw back in the ellipsoid algorithm. For more information on numerical methods of nonlinear programming please refer Ecker *et al* (1988)[14].

4.6 Various energy losses in the plant

Various designs for the oxygen manufacturing plant would contain components connected by various connections in a number of ways. In any of these configurations, there may be various thermal and pressure losses in the pipelines and radiation losses from and between various components of the plant. The energy losses that are considered in pipelines are the frictional pressure loss (when the fluid flows through the pipeline) and the energy loss due to the radiation of heat from the surfaces of pipelines. The energy losses due to the radiation of heat from and different components, due to the difference in their surface temperatures is termed as the *thermal proximity loss* for the sequel. Each one of these losses are dealt with separately in the following sections.

4.6.1 Frictional pressure loss in pipes

The pressure loss due to the presence of friction between a pipe and fluid flowing through it is termed as *frictional pressure loss*. The general formula for the frictional pressure loss of compressible fluids, expressed as the height of a fluid column of the same density is (in differential form):

$$dp = \frac{f\ v^2\ dl}{2\ g\ d} \tag{4.8}$$

where f is the friction factor, v is the velocity of the fluid flow, dl is the length of the pipe over which the pressure drop occurs, g is the gravitational acceleration and d is the diameter of the pipe. There are a number of simple empirical formulas that has been developed to find the friction factor f. The formula used most frequently is the Weymouth's formula for fluids with rough turbulent flow. According to Weymouth's formula:

$$f = 0.0319 \ (d^{-\frac{1}{3}}) \qquad (d \text{ in inches})$$

$$f = 0.0938 \ (d^{-\frac{1}{3}}) \qquad (d \text{ in mm})$$

Since the Reynolds number does not appear in the Weymouth formula, this formula is correct for rough turbulent flow only. A rough turbulent flow is assumed for this analysis. For more information on frictional pressure loss in pipes, please refer Vincent-Genod (1984)[15].

4.6.2 Radiative heat loss

The heat energy radiated from a body B_1 with a surface area A_1 to another body B_2 with a surface area A_2 is given as:

$$Q_{1-2} = \sigma \in_1 T_1^4 A_1 F_{1-2} \tag{4.9}$$

where Q_{1-2} is the radiated heat energy from B_1 to B_2 in watts, T_1 is the absolute surface temperature of B_1 in °K, F_{1-2} is the view factor from B_1 to B_2, $\in_1$ is the emissivity of B_1 and σ is the *Stefan-Boltzmann* constant which has the numerical value

$$\sigma = 5.670 \times 10^{-8} \text{ W/m}^2.\text{K}^4$$

Similarly the heat radiated from B_2 with surface area A_2 to B_1 with surface area A_1 is given as:

$$Q_{2-1} = \sigma \in_2 T_2^4 A_2 F_{2-1} \tag{4.10}$$

where Q_{2-1} is the radiated heat energy from B_2 to B_1 in watts, T_2 is the absolute surface temperature of B_2 in Kelvin, F_{2-1} is the view factor from B_2 to B_1, and $\in_2$ is the emissivity of B_2.

The net heat transfer from B_1 to B_2 is given as:

$$Q_{1\Rightarrow 2} = Q_{1-2} - Q_{2-1} = \sigma \in_1 T_1^4 A_1 F_{1-2} - \sigma \in_2 T_2^4 A_2 F_{2-1} \quad (4.11)$$

By the use of the reciprocity relation

$$A_1 F_{1-2} = A_2 F_{2-1} \quad (4.12)$$

Equation 4.11 can be written as:

$$Q_{1\Rightarrow 2} = \sigma A_1 F_{1-2} (\in_1 T_1^4 - \in_2 T_2^4) \quad (4.13)$$

The two bodies are assumed to be in the form of a rectangular box and their facing sides

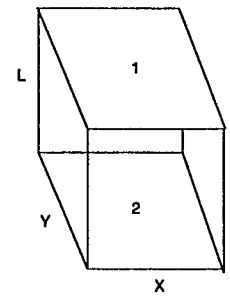

Figure 4.5 Aligned parallel rectangles

to be aligned with each other as shown in Figure 4.5. This assumption is done in order to avoid the large execution time which would result if actual component configurations are considered. The execution time is further decreased by considering the radiation losses only between connected components. The view factor for such a geometry is given as:

$$\overline{X} = \frac{X}{L}$$

$$\overline{Y} = \frac{Y}{L}$$

$$F_{1-2} = \frac{2}{\pi \bar{X}\bar{Y}}\left\{ \ln\left(\frac{\left(1+\bar{X}^2\right)\left(1+\bar{Y}^2\right)}{1+\bar{X}^2+\bar{Y}^2}\right)^{\frac{1}{2}} + \bar{X}\left(1+\bar{Y}^2\right)^{\frac{1}{2}} \arctan\left(\frac{\bar{X}}{\left(1+\bar{Y}^2\right)^{\frac{1}{2}}}\right) + \bar{Y}\left(1+\bar{X}^2\right)^{\frac{1}{2}} \arctan\left(\frac{\bar{Y}}{\left(1+\bar{X}^2\right)^{\frac{1}{2}}}\right) - \bar{X}\arctan\bar{X} - \bar{Y}\arctan\bar{Y} \right\} \quad (4.14)$$

4.6.3 Radiative heat loss in the plant

The radiative thermal loss from a pipe to the ambient surroundings is given by

$$Q_c = \sigma A_1 \in_1 (T_1^4 - T_a^4) \quad (4.15)$$

where Q_c is the heat radiated from the pipe to the surroundings, A_1 is the surface area of the

DESCRIPTION	Emmisivities at various temperatures (K)						
	100	200	300	400	600	800	1,000
Aluminium	0.02	0.03	0.04	0.05	0.06		
Chromium	0.05	0.07	0.1	0.12	0.14		
Copper			0.03	0.03	0.04	0.04	
Nickel					0.09	0.11	
Silver			0.02	0.02	0.03	0.05	
Stainless steel			0.17	0.17	0.19	0.23	
Tungsten							0.1

Table 4.1 Emissivity of various metallic solids

pipe, $\in_1$ is the emissivity of the connection material (given in Table 4.1, for various metals at various temperatures), T_1 is the surface temperature of the pipe, and T_a is the temperature of the surroundings.

The radiative thermal loss between any two components can be written as:

$$Q_{1\Rightarrow 2} = \sigma A_{c1} F_{1-2}(\in_1 T_{c1}^4 - \in_2 T_{c2}^4) \qquad (4.16)$$

where T_{c1} is the surface temperature of `Component 1`, t_{c2} is the surface temperature of `Component 2`, A_{c1} is the area of the rectangular facing side in `Component 1` (i.e. the side that faces `Component 2`) and F_{1-2} is the view factor as given in Equation 4.14. For more information on radiative thermal loss, please refer Incropera *et al* (1990) [16] and Siegel *et al* (1981) [17].

4.7 Objective functions related to the plant

When components are placed at discrete locations in the plant model for computing the minimum of an objective function whose value depends on the location of components, the objective function has variables which vary discretely. These locations are further subject to certain constraints such as the overall size of the plant and the requirement that there be no physical interference between components. This type of objective function is termed as the *discrete objective function* for the sequel. The mass and cost of components have discrete values. Thus the objective function relating the mass and cost of components has variables which have a discrete domain. Thus this function is a also discrete objective function. Every connection in the plant model has a number of models. Each model has its own size and material. The material of any connection determines the emmissivity of that connection. The size

and emmissivity of a fluid connection influences the radiation and frictional pressure losses in that connection. The mathematical expression for this relation is discussed in the following sections. Thus the objective function relating the size and emmissivity of all the connections in the plant has variables which are discrete.

When the components are placed at all possible locations within a constrained space in the plant, then the objective function which depends on the location of the components has variables which vary continuously. This type of objective function is termed as the *continuous objective function* for the sequel.

All objective functions are user-defined ones, because the user can assign suitable weighting factors to each one of the various parameters involved in the objective function. A weighting factor is any constant which is used to multiply a particular value of a parameter. Weighting factors determine the relative importance of a parameter over the other.

4.7.1 Objective function involving the various losses in the plant

The objective function which involves the losses discussed above is as follows:

$$O_l = \sum_{i=1}^{N_c} \left[w_{ri} Rc_i + w_{fi} Fc_i \right] + \sum_{j=1}^{(n-1)} \left[w_{tj} Tp_j \right] \qquad (4.17)$$

where O_l is the objective function involving the various losses in the plant, N_c is the number of connections in the plant, Rc_i is the radiative thermal loss in from connection i, w_{ri} is the weighing factor associated with the radiation losses in connection i, Fc_i is the frictional pressure loss in fluid connection i, w_{fi} is the weighing factor associated with the frictional pressure losses in connection i, n is the number of components in the system, w_{tj} is the weighing factor associated with the radiative heat loss due to thermal proximity from component j and Tp is the radiative heat loss

due to the thermal proximity from component *j* the component to which it is connected. Here the radiative heat loss due to thermal proximity is considered for two components which are connected with each other.

4.7.2 Objective function involving the mass and cost of components

Every component has a number of models as explained above. The mass and cost of each of these models are different. The objective function involving the mass and cost of components is as follows:

$$O_{mc} = \sum_{i=1}^{n} \left[w_{mi} m_i + w_{ci} c_i + w_{pi} p_i \right] \quad (4.18)$$

where n is the number of components in the plant, w_{mi} is the weighing factor for the mass of component i, m_i is the mass of component i, w_{ci} is the weighing factor for the cost of component i, c_i is the cost of the component i, w_{pi} is the weighting factor for the power requirement of component i and p_i is the power requirement of the component i.The last item (i.e. power term) could also be considered.

4.7.3 Objective function involving the material and size of connections

Every type of connection has a number of models. The material and size of every connection model are different. The emissivity property of every material is different. The emissivity and size of a connection affects the radiative heat loss from it. The size also affects

the frictional pressure in a connection. Thus the objective function involving the material and size of connections is

$$O_I = \sum_{i=1}^{N_c} \left[w_{ri} Rc_i + w_{fi} Fc_i \right] \tag{4.19}$$

where N_c is the number of connections in the plant. In this case the radiative heat loss in connections Rc_i and the frictional pressure loss in connections Fc_i do not depend on the position of components but depend on the model of connection (with a particular material and size).

4.8 Minimization of the objective functions

The minimization of these objective functions is performed in four stages. First, the optimal layout of the component models (which are initially picked by the user) in the plant is determined. This is achieved by computing the minimum value of the discrete objective function given in Equation 4.17. Second, a set of component models is chosen such that the total mass and cost of all component models in the plant, is a minimum. This is performed by calculating the minimum value of the discrete objective function given in Equation 4.18. Third a set of connection models from a given set of such connection models is chosen, such that the radiation and frictional pressure loss in all connections in the plant is a minimum. This is performed by calculating the minimum value of the discrete objective function given in Equation 4.19. Lastly, the fine-tuning of the locations of the component models in the plant is performed so that the value of the objective function (which depends on location) is further reduced. This is achieved by minimizing the objective function given in Equation 4.17. In this case, the variables of Equation 4.17 takes continuous values. A flow chart which represents these stages is shown in Figure 4.6. Each of these stages is dealt in greater detail as follows:

4.9 Determination of the optimal layout of component models

After the user builds the test-bed oxygen manufacturing plant, all components are placed at all possible locations and the value of the objective function is determined for each configuration. The minimum value of the objective function is then computed and thus the configuration with the minimum of losses is found. However not all possible locations can be searched continuously because the number of iterations would be very large.

As a result, set of discrete component locations in the plant for *n* components is found by dividing the plant into *n* approximately equal regions.

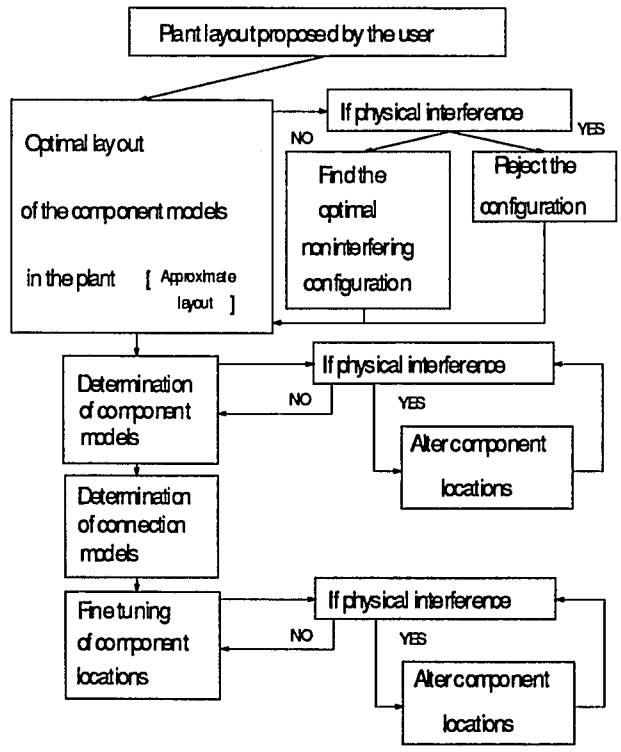

Figure 4.6 Stages in the optimization process

4.9.1 Division of plant into regions

A novel method has been adopted to divide the plant into n approximately equal regions, where n is the number of components in the plant model. When n is divided by a number a (a being either $\lfloor \sqrt{n} \rfloor$ or $\lceil \sqrt{n} \rceil$) and if z is the greatest integer in that division then z can be written as $z = \lfloor \frac{n}{a} \rfloor$. The remainder is written as MOD(n,a). The number n can be written as:

$$\begin{aligned} n &= az + MOD(n,a) \\ &= az - [MOD(n,a)](z) + [MOD(n,a)](z) + MOD(n,a) \\ &= [a - MOD(n,a)] \lfloor \tfrac{n}{a} \rfloor + MOD(n,a) \lceil \tfrac{n}{a} \rceil \end{aligned}$$

Therefore any number n can be expanded as

$$n = [\lfloor \sqrt{n} \rfloor - MOD(n, \lfloor \sqrt{n} \rfloor)] \left\lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \right\rfloor + \left[MOD(n, \lfloor \sqrt{n} \rfloor) \left\lceil \frac{n}{\lfloor \sqrt{n} \rfloor} \right\rceil \right] \quad (4.20)$$

where, n is the number of components in the plant model, $\left\lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \right\rfloor$ is the nearest integer that is less than or equal to the quotient of $\frac{n}{\lfloor \sqrt{n} \rfloor}$, $\left\lceil \frac{n}{\lfloor \sqrt{n} \rfloor} \right\rceil$ is the nearest integer that is greater than or equal to the quotient of $\frac{n}{\lfloor \sqrt{n} \rfloor}$, MOD($n$, $\lfloor \sqrt{n} \rfloor$) is the modulus operator acting on n and $\lfloor \sqrt{n} \rfloor$ which is the remainder obtained upon dividing n and $\lfloor \sqrt{n} \rfloor$. The number n can also be expanded as,

$$n = [\lceil \sqrt{n} \rceil - MOD(n, \lceil \sqrt{n} \rceil)] \left\lfloor \frac{n}{\lceil \sqrt{n} \rceil} \right\rfloor + \left[MOD(n, \lceil \sqrt{n} \rceil) \left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil \right] \quad (4.21)$$

From Equation 4.20 and Equation 4.21 it can be found that the plant can divided into [a - MOD (n,a)] rows with $\lfloor \frac{n}{a} \rfloor$ elements each and MOD(n,a) rows with $\lceil \frac{n}{a} \rceil$ elements each. Here a is either $\lceil \sqrt{n} \rceil$ or $\lfloor \sqrt{n} \rfloor$. The mechanism of this division is illustrated in Figure 4.7 for a 5-component system.

The value of a determines the number of rows and number of elements in the divided plant and also controls the spacing of the regions in the plant. The values of a are chosen by an indexing algorithm which switches the values of a between $\lceil \sqrt{n} \rceil$ and $\lfloor \sqrt{n} \rfloor$, depending on the value of n. If a is an integer then the plant is divided into a rows with a elements each. In this way, the plant is divided into n approximately equal regions. The flow chart which explains the indexing algorithm is shown in Figure 4.8.

4.9.2 Locating components in plant

Once the plant is divided into regions, all components in the plant model are placed in all possible regions, and the discrete objective function (which depends on the location of components in the plant) is computed for each possible set of component locations. The minimum value of this objective function (as given by Equation 4.17) can be computed by a brute search method which searches through the domain of all the discrete values taken by the variables present in the objective function. This would result in a full brute force search and would invoke *n*! iterations. This is because no component can be placed at a region which has already been occupied by another component (in order to avoid physical interference).

To avoid this problem, certain heuristics are introduced, so that only the connected components are placed in the adjacent regions of the considered component. This approach reduces the number of iterations. When the components are placed in the divided regions of the plant, a physical interference check is performed in parallel, so that interfering component configurations are rejected. Only the non-interfering component configurations are considered for minimization (i.e. the minimum value of the objective function for different non-interfering component configurations is determined). Numerical analysis of minimization and further discussions are dealt with in the next chapter.

Once the minimum value of the objective function is determined, the resulting component and connection locations are transferred by the Foxpro subroutine to text files and to a generic database file called `Optim` which stores the updated values. The text files are accessed by a graphics routine written in C, which reads these text files and draws the plant layout using C-graphics commands.

4.10 Determination of the optimal set of component models

The mass and cost of all component models are stored in database files. This information is transferred to text files which are accessed by a C program. The C program reads the information from the text files. After retrieving the information, the C algorithm searches

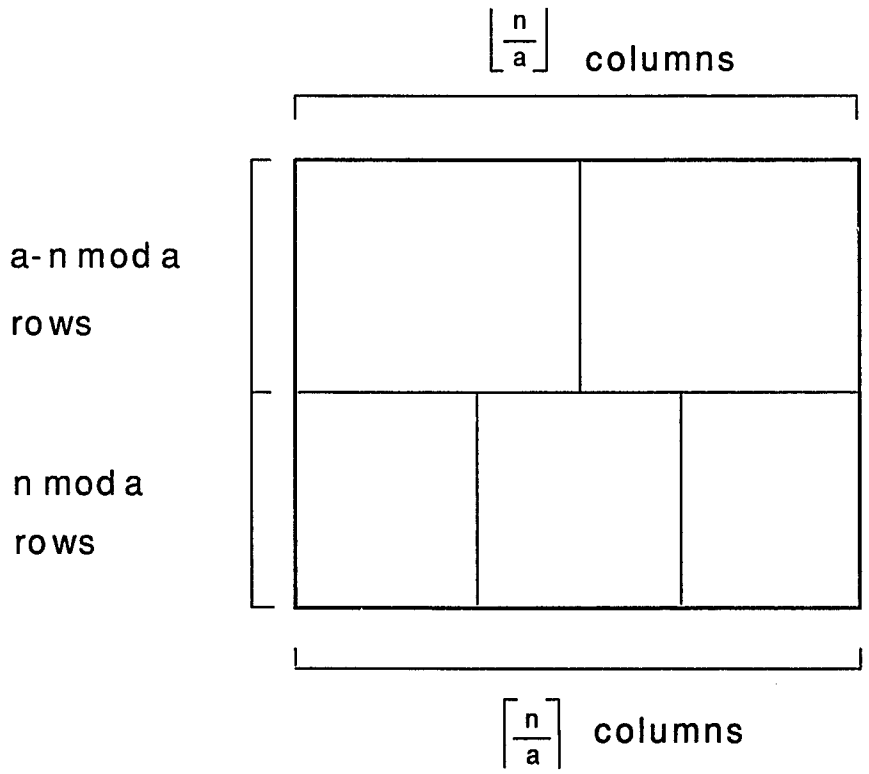

Figure 4.7 Division of plant into regions for a five component system

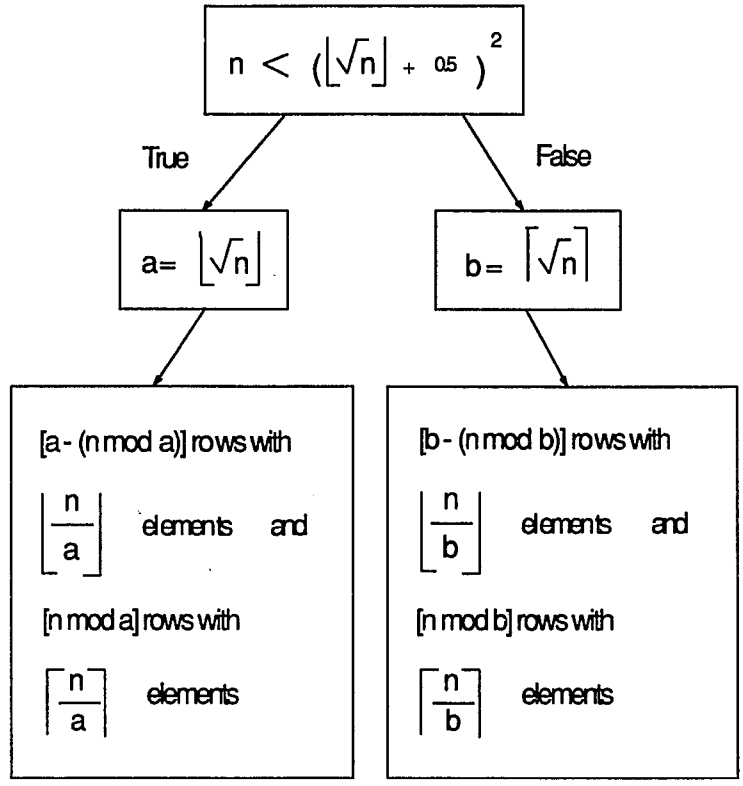

Figure 4.8 The indexing algorithm

through all the sets of component models and computes the value of the resulting discrete objective function as given in Equation 4.18. The computed values are compared with values obtained from other combinations. The minimum value of the discrete objective function is then determined.

The dimensions of the selected set of component models, are then transferred by the C program, to a text file. This text file is accessed by the Foxpro database program, which reads the information into a database file.

A Foxpro algorithm then checks for physical interference between the new component models. If a physical interference exists then an appropriate correction is made by altering the component location. The alteration of the component location is performed till there is no physical interference between components. This mechanism is illustrated in Figure 4.6.

Once a non-interfering set of component models is obtained, the resulting component and connection locations are transferred by the Foxpro subroutine to text files and to the `Optim` file. These text files are accessed by a C-graphics routine which graphs them as previously discussed.

4.11 Determination of the optimal set of connection models

A particular set of connection models is searched such that the discrete objective function connecting the material and size of connection models is minimum. Every connection is replaced by all of its connection models, in a systematic way, and the value of the discrete objective function is recorded for every connection configuration. The minimum value of this objective function is then determined by the brute search method .

The properties of connection models (e.g. material, size etc) are stored in database files. This information is utilized by a Foxpro routine which searches through all the sets of connection models and computes the value of the resulting discrete objective function. The computed values are compared with values obtained from other combinations. The minimum value of the discrete objective function is then determined. The resulting connections models and their properties are transferred to the `Optim` database file.

4.12 Fine tuning of the location of the components in the plant

Once the approximate component locations are determined by minimizing the discrete objective function as given in Equation 4.17, the fine-tuning of the component location is performed by the steepest descent method, so that the value of the objective function is further reduced. This is performed by moving all the components in the plant within a constrained space. The objective function given in Equation 4.17 has variables which take continuous value when the components are placed at continuous locations. Thus the minimum value of this continuous objective function is computed by employing a steepest descent search algorithm (for non-linear constrained equations) as previously explained.

All the required properties of the plant model are transferred from the `Optim` database file to a text file. This text file is accessed by the steepest descent algorithm written in C. The resulting component locations (i.e. fine tuned component locations) are transferred to text files by the C subroutine. These modified component locations are read into a separate database file.

The new locations are then tested for physical interference. If physical interference exists between any two components then appropriate corrections are made as previously discussed. The coordinates of the non-interfering component locations are then transferred by the Foxpro subroutine to text files that are read by a graphics routine in C which draws the updated component locations in the plant.

CHAPTER 5

RESULTS AND DISCUSSION

The results of the various stages in the optimization process are presented in this chapter. The results are presented in the form of graphs which show the reduction of the objective functions and bar charts that display the execution time taken for the relevant iterations.

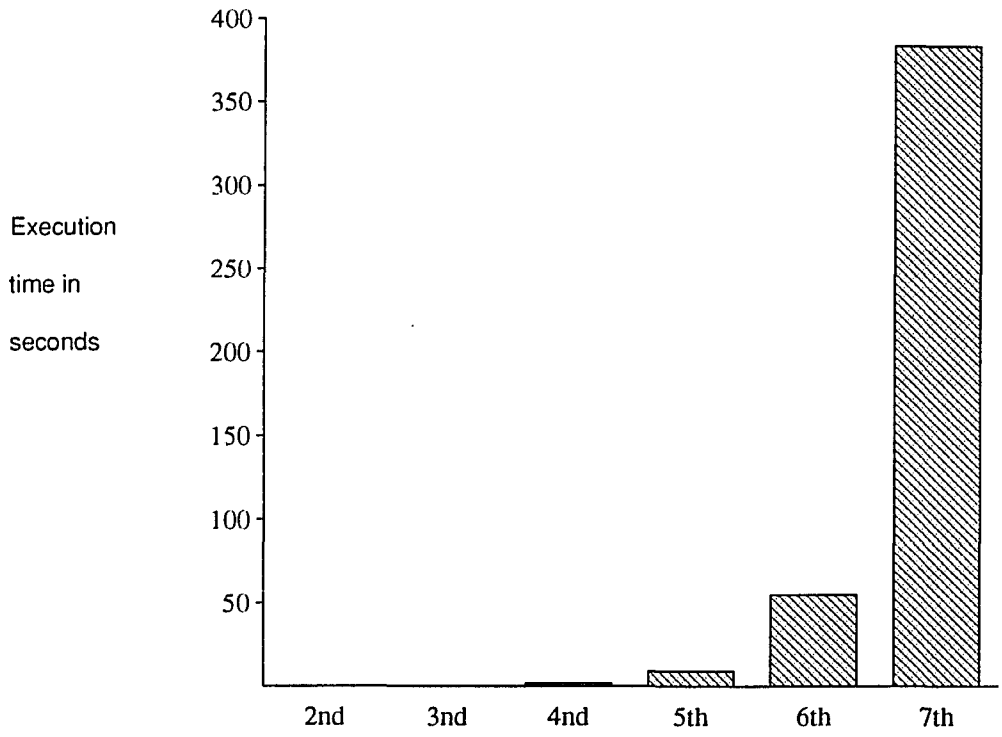

Figure 5.1 Variation in execution time with number of components (using a full brute force search).

5.1 Locating components in the plant

When all components are placed at all possible regions in the plant the number of possible component combinations will be equal to the factorial of the number of components, as discussed in Section 4.92.

The estimated execution time for determining the component combination which results in the minimum value of the objective function as given by Equation 4.17 is shown in Figure 5.1. The ordinate represents the execution time in seconds and the abscissa represents the number of components. It can be seen from Figure 5.1 that the execution time increases to nonpractical values as the number of components increases.

Since the execution time is very large when all components are placed at all possible regions, a different approach was taken to reduce the execution time. This is done by placing connected components (i.e. components that are connected with each other) at adjacent regions. Each connected component is placed at regions that were adjacent to the considered region, both along the vertical and horizontal axis. The number of iterations to be performed in this case will be less than 4^n (assuming that every connected component is placed at 4 regions). This is less than $n!$ for large values of n. The total execution time taken for 9 components when they are placed at adjacent regions (horizontally and vertically) is 7 minutes and 19 seconds. The resulting component configuration is shown in Figure 5.2. The execution time in this case increases to nonpractical values when the number of components increases. Eventhough this method searches for the minimum value of the objective function shown in Equation 4.17, for many component configurations and also reduces the total execution time taken when comparing with the idea of placing all components at all possible regions, it is still nonpractical because of its large execution time.

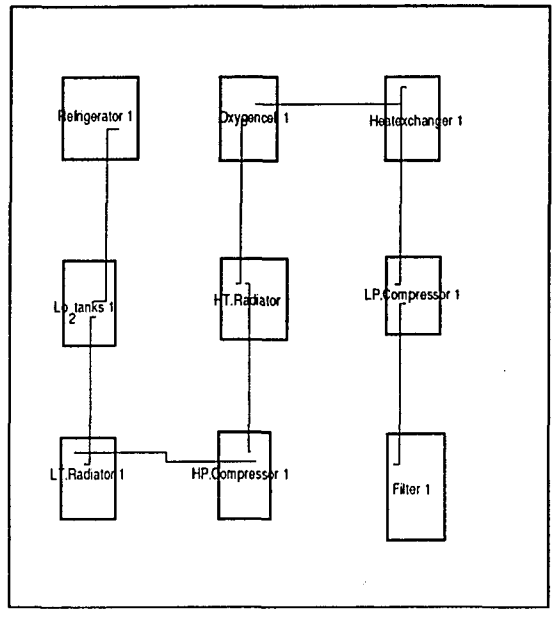

Figure 5.2 Plan of a 9-component system after locating components in plant regions

A different approach was adopted to further decrease the execution time. This was achieved by employing the heuristics of placing only connected components in horizontally adjacent regions as discussed in Section 4.92. This reduces the number of iterations to 2^n, where n is the number of components. This reduction in the number of component combinations reduces the execution time to be within 90 seconds for all components up to 9.

The reduction in the value of the objective function as given in Equation 4.17 for a 9-component system is shown in Figure 5.3. The values used for the weighting factors in Equation 4.17 are 0.01, 10 and 20 for w_{ri} ,w_{fi} and w_{ij} respectively. The ordinate represents the value of the objective function and the abscissa represents the number of component combinations that result in a reduction of the value of the objective function. The resulting component configuration is shown in Figure 5.4.

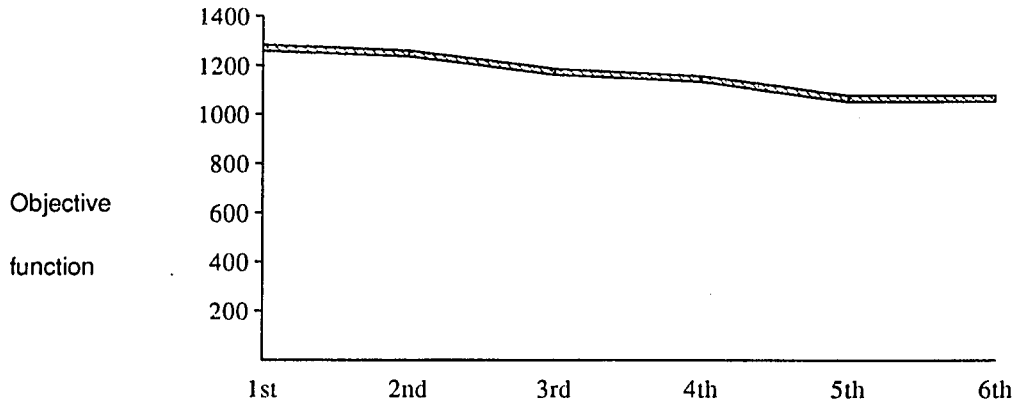

Figure 5.3 Reduction in the value of the objective function (with discrete variables) when locating components in plant regions for a 9-component system.

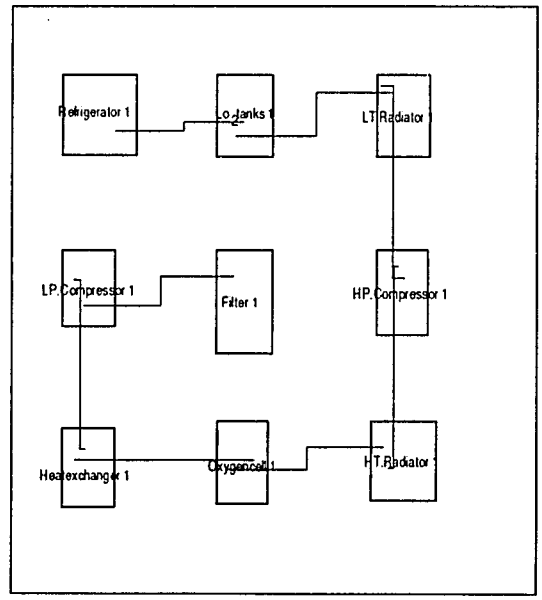

Figure 5.4 Plan of a 9-component system after locating components in plant regions

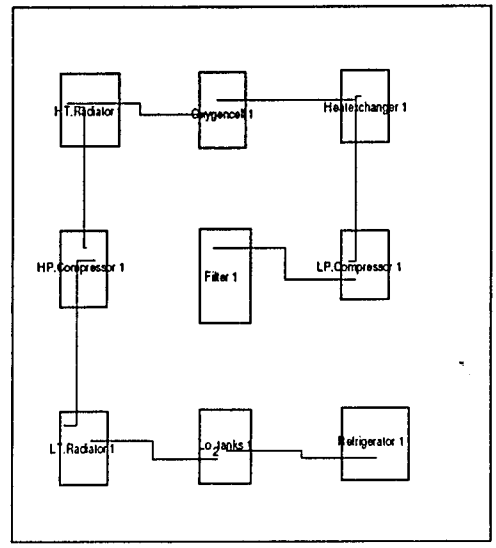

Figure 5.5 Plan of a 9-component system after locating components in plant regions

When the values for the weighting factors shown in Equation 4.17 is changed to 0.01, 10 and 2000 for w_{ri} ,w_{fi} and w_{tj} respectively, a different component configuration is obtained as shown in Figure 5.5. The thermal proximity is proportional to the view factor which is in turn proportional to the distance between any two components as given by Equations 4.13 and 4.14 respectively. The distance between two components is assumed to be equal to the horizontal distance between the connection locations at these components. Since the thermal proximity loss, in this case, is weighted more than that of the other losses it can be seen that the distance between connection locations of the Filter and the L.P. Compressor in Figure 5.5 is more than that shown in Figure 5.4. This explains the effect of the weighting factors.

The reduction in the value of the objective function as given in Equation 4.17 for a 7-component system is shown in Figure 5.6. The resulting component configuration for a 7-component system is shown Figure 5.7. The values of the weighting factors are the same as

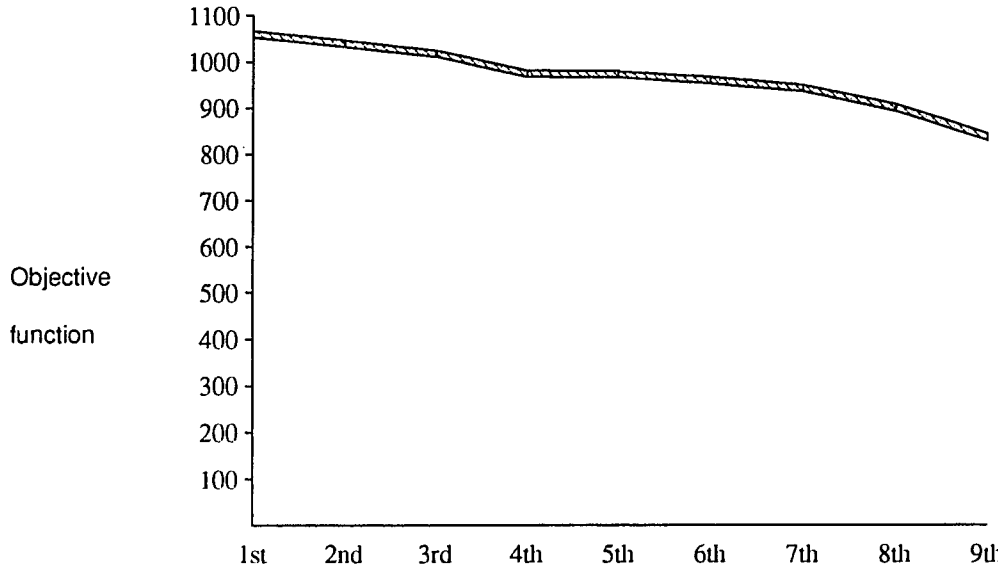

Figure 5.6 Reduction in the value of the objective function (with discrete variables) when locating components in plant regions for a 7-component system.

that for the 9-component system. Since the plant is divided into approximately equal regions, it could be seen that the components are well spaced in this case when comparing the same with that for a 9-component system as shown in Figure 5.4.

5.2 Determination of the optimal set of component models

The execution time for a 9-component system with different number of component models is shown in Figure 5.8. The execution time is usually calculated by printing the time before and after the execution of the algorithm. In this case since the execution time is

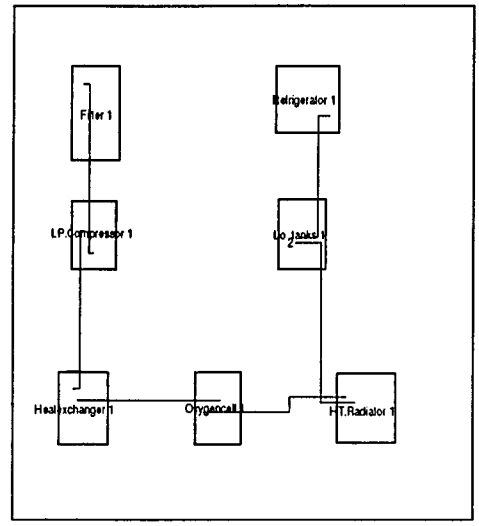

Figure 5.7 Plan of a 7-component system after locating components in plant regions

very high and will soon reach nonpractical values, the execution time for one iteration is found and then total execution time for all the iterations is calculated. The ordinate represents the execution time in seconds and the abscissa represents the number of component models. The bar chart shows the increase in the execution time when all the components in the 9-component system have an increasing number of component models. The total number of iterations is equal to n^n when there are n components with each component having n component models.

It can be seen that the execution time increases to large values as the number of component models increases. This problem could be avoided in future by setting a filter that will reduce the number of component models. The filter could be designed in such a way that every component model would be tested to satisfy certain basic requirements, which are either local to a particular component or a general requirement that should be satisfied by all component models. The models which do not satisfy these requirements are rejected. The execution time

taken by the filtering algorithm will not be large. This idea will reduce the number of component models and will thereby reduce the execution time to practical values.

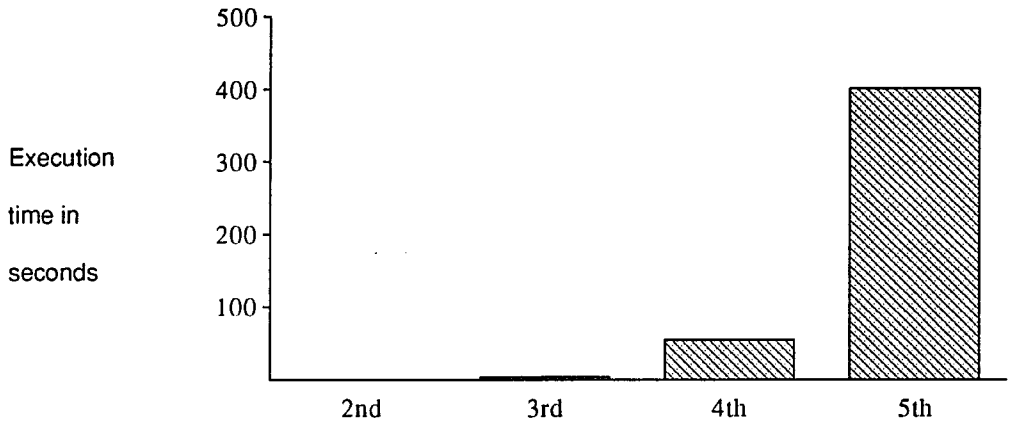

Figure 5.8 Variation in execution time with number of component models in a 9-component system.

The reduction in the value of the objective function (given in Equation 4.18) for a 9-component system is shown in Figure 5.9.

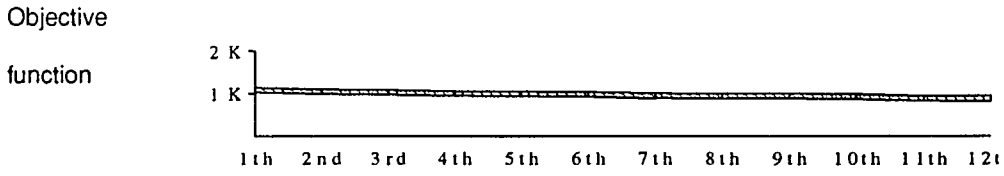

Figure 5.9 Reduction in the value of the objective function (with discrete variables) when a brute force search is performed which searches through all component models (9-component system).

The ordinate represents the value of the objective function and the abscissa represents the number of iterations that result in a reduction of the value of the objective function. The values of the weighting factors w_m and w_c are taken to be 1. The graphs show the value of the objective function of all component model combination whose value is lower than that of the previous component model combination. Since the operations to be performed in choosing the optimal set of component models is not complex there is a very small difference in the execution time between a 7-component system and a 9-component system.

5.3 Determination of the optimal set of connection models

The execution time for a 9-component system with different number of connection models is shown in Figure 5.10.

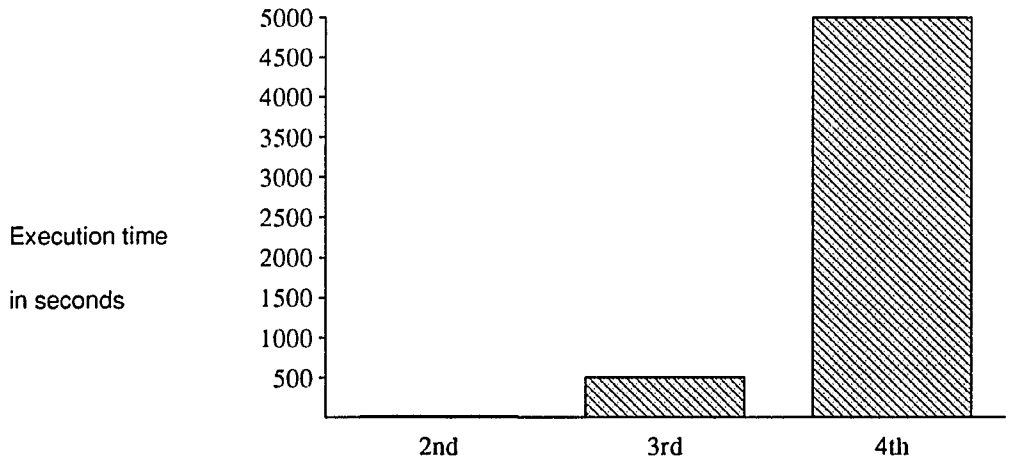

Figure 5.10 Variation in execution time with number of fluid connection models in a 9-component system

The number of iterations is equal to $[n_{cm}]^{n_c}$ where n_{cm} denotes the number of connection models and n_c denotes the number of connections in the plant. The number of connections in the considered 9-component system is 8.

It can be seen that the execution time will increase to large values when the number of connection models increases. This problem could also be solved by setting a filter as discussed in Section 5.2, which checks every connection for some basic requirements. Thus the number of connection models that will be used to find the optimal set of connection models, will be reduced. This will decrease the execution time to practical values when the number of connection models increases.

The reduction in the value of the objective function given by Equation 4.19 for a 9-component system is shown in Figure 5.11.

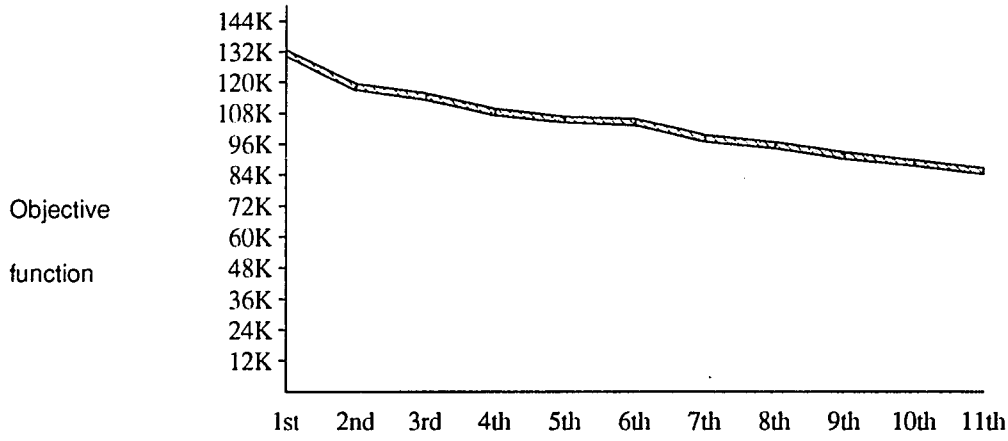

Figure 5.11 Reduction in the value of the objective function (with discrete variables) when a brute force search is performed which searches through all connection models in a 9-component system.

The values of the weighting factors w_{ri} and w_{fi} are taken to be 1 and 1000 respectively. The resulting component configuration is shown in Figure 5.12.

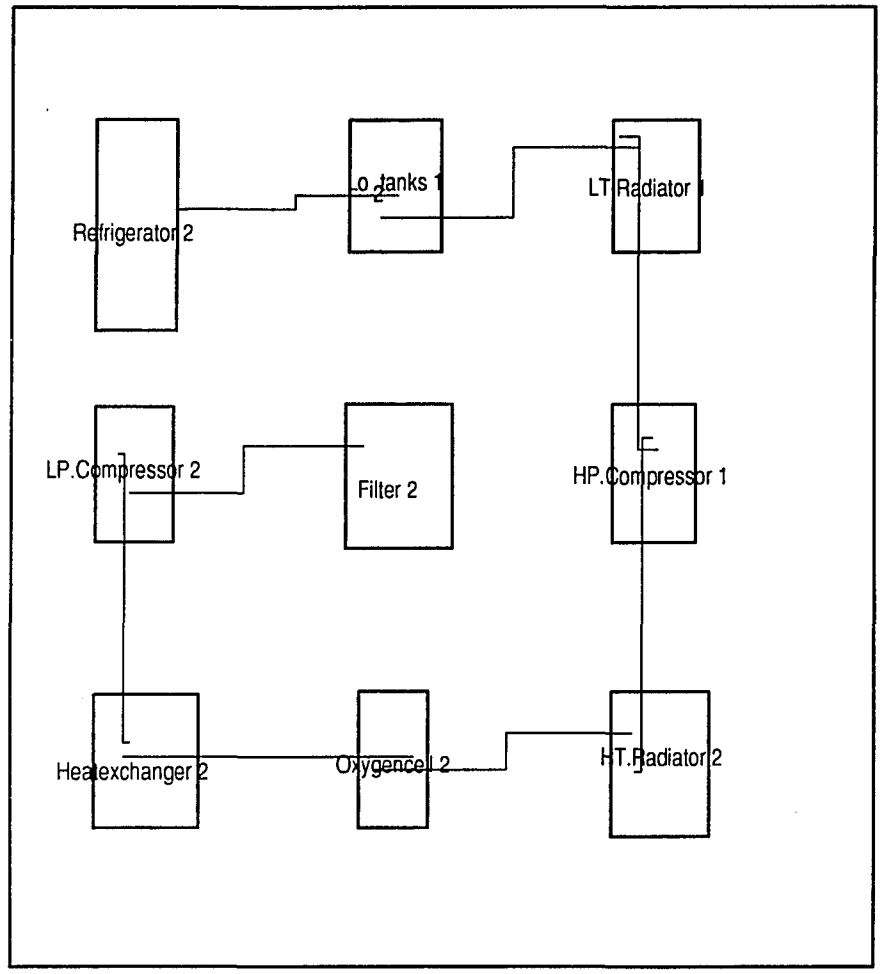

Figure 5.12 Plan of a 9-component system after choosing optimal component and connection models

The corresponding results for a 7-component system are shown in Figure 5.13, Figure 5.14 and Figure 5.15.

Figure 5.13 Variation in execution time with number of fluid connection models in a 7-component system

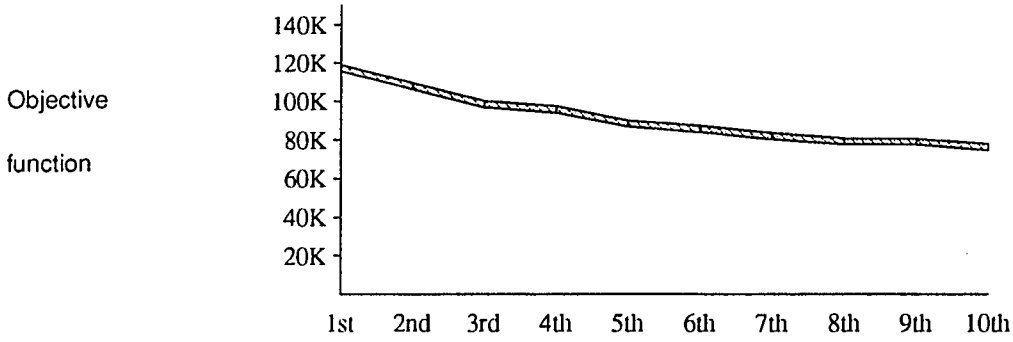

Figure 5.14 Reduction in the value of the objective function (discrete variables) when a brute force search is performed which searches through all connection models in a 7-component system.

The values of the weighting factors are same as that of the 9-component system.

5.4 Fine tuning of the locations of the components in the plant

The fine tuning of the location of the components in the plant is performed by minimizing the objective function shown in Equation 4.17 with continuous variables.

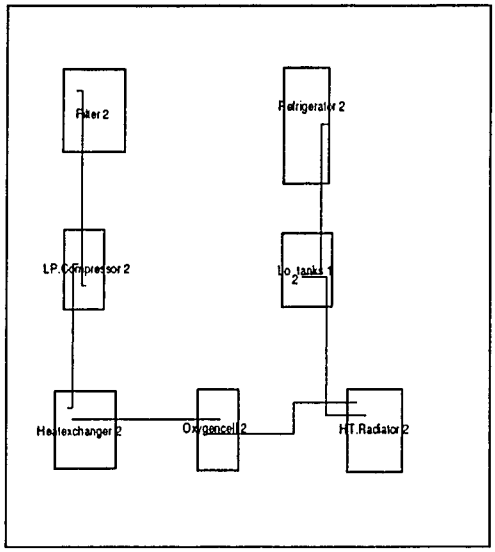

Figure 5.15 Plan of the 7-component system after choosing optimal connection and component models

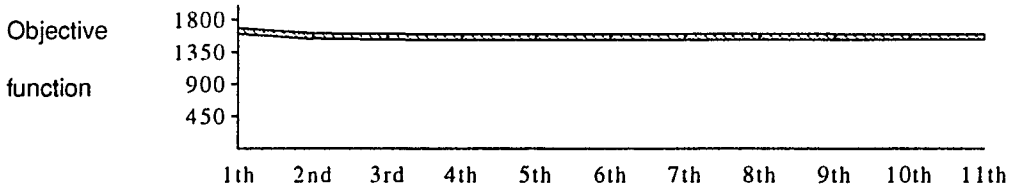

Figure 5.16 Reduction in the objective function (with continuous variables) for a 9-component system when the component locations are fine-tuned by the steepest descent algorithm.

The reduction in the value of this objective function for a 9-component system is shown in Figure 5.16 with the number of iterations as the abscissa and the value of the objective function as the

ordinate. The values used for the weighting factors are 0.01, 10 and 6000 for w_{ri} , w_{li} and w_{ij}, respectively. A higher weighting factor for the thermal proximity losses is taken in order to account for the assumptions that were used in calculating the thermal proximity loss as discussed in Section 4.6.2. The resulting change in the component configuration is shown in Figure 5.17.

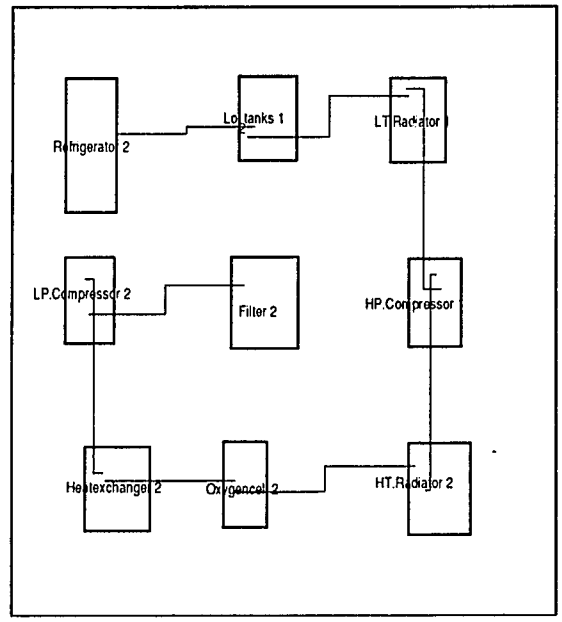

Figure 5.17 Plan of the 9-component system after fine tuning component locations

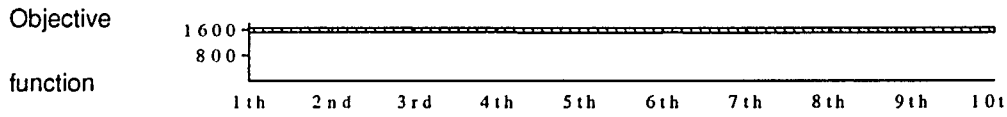

Figure 5.18 Reduction in the objective function (with continuous variables) for a 7-component system when the component locations are fine-tuned by the steepest descent algorithm.

The corresponding results for a 7-component system are shown in Figure 5.18 and Figure 5.19. The values used for the weighting factors in this case are 1, 1000 and 6000000 for w_{ri}, w_{fi} and w_{tj} respectively.

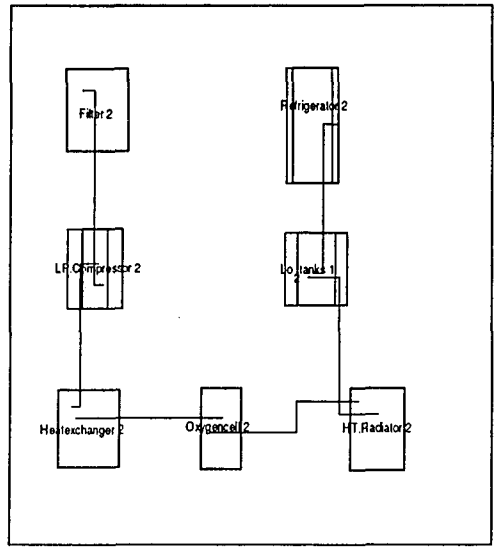

Figure 5.19 Plan of the 7-component system after fine tuning component locations

5.5 Conclusions

A two-dimensional component configuration for the given number of components is obtained. This is done after minimizing the user-defined objective functions given by Equations 4.17, 4.18 and 4.19. This work assumes that each component has one inlet and one outlet. It also assumes that any component can have two connections at the maximum. The radiative heat loss due to the thermal proximity components is assumed to have a parallel plate configuration as discussed in Section 4.62. Further the thermal proximity heat transfer is considered only

between connected components. The radiative heat transfer between non-connected components is ignored in order to decrease the execution time taken as discussed in Section 4.6.2. The distance between surfaces of connected components is assumed to be equal to the horizontal distance between connection locations in these components.

Since recursion is not supported by Foxpro, the main program written in Foxpro calls various subroutines to calculate the facing area in the parallel plate configuration as discussed in Section 4.62. These subroutine calls depend on the number of components in the plant model. When components are placed at adjacent regions (at horizontally located regions) the main program calls other subroutines to perform the operation of arriving at the approximate layout of component models as discussed in Section 4.9. These subroutines change for different number of components. This problem could be solved by exporting the relevant data from Foxpro to text files that can be read by a subroutine written in C which will perform the necessary recursive operations.

5.6 Directions for future work

The present work focuses primarily on a two-dimensional model for components. Future work can be done on a three-dimensional model for components. The thermal proximity losses for components is presently considered only for connected components. A different method or an extension of the present method can be used to evaluate the thermal proximity loss between all components in the plant. This will result in a more accurate modelling of the component system. The present work considers the distance between connection location locations while evaluating the thermal proximity distance (i.e. the surface distance between two components). A better algorithm could be developed which will accurately find the surface distance between two components when evaluating the thermal proximity distance. The present work is based on

model whose components are connected by a two connections. This can be used as a base to construct a model which would have many kinds of connections with each components connected with other components by multiple connections.

References

1. Evans, P. M., "Integrated Structural Design Software in the IBM Architecture and Engineering Series", Proceedings of the 10th Conference on Electronic Computation, ASCE Publications, 1991, 98-105.

2. Vassilev, V., Deyanov, A., Djambov, V., and Kitchovitch, M., "Software Tools for Nonlinear Programming", Proceedings of the 11th Triennial World Congress of the International Federation of Automatic Control, Pergamon Press Inc, **3**, 1991, 403-407.

3. Chiu, G. and Ragavendra, C.S., "A Model for Optimal Database Allocation in Distributed Computing Systems", Proceedings of IEEE INFOCOM 1990; Ninth annual Joint Conference of the IEEE Computer and Communications Societies, IEEE publications, 1990, 827-833.

4. Hirano, T., Yamada, T. and Teraki, J., "The Inverse Analysis Method Combining Knowledge Engineering with Mathematical Programming", A Hen/Transactions of the Japan Society of Mechanical Engineers, **57**, Jan 1991, 209-214.

5. Sadeghi, T and Wozny, M.," Computer-Aided Design of Control Systems via Parameter Optimization Method (CADCS/POM) - An Interactive Graphics Approach", 1984 American Control Conference, IEEE publications, **3**, June 1984, 1634-1640.

6. Mulleneers, R.G.A., Dassen, W.R.M., Smeets, J.L.R.M., Brugada, P. and Wellens, H.J.J., "Integrated Cardiology Retrieval and Storage System (ICARUSS) --A Data Management System for the Cardiologist Working in Electrophysiology", Proceedings - Computers in Cardiology, IEEE publications, Sep 1989, 493-496.

7. Ramohalli, K.N.R. and Sridhar, K.R. "Extraterrestrial Materials Processing and Related Transport Phenomena", Plenary Lecture presented at AIAA 29th Aerospace Sciences Meeting, A.M.E. Research Report 91-5, Jan 1991, 1-4.

8. Ramohalli, K.N.R., Lawton, E., and Ash,R. , "Recent Concepts in Mission to Mars: Extraterrestrial Processes", IAF-86-154 at the 37th Congress of the international Astronautical Federation, American Institute of Aeronautics and Astronautics. Inc, 5,Mar-Apr. 1989, 181-186.

9. Venkatesh, A.I. and Sridhar, K.R., "Thermal Analysis, Optimization and Design of a Martian Oxygen Production Plant ",NASA - Space Engineering Research Center for utilization of local planetary resources - University of Arizona, Annual Progress Report 1990-91- Apr-91, IB 9-14.

10. Ramohalli, K.N.R. "Technologies of I.S.R.U./ I.S.M.U.", Proceedings of the 42nd Congress of the International Astronautical Federation - IAA-91-659, A.M.E. Research Report 91-24, Oct 1991, 1-5

11. Jones, E., "Foxpro Made Easy", McGraw-Hill publications, 1990, 1-10

12 Robert, T.L., "A Practical Introduction to Impedance Matching", Artech House Publications, 1976, 1-3.

13 Rao, S.S., "Optimization Theory and Applications", Wiley Eastern Limited publications, second edition, July 1984, 1-29.

14 Ecker, J. G. and Michael, K., "Introduction to Operations Research", Wiley Publications, 1988, 304-320.

15. Vincent-Genod, J. "Fundamentals of Pipeline Engineering", Gulf Publishing Company, 1984, 20-25.

16. Incropera, P. F. and De Witt, P. D., "Introduction to Heat Transfer", John Wiley and Sons publications, second edition, 1990, 752-762.

17. Siegel, R. and Howell, R. J., "Thermal Radiation Heat Transfer", Hemisphere Publishing Corporation, second edition, 1981, 187-189