Order Number 9021702

Computer-aided systems for environmental engineering decision-making

Kao, Jehng-Jung, Ph.D.

University of Illinois at Urbana-Champaign, 1990

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

COMPUTER AIDED SYSTEMS FOR ENVIRONMENTAL ENGINEERING DECISION MAKING

BY

JEHNG-JUNG KAO

B.S., National Cheng Kung University, 1982
M.S., University of Illinois, 1987

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Environmental Engineering in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1990

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

JUNE 1989

WE HEREBY RECOMMEND THAT THE THESIS BY

JEHNG-JUNG KAO

ENTITLED COMPUTER AIDED SYSTEMS FOR ENVIRONMENTAL ENGINEERING DECISION MAKING

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

E D Brill
Director of Thesis Research

W J Hall
Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

O-517

ABSTRACT

This research explores the use of computer-based environments to facilitate environmental engineering decision-making. Two prototype systems are developed as exploration tools and to demonstrate the techniques and principles proposed. Several mathematical techniques, a modeling language, interactive graphic displays and user friendly interfaces are used. The mathematical techniques are: (1) linear programming, (2) a finite element method for a groundwater simulation model, (3) mass and water balances for an analysis program for wastewater treatment plant design, (4) the Vector Method to obtain the exact noninferior set of a multicriterion problem, and (5) the Modeling-to-Generate-Alternatives (MGA) approach for generating potential alternatives for a decision-making problem. The modeling language is designed to relieve the analyst of the burden of formatting a general mathematical model for solution using existing mathematical programming packages. The interactive graphic displays provide visual data for effective comparisons, and the user friendly interfaces are designed for engineers who are not necessarily computer experts. The two computer aided systems are for wastewater treatment plant design and groundwater resources management.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In analyzing environmental engineering decision making problems, such as the wastewater treatment plant design (WTPD) problems or the groundwater resources management (GRM) problems described in chapters 5 and 6, cost is not the only important issue, and other modeled as well as unmodeled issues must usually be considered, e.g. uncertainty, reliability, equity, etc. Since such problems are complex, exact mathematical methods to solve them are not available. Furthermore, presentation of models or alternatives is usually difficult, and the computer interfaces needed to modify or rebuild a model are cumbersome. This research explores approaches for dealing with these issues by means of two prototype computer aided systems using several mathematical techniques, a modeling language, graphic displays, and user-friendly interfaces to deal with these issues for the WTPD and GRM problems, respectively.

The first prototype computer aided system is being developed for the design of wastewater treatment plants. Such a system should contain the tools for analysis and preliminary design of environmental engineering processing plants. The concept of "design" implies selection of process chain, determination of mass and water, and facility cost estimation.

One procedure for designing a wastewater treatment plant involves establishing influent and effluent conditions, selecting unit processes for an appropriate treatment train, applying appropriate performance models for unit processes selected, setting up a mathematical model, solving the mathematical model to find a feasible design, estimating the total cost, generating and comparing alternatives, and checking the design against standards. Such a procedure is complex, time-consuming, and sometimes tedious. The computer aided system presented in this research is intended to facilitate the procedure by eliminating the burden on a designer for formulating a model, solving the model, generating alternatives, etc. Moreover, the computer aided system increases the power of the traditional trial-and-error procedure. A trial-and-error procedure is usually tedious if each trial takes a long time to set up and finish. However, the trial-and-error procedure would be powerful if only pressing several buttons were needed to finish a trial with the results presented immediately to the designer. The computer aided system provides almost real-time responses to the designer; this advantage lets the designer easily generate alternatives, compare alternatives under different design conditions, and significantly shorten the time required to analyze the design of a wastewater treatment plant. The prototype has been drawn from the wastewater

treatment plant models developed by Tang, et al (1984). A general analysis and cost-estimation program has been developed.

The second prototype computer aided system is for groundwater resources management. A simulation model, optimization technique, multiobjective programming, alternatives generation approaches, interactive graphic display, solution management, and a friendly user interface are incorporated into the system.

To analyze a GRM model, many issues other than cost are usually considered. The general purposes of groundwater resources management are: (1) to determine the potential yield; (2) to allocate groundwater resources to competing water demands; (3) to control groundwater quality; (4) to prevent undesirable overdraft of the groundwater basin; (5) to analyze the impact of hydraulic or other characteristics; and (6) to maximize the benefits. A cost optimal solution generally does not consider all of these issues. One approach to deal with these issues is a multicriteria technique. By examining the tradeoff curve (or noninferior set), it is believed that insights may be gained that will assist in making a good decision. However, a GRM model may contain uncertain or unquantifiable issues. For example, there may be uncertainties in values of hydraulic parameters. Unquantifiable issues such as social values usually cannot be mathematically modeled. For such an incomplete model, it is desirable to generate a variety of alternatives for evaluating the effects of uncertain or unquantifiable issues. Alternatives generation techniques are thus incorporated in the prototype. For making a decision, it is unavoidable that a significant amount of comparisons must be implemented. Alternatives would be generated, replaced, or modified during these comparisons. From earlier experiences [Brill et al., 1988], only a small number of alternatives can be compared at the same time by a decision maker, and comparisons may be made most effectively by graphical presentations. A user-friendly interface with graphic displays is used in the prototype to facilitate the comparison tasks. In addition to the advantages described above, the prototype can also reduce work complexity, as mentioned for the WTPD system, for tasks such as formulating a model, modifying a design, etc.

The prototypes have been implemented on the Apollo workstations. Although the discussions above and in Chapters 3 to 6 focus on the design of computer aided systems for two particular models, WTPD and GRM, the issues raised in developing the systems also apply to many other engineering models.

Several techniques were used to develop the computer aided systems. The techniques include a finite element method, mass and water balances, linear programming, multiobjective programming, modeling-to-generate-alternatives (MGA) methods, a modeling language, an interactive graphical

display, and a user interface. These techniques, except for the finite element method, mass and water balances, and linear programming which are common tools, are briefly described as follows.

Multicriterion optimization approaches to aid decision making have been widely used in many disciplines in recent years. The tradeoff (or noninferior or Pareto optimal) set, consisting of solutions in which no objective can be improved without making others worse, provides an analyst insight into a design problem. The analyst or decision maker can examine this noninferior set and may produce meaningful alternatives or make an appropriate decision based on a utility function or preference information. Although the importance of examining a noninferior set is widely recognized, difficulty exists in efficiently obtaining the complete noninferior set for problems with three or more objectives. The Vector Method described in Chapter 4 is modified from the noninferior set estimation (NISE) method [Cohon et al., 1978] for approximating a noninferior set accurately and efficiently. Usually, the noninferior set of a linear problem is approximated using a set of noninferior points. It may, however, be difficult to determine the noninferior set correctly. The exact noninferior surface can be obtained by using the Vector Method. This makes it possible to analyze a problem directly using the complete noninferior surface.

Another technique, MGA, was developed and used in another context. If there are several unmodeled issues or uncertainties, it may not be meaningful to attempt to analyze a noninferior set and to attempt to locate a best compromise solution [Brill, 1979]. Unmodeled issues may be known during the mathematical model development stage, or they may be raised during the analysis process. MGA techniques have been designed to deal with incomplete or changing models (e.g, Brill [1979], Chang [1983], and Kshirsagar [1984]). Since a decision maker can usually handle only a small set of alternatives at a time, it is desirable to have alternatives that are significantly different in decision space. For this purpose, MGA methods generate maximally different alternatives.

In recent years, computer software and hardware have been developing rapidly. Many problems which seemed to be too large can now be handled readily. Today, important problems in modeling usually relate to development, maintenance, and presentation of the model as opposed to just model solution. Different modelers would use different modeling forms, and different software packages usually use different input forms. The modeling forms used by modelers generally are easily understood by other modelers, but input forms for software packages are usually hard to maintain and modify. One early approach was to use a matrix generator which took a code in a structural syntax and translated the code into an input form used by a package. The drawback in using a matrix generator is that it requires some programming skill. Learning a matrix generator sometimes is as difficult as learning a computer language. A better solution is to develop a language which requires no programming skill. Although some modeling

languages recently developed require only a little programming skill, they are usually designed to use a restricted formatted modeling form. The developed modeling language is in a syntax which can be used to construct most modeling forms used by modelers and requires minimal programming skill. It meets the characteristics described by Fourer[1983]; the modeling forms are symbolic, general, concise, and understandable.

A decision making problem may require many comparisons before a decision is made. Thus, graphical presentation of data instead of tabular is used to make the comparisons easier. Additionally, a graphic display usually can be used to provide a comfortable interface. An interactive graphical display with a pointing device, a computer mouse, was used for this research. The designs of the user interface of the current prototypes are based on experience in developing software for engineers who are not necessarily computer experts.

By combining all mathematical techniques described above and using the developed modeling language and user-friendly graphical interfaces, two prototype computer aided systems have been developed for two environmental engineering decision making problems. The systems are described and demonstrated in Chapters 5 and 6.

1.1.Thesis Outline

Chapter 2 reviews literature from a variety of disciplines on issues and techniques for developing computer aided systems. Chapter 3 discusses the characteristics of computer aided systems in general. Techniques and their contributions to decision making processes are described. Chapter 4 presents several new techniques, the Vector Method, modeling language, and new MGA methods. Chapter 5 presents the prototype for WTPD. A discussion of design approaches and a demonstration of the prototype are detailed. Chapter 6 presents the prototype for GRM. Discussions and research conclusions are given in Chapter 7. Appendix A shows the algorithm to formulate a groundwater simulation model and compute impact coefficients. Finally, the program structures of the computer aided systems are provided in Appendix B.

CHAPTER 2

LITERATURE REVIEW

Decision making is an iterative process of examining, modifying, comparing, and selecting preferred solution(s) among many feasible alternatives. This process is not a single step optimization analysis. Numerous attributes or criteria may be employed in evaluating the alternatives. The criteria, however, may be conflicting, and the best solution may not be obvious. To generate alternatives, to do analysis tasks, and to present the solution(s) to a decision maker may be difficult. Also, there may exist unmodeled issues or uncertainties, and thus a mathematically optimal solution(s) may not imply the best solution. There is no single method available to deal with these issues. However, by drawing on the literature on wastewater treatment plant design, groundwater resources management, computer aided systems, multiobjective techniques, alternative generation techniques, and modeling languages, a variety of applications and results are discussed.

Wastewater Treatment Plants Design

Wastewater treatment plant design (WTPD) is an important environmental engineering decision making problem. Many unit processes and characteristics of chemical, physical, or biological reactions are not well understood. Usually, a designer must use his experiences and a trial and error procedure to deal with these uncertainties for developing a sound design.

Geselbracht et al. [1988] used a rule-based technique to develop an approximate reasoning model for sludge bulking judgment. Two sets of 15 plant designs evaluated by an experienced engineer were used in calibrating the model. The model is designed not only for evaluating the bulking potential of an existing design but also for incorporation directly into an optimization model to determine the increased cost of reducing the likelihood of bulking.

Since a WTPD model is highly nonlinear, mathematical difficulty generally exists. Although several models had been developed for optimizing an activated sludge plant (see Tang[1987], Uber[1988], and Kao[1987]), the modification of the models for different plant schemes is difficult and sometimes the modified models can not be solved by currently available mathematical software. Tang [1984] formulated a comprehensive WTPD model. The model is used as a base design for the prototype computer aided system developed in this research. However, since a variety of plant configurations can be used, Tang's optimization model is not used in the prototype because of mathematical difficulty.

Groundwater Resources Management

Groundwater resources are significant water resources in this country and much research has been carried out to examine management issues.

Louie et al. [1984] used the constraint method for analyzing a multiobjective water resources management problem. The three objectives considered were: water supply allocation, water quality control, and prevention of undesirable overdraft of the groundwater basin. Only a subset of noninferior solutions was generated, and the payoff table approach was used to compare solutions.

Willis et al. [1984] presented a bi-objective optimization model for groundwater planning. Three objectives considered in pairs were: total water deficit and (1) maximum pumping rate, and (2) minimum permissible head values in the aquifer system. Instead of calculating impact coefficients, a finite element simulation model was formulated and included in an optimization model. Tradeoff curves were generated by the constraint method.

Reichard [1987] presented an optimization model for analyzing the benefits of basinwide groundwater management in agricultural areas. Reservoir operation was found important for its effect on the availability of streamflow for groundwater recharge, which is a significant factor for optimizing the benefits. A case study for areas in the Salinas Valley in California was presented. The research demonstrated the advantage of using basinwide groundwater management and/or reservoir operation to increase the total revenue, compared with the total revenue gained from uncontrolled pumping without reservoirs.

Wagner et al. [1987] incorporated uncertainty of model parameters into the decision-making process for optimal groundwater quality management. A finite element simulation model, first-order first- and second- moment analysis, and nonlinear chance-constraint stochastic optimization method were used to deal with the uncertainty.

Marin et al. [1988] formulated a nonlinear optimization model based on a spatial equilibrium approach for screening level water resources assessment. Four objectives considered were supply-demand balance, economic efficiency, cost recovery, and equity. A case study for groundwater resources management in Cyprus was presented and analyzed. A near-optimal solution was generated using an approach similar to the MGA approach. The local or global nature of the optima obtained was not discussed.

The groundwater resource management problem discussed in Chapter 6 is similar to the models used by Louie et al. [1984] and Willis et al. [1984].

Computer Aided Systems

A decision making process may consist of two stages: exploration of possible good alternatives by analyst(s), and then examination of those alternatives by decision maker(s). The two stages may be iterative since no compromise solution may be selected initially, and the analyst(s) may be required to explore more alternatives. For both stages, there is complexity in modeling, modifying, and presenting the problem, especially if an interactive decision making process is necessary. To reduce the working complexity, several ideas for developing computer aided systems have been proposed and demonstrated in the literature.

Johnson et al. [1980] incorporated computer graphics in an interactive multiobjective decision making process for water supply planning. The computer graphics provided not only a rapid means of information transfer, but also an effective interface for better understanding and evaluation.

Teitelman [1984] and Goldberg [1984] discussed the display-oriented and object-oriented environment to assist programming. By using the proposed working environment, the task of programming can be significantly improved.

Sagie [1986] developed a computer-aided modeling and planning system for general linear problems. Several languages were provided for data definition, model definition, picture definition, and text definition. A multilingual capacity was made available by translating a key word dictionary from English to other national languages. The system was controlled by command languages which require knowledge of computer programming and linear programming.

Dyksen et al. [1987] demonstrated the application of an interactive problem-solving environment for elliptic partial differential equations. The environment utilized a high level menu driven language, a high-resolution graphics terminal, and a FORTRAN routine library for solving elliptic partial differential equations interactively and graphically.

Cohen et al. [1987] presented the application of an intelligent workstation for electrocenter design. The enhancement of an engineer's productivity and improvement in the creative processes for engineering design were discussed.

Brill et al. [1989] implemented an experiment for evaluating modeling-to-generate-alternatives approaches by using a design system for airline network. The system, called interactive design

environment for airline systems (IDEAS), used features of a workstation environment to aid in the design of airline networks. A graphical tutorial was provided for subjects to learn the system in less than thirty minutes. IDEAS is the first computer aided system developed by the author, and several ideas used for the two computer aided systems described in this thesis are from the experience of developing IDEAS.

In view of the current literature in the field of environmental engineering there are few satisfactory computer aided systems for decision making problems. Possible reasons are the complexity of a decision making process and the difficulty of integrating decision support tools into a friendly working environment. Currently, tools and environments which exploit the high-resolution graphics capabilities of independent workstations are being developed. In this research the workstation environment is used to implement mathematical tools to reduce the working complexity encountered in a decision making process.

Multiobjective programming (or Multicriterion Decision-Making)

To deal with several conflicting objectives, mulitobjective programming techniques have been suggested. A review and introduction of available techniques can be found in Cohon et al. [1975] and Hwang et al. [1980]. Two applications by Willis et al. [1984] and Louie et al. [1984] have been described above. Three other typical applications are described as follows.

Loparo et al. [1980] presented an application of the surrogate worth trade-off method [Haimes, 1974] for multiobjective statistical optimization of interior drainage systems. A case study for the interior drainage system in Moline, Illinois, was discussed.

Gershon et al. [1983] compared four multiobjective decision making techniques, the ELECTRE method, compromise programming , cooperative game theory, and multiattribute utility theory, for river basin planning and presented a case study for the Santa Cruz River Basin. No significant difference was observed for results obtained by the four techniques.

Tecle et al. [1988] applied three multicriterion decision making techniques for selecting an appropriate management scheme. The techniques are compromise programming, cooperative game theory, and ELECTRE I. The Nogales International Wastewater Management Project was the case study used to evaluate the three techniques.

Although the applications of multiobjective techniques to several real world problems are demonstrated in the papers described above, there are difficulties in obtaining the complete noninferior set and presenting the noninferior solutions. The Vector Method introduced in Chapter 4 is efficient for

generating the complete noninferior set for linear problems, and the graphical display can improve the presentation without pruning too many solutions.

Modeling-To-Generate-Alternatives

Starr and Zeleny [1977, p. 25] described decision making as follows:

> Decision making is a dynamic process: complex, redolent with feedback and sideways, full of search detours, information gathering and information ignoring, fueled by fluctuating uncertainty, fuzziness and conflict; it is an organic unity of both pre-decision and post-decision stages of the overlapping regions of partial decisions.

Sometimes a real world problem is not easy to model mathematically. There may exist several uncertainties or unquantifiable issues. Several other issues may be raised by different perceptions of the model during analysis. To deal with complex problems and dynamically changing models, the MGA approach has been suggested by Brill [1979]. The MGA approach was first applied to public-sector planning to deal with unmodeled issues. The idea is to generate maximally different but good alternatives when compared with the optimal solution for the model. Through examining several maximally different alternatives generated by using the MGA approach, it may be possible to obtain insight into the system and to consider the unmodeled issues.

Nakamura et al. [1979] applied the MGA approach to a model for regional wastewater systems using an extended branch-and-bound method. Chang et al. [1982] introduced new MGA techniques and illustrated them for water resources and land-use planning problems. Several MGA methods, including the Hop-Skip-Jump (HSJ) method, were developed and applied; the HSJ method was applied to a 0/1 integer program in addition to continuous variable problems. A fuzzy approach by Chang et al. [1983] was also developed for implementing the MGA concept for the same problems.

Chang et al. [1984] employed the MGA technique to generate alternatives for a wastewater treatment system. Their idea is similar to that used in the proposed research: they incorporated several MGA approaches to obtain alternative solutions with objective values close to the optimal value.

Kshirsagar et al. [1984] developed a generalized HSJ method, which is interpreted as an inner product measure in ideation space, for implementing the MGA approach. A land-use planning problem was used to illustrate the method. Clustering analysis was used to group generated alternatives for comparison.

The usefulness of the MGA approach in a decision making process has been described in these papers. One general method to implement the MGA approach is the HSJ method; it is simple and computationally efficient and is used in this research.

Modeling Language

Greenberg [1983] wrote, " Comprehension is the present bottleneck in using large scale models– in particular, linear programs." Although many mathematical programming packages can produce useful results for analyzing a system, presenting the results, understanding the presentation, and modifying the model are generally time-consuming tasks. To improve the presentation and maintenance of models for increasing a modeler's productivity, several matrix generators and modeling languages have been developed. They are described as follows.

Ellison et al. [1982] developed a matrix-generator and report-writer system for mathematical programming. The system allows a modeler to define data in structural forms. The language syntax is similar to the COBOL syntax. Although the system is useful for modeling and presentation, it is not easy to use without programming skill.

Fourer [1983] discussed general drawbacks of existing matrix generators and proposed characteristics of a hypothetical modeling language. Comparisons among matrix generators and between matrix generators and the modeling language were provided. The hypothetical modeling language was discussed based on a specific modeling form suggested.

Lucas et al. [1988] presented a computer-assisted mathematical programming system. The system uses a menu-driven approach to construct a model. A modeler is taken into several screen forms to set up a model in an author-preferred sequence. A model is divided into several sections which must be defined separately, e.g. all constants and variables must be defined at the beginning of modeling. The language syntax requires only a little programming skill, but it is not flexible enough to use a modeling form other than the one incorporated into the system.

Paul [1989] described a commercial modeling language, LINGO/PC, for use in linear and integer programming. Although the capacity of the language for large-scale problems is attractive, the syntax of the language is like a computer programming language.

The developments of modeling languages described above are mostly rooted in the characteristics of a general computer programming language. A modeling language, however, would be easier to use if it were simpler than a programming language. The modeling language should be close to the modeling

forms commonly used and have flexibility in forms allowed. A modeler should be able to write a model easily and quickly in a computer-understandable form using the modeling language. The modeling language developed in this research is intended to provide this capability.

CHAPTER 3

COMPUTER AIDED SYSTEMS

For a complex decision making problem, such as the GRM or WTPD problems described in Chapters 5 and 6, it is usually difficult to set up or modify a model, to generate potential alternatives, and to present the model or alternative solutions. This chapter first describes, in general, how to use computer aided systems to reduce the complexity of these tasks based on an assumed working process. The techniques used in developing two prototype computer aided systems are then discussed. A discussion of relationships among decision maker(s), analyst(s), computer aided system(s), and decision making process(es) follows. Finally, issues such as extensions and suggestions for improving the computer aided systems are discussed.

3.1. General Issues Related to Computer Aided Systems

Figure 3.1 shows a general process for decision making using mathematical models. Before a compromise solution is selected by a decision maker(s), the working process is expected to be implemented iteratively. This research has focused on six of the stages (A through F in Figure 3.1). Figure 3.2 shows where to incorporate new features of computer aided systems into the working process for decision making. Two new techniques, the Vector Method, modeling language, and two new MGA methods and their contribution to the process are discussed in Chapter 4. The following sections provide general discussions of the contributions of the other techniques (mathematical techniques and tools, graphical interface, and user interface) to the process.

3.2. Mathematical Techniques and Tools

Linear programming, mass and water balances, finite element method, multiobjective programming, and an MGA approach are mathematical techniques used for developing the prototypes. Except for the MGA approach and the newly developed Vector Method, all of the techniques are commonly used and their concepts and applications are widely described in the literature. Details of the Vector and new MGA methods are described in Chapter 4.

Mathematical techniques were used for doing simulation analysis, obtaining mathematical solutions (stage C in Figure 3.1), and generating alternatives (stage D). All mathematical techniques were coded in FORTRAN except the mass and water balances which were coded in PASCAL.

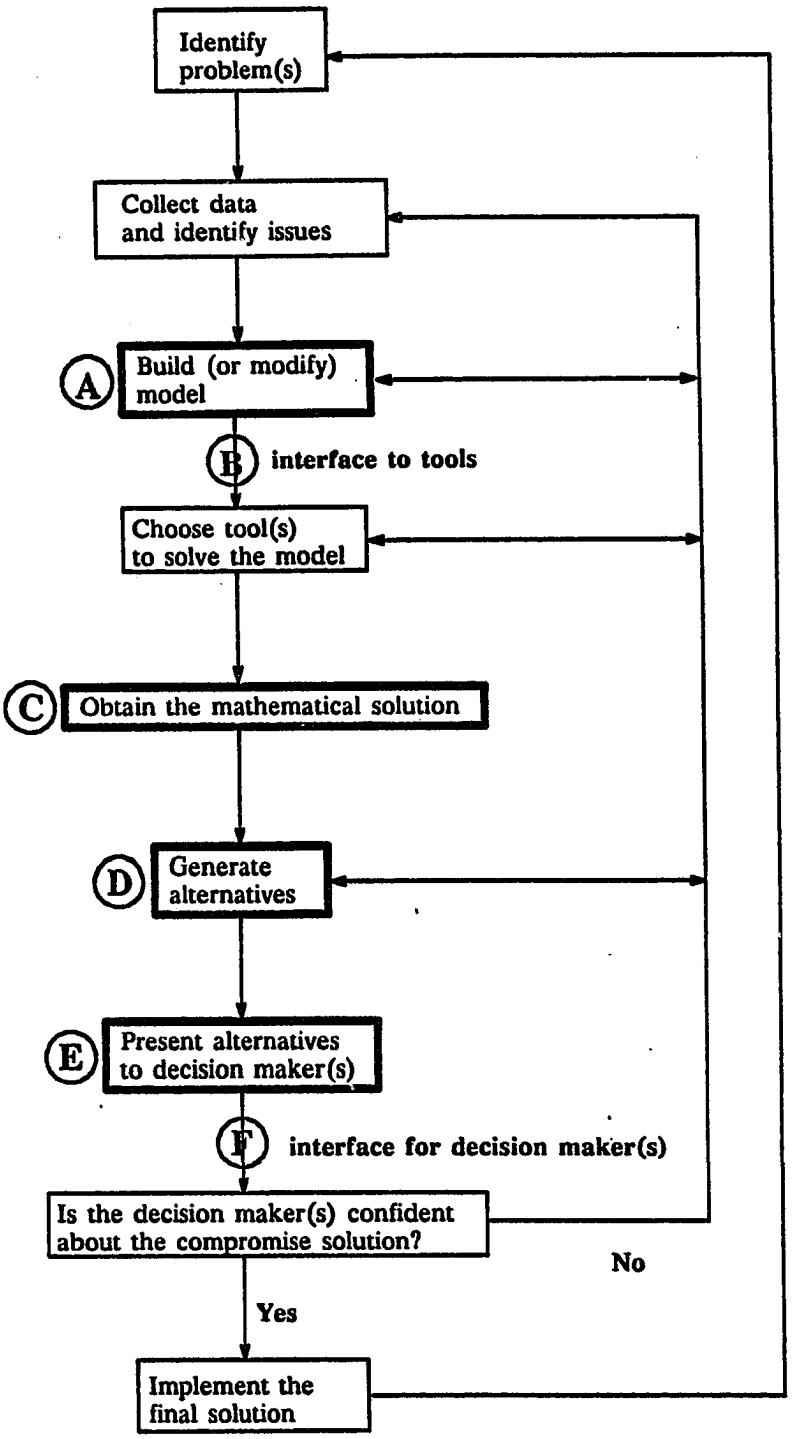


Figure 3.1. A Working Process For A Decision Making Problem

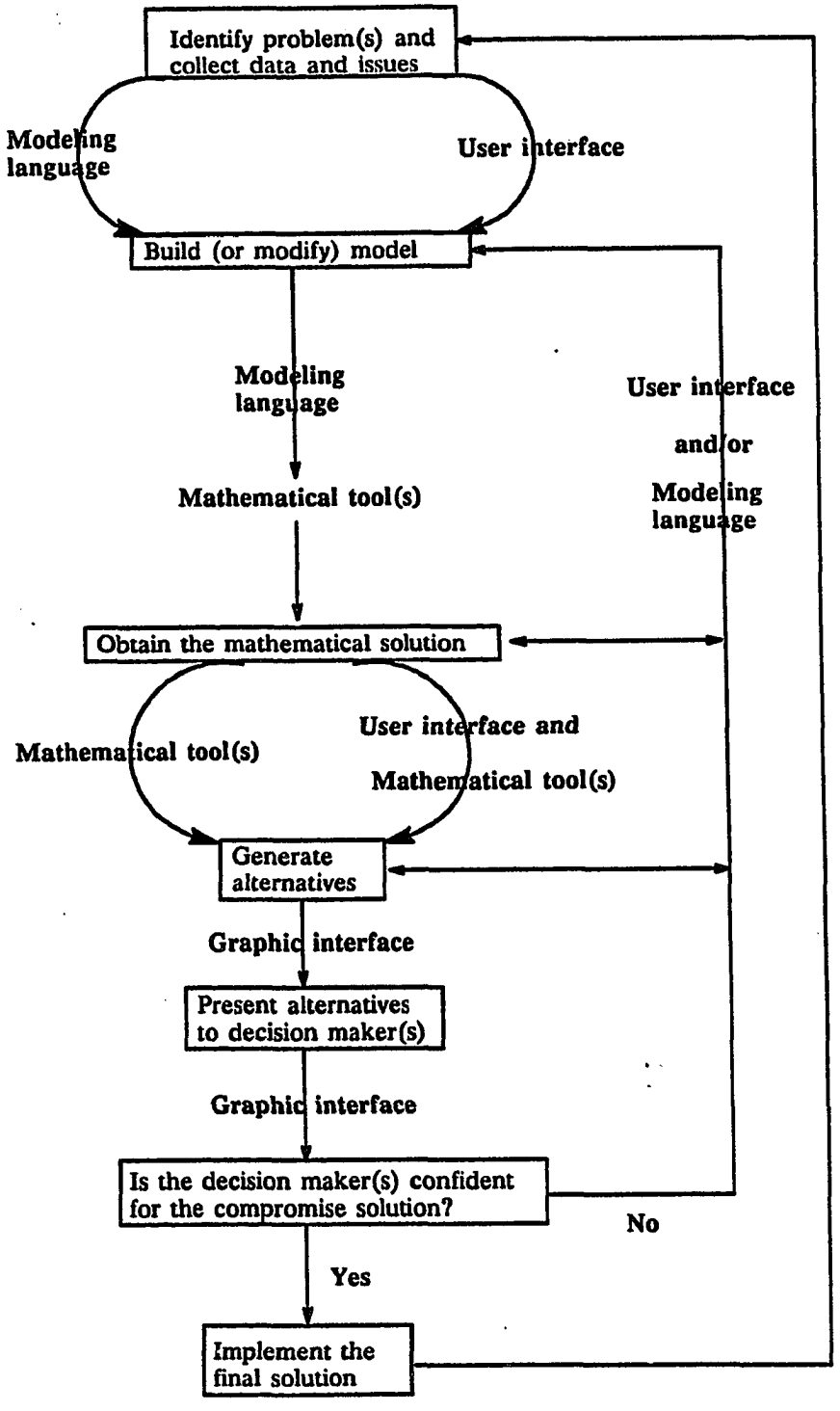


Figure 3.2. Incorporating New Features of Computer Aided Systems into the Working Process for Decision Making

3.3. Graphic Interface

The purpose of the graphic interfaces designed in the two prototypes is mainly for presentation (stage E) and to provide an interface to a decision maker (stage F).

Attributes, Configurations, or Results

Attributes such as cost are usually important for a decision making problem. The display of attributes, however, may be difficult. For example, the display of cost curves may be difficult because the curves are exponential and might cover a wide range of design parameter values. Although semi-log plots can be used to represent cost curves, to perceive the approximate value of cost from the plot is difficult. A semi-log curve gives only the shape of curve and does not provide much help for examining a cost region. In this research, the cost curves are presented in normal scale so that the approximate cost associated with a design parameter value can be easily seen (see Chapter 4). Similarly, the presentation of a problem configuration, such as a wastewater process scheme or groundwater domain, and results such as a set of numerical output data, also require appropriate presentations. The goal of using graphic interfaces for attributes, configurations, and results is to present them in a manageable manner for instant evaluations.

Alternatives

As mentioned, comparison is a significant activity in a decision making process. Usually, numerous comparisons must be completed before a decision can be made. Although tabular data provide detailed information about an alternative, to compare alternatives using tabular data is likely to be difficult and time-consuming. For comparisons, the differences among alternatives are more important than the exact numerical values, and differences can be effectively presented graphically. Even though it may be desirable to compare numerous alternatives, a human is capable of handling only a small number of alternatives at a time. As a compromise between the number of alternatives and computer display resolution, it was decided to show four alternatives at a time for comparisons in this research (see Chapter 5). The graphic representation of alternatives and approaches designed to keep the number of alternatives manageable for each comparison are expected to increase greatly the efficiency of selecting a final alternative in a decision-making process.

Noninferior Set

For a problem with two or three objectives, it is very desirable to present the noninferior set graphically instead of tabularly. Figure 3.3 shows how to improve the presentation of a noninferior set in comparison to using Tables 3.1 and Table 3.2,. The graphic expression of the noninferior set provides not only a better presentation, but also a means to an interactive interface to aid decision making. For example, if an analyst wants to explore an alternative noninferior solution, then he can pick a point on a tradeoff curve by moving a mouse cursor to the desired point and pressing the mouse button. The desired point is then generated for comparison purpose. This interactive capacity is potentially very useful in a decision making process.

In the research, a program has been developed to display a 2-D tradeoff curve which also provides an interface for examining noninferior solutions (see Chapter 5). A 3-D noninferior set can be presented, but it is not suitable for interactive uses. The 3-D presentation can be improved if a good 3-D software package can be linked with the prototypes and a high resolution color monitor is available.

3.4. User Interface

The designs of the user interfaces of the current prototypes are intended to make data entry and function choices as simple as possible. It is assumed that the users would be engineers but not necessarily computer experts.

Characteristics

The user interface is a key component in a good computer aided system. An analyst, modeler, or designer may be reluctant to use a system that is computationally efficient but that has a poor user interface. Some characteristics of a good user interface are: (1) simplicity of learning; (2) minimal possibility of making mistakes; (3) flexibility in modification of models and data; (4) efficiency of data organization or solution management; and (5) clarity of instructive feedback (stage A & D).

Menu Selections

Requiring the user to type directly the number or letter of an entry in a list requires that the user is familiar with the keyboard layout and generally takes longer. A menu-driven interface with a pointing device (mouse) to make selections is developed. Such an interface is expected to require less learning time.

Plane 1-5-6 and 2-5-10 are perpendicular to the boundary plane, Z3=0.
Points 4, 5, 6, 7, 9, and 10 lie on the same plane.
Points 3, 4, 8, and 9 lie on the same plane.

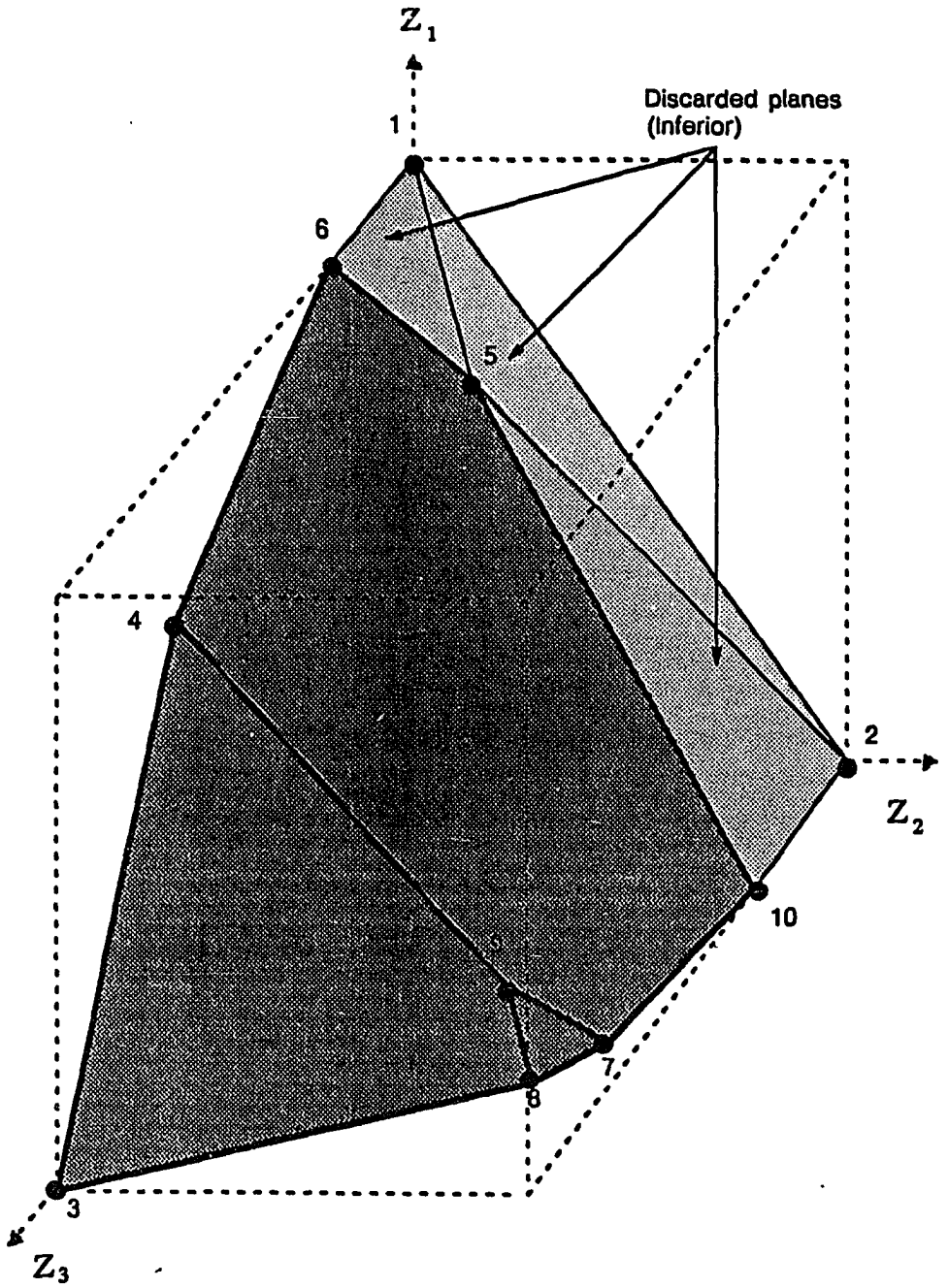


Figure 3.3. Noninferior Points and Planes for the Sample Problem

Modification

For an analyst to construct good alternatives for a complex problem, it is often desirable to use a variety of mathematical tools and to examine a significant amount of data. For obtaining a good solution, it is necessary to examine a significant number of alternatives and to choose appropriate issues to be modeled. How to select and model these issues is an important issue. The model may be frequently changed before a decision is made. Furthermore, several known but unmodeled attributes may be significant when alternative solutions are examined, and new attributes may be discovered during the analysis. Modification is a task which may occupy most of the time of an analyst or designer for building up good alternatives or solutions. To simplify the modification process and to increase the productivity of an analyst or designer, several functions requiring no more than pressing a mouse button were developed to make this task as simple as possible.

Graphic Oriented Object In Exploratory Design

Another significant part of this research is to explore and compare several different ways to help the designer in doing exploratory design by using graphic oriented objects. In this research, several graphic objects are used to represent physical objects, e.g. unit processes and pumping wells, and abstract objects, e.g. check points and boundary condition locations. By manipulating these objects by simply moving a mouse and pushing a button, an analyst can easily explore a good design.

Solution (Data) Management

As mentioned above, many comparisons among alternatives may be needed to make a good decision. A good solution management system is thus needed to keep all alternatives. The system should be efficient for retrieving, deleting, replacing, and grouping alternatives and, of course, easy to use. A solution management system extracted from the earlier work by Brill et al. [1989] was used in the research (see Chapter 6).

Feedback (checking, warning, and message)

An important component of a user-friendly system is an intelligent feedback system that can respond to all possible actions (valid or invalid) selected by a user. For example, if the user makes a valid action, then a message should appear to explain what has been accomplished or changed as a result of

the action. If an invalid action is made, then a message should appear to explain why it is invalid and suggest one or several valid actions which might be appropriate substitutes. The responses need not be in text but they should be unambiguous. In this research, a feedback system which includes error messages, warning messages, help messages, a beeper, valid action response messages, and graphical displays was developed. However, since these computer aided systems are prototypes, some inappropriate feedbacks likely still exist in the systems and will be identified only through additional testing and development of the systems.

3.5.Summary

By using the new modeling language (see Chapter 4), a mathematical model can be easily set up. With the graphic display and user interface, the working complexities of presentation, modification, and working with software packages can be significantly reduced. By combining these techniques, general optimization techniques (e.g. those used by XMP), and mathematical techniques, two prototype interactive computer aided systems were developed for two environmental decision making problems, WTPD and GRM. These systems are demonstrated in Chapters 5 and 6. The systems not only provide an analyst or decision maker with information, but also provide good interfaces for analyzing and modifying a model. Although the two developed systems are prototypes and are for two specific problems, they should illustrate concepts of developing computer aided systems and provide a way to identify and address some of the issues which are important for other engineering decision making problems.

CHAPTER 4

VECTOR METHOD, MODELING LANGUAGE, AND MGA

One new multiobjective technique (Vector Method), a new modeling language, and two new MGA methods were developed during the course of the research. They are described in Section 4.1, 4.2, and 4.3, respectively.

4.1. Vector Method

The Vector Method for generating the complete noninferior surface of a linear multiobjective problem in a bounded space is described in this section. The method overcomes complexities that may be encountered in using the Noninferior Set Estimation (NISE) method [Cohon, 1978]. The NISE method has been demonstrated successfully for a two objective problem. For a three objective problem, however, several complexities exist in using the NISE method. As a result, the NISE method may miss part of the noninferior set and/or may be computationally inefficient. The Vector Method is intended to overcome these difficulties for a three or more objective problem. Within a bounded space, the method does not miss any part of the noninferior surface or any of the extreme points. Also, with the proposed updating procedure, the method does not generate a noninferior point more than once. This procedure eliminates the redundant computations which may occur in using the NISE method.

In the following subsections, the NISE method is briefly described, and discussions of the complexities of using the NISE method for a three objective problem are provided. A geometric proof of the completeness of the noninferior surface obtained using the Vector Method is described based on convexity. Then, the details of the Vector Method are provided with a demonstration for a simple three objective problem. The procedure of the method for n-dimensional problems is also discussed, although a proof is so far not available. Finally, algorithms for checking inferiority are discussed.

4.1.1. Brief Description Of The NISE Method

The details of the NISE method are given by Cohon [1978] and Balachandran et al. [1985]. A brief description is provided below with clarification of some steps of the method. Objectives are assumed to be maximized.

A Two-Dimensional Case

For a two objective problem, the NISE method starts by optimizing each objective separately to obtain two noninferior points, such as A and B in Figure 4.1. The line AB is called the primary line and used as the basis line in the first iteration. Then, by pushing out this first basis line in the normal direction, as shown by vector n in Figure 4.1, the farthest reachable noninferior point in that direction is located, such as point C in Figure 4.1. Two basis lines, AC and BC, are then generated for the following iterations. A tolerance is defined in terms of the perpendicular distance between the farthest possible noninferior point and the basis line to be used. For example, point D is the farthest possible reachable noninferior point that can be obtained by using the basis line AC. The two end points, A and C, were generated by using the basis lines AD and AB respectively. The perpendicular distance between D and the basis line AC can serve as a tolerance. If the tolerance associated with the basis line AC or CB exceeds the maximum allowable tolerance, which is pre-set by the analyst, the procedure is continued by pushing the basis line out in the normal direction to obtain other noninferior points. The procedure is terminated when all tolerances are acceptable.

A Three-Dimensional Case

For a three objective problem, as in the two-dimensional case, the NISE method starts by optimizing each objective separately to obtain three noninferior points, such as A, B and C in Figure 4.2. The plane ABC is called the primary plane and used as the basis plane in the first iteration. Then, by pushing out this first basis plane in the normal direction, as shown by vector n in Figure 4.2, the farthest reachable noninferior point is located, such as point D in Figure 4.2. Three basis planes, DBC, DCA, and DAB, are then generated for the following iterations. Similar to the two-dimensional case, a tolerance is used to determine whether the procedure should be continued or not. The tolerance is defined in terms of the perpendicular distance between the basis plane and the intersection of the three planes which were used as basis planes to locate the three vertices which form the basis plane. If the tolerance associated with any basis plane exceeds the maximum allowable tolerance, which is pre-set by the analyst, then the procedure is continued by pushing the basis plane out in the normal direction to obtain other noninferior points. The procedure is terminated when all tolerances are acceptable.The general algorithm for the NISE method for a 3D case is listed below.

Algorithm NISE-3D

A stack is used to keep unexplored basis planes.
A noninferior point data set is used to store information for all noninferior extreme points.

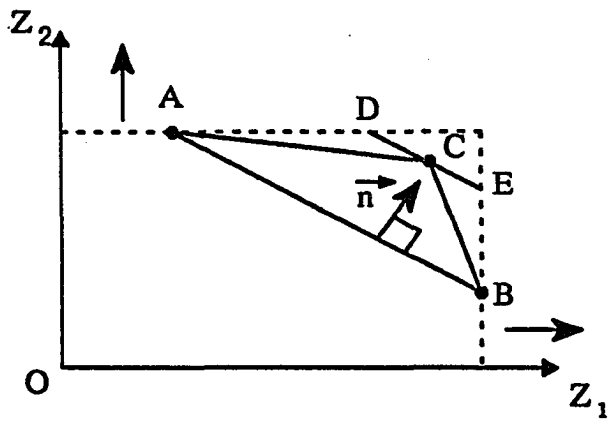


Figure 4.1 NISE – a 2D case

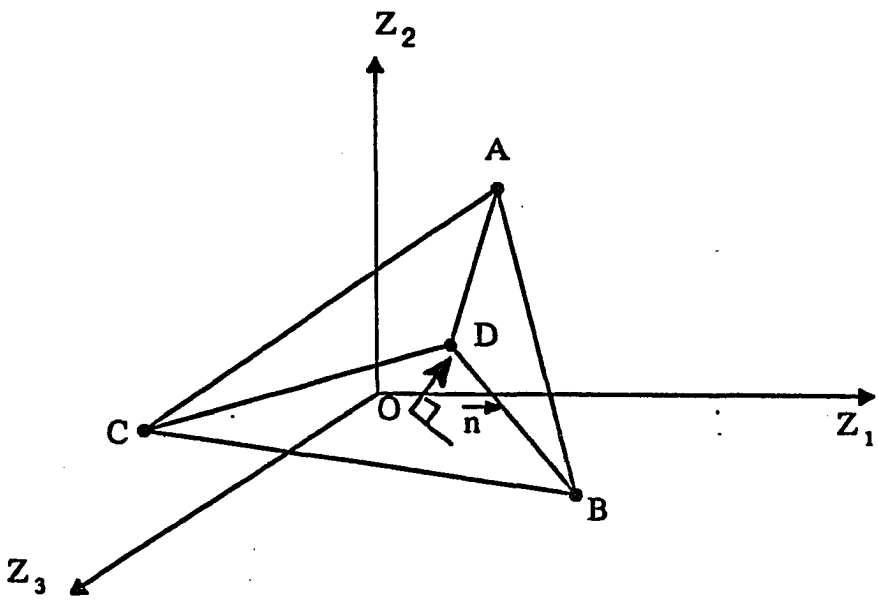


Figure 4.2 NISE – a 3D case

Initial Steps

1. set the tolerance and empty the unexplored plane stack.
2. maximize each objective separately to obtain three noninferior points, as A, B, and C shown in Figure 4.2.
3. compute the set of weights of objectives (or the normal vector as n shown in Figure 4.2). (Let the weights be w1, w2, and w3.)
4. add plane A-B-C with the weight set into the unexplored basis plane stack.
5. save the information about generated points A, B, and C into the noninferior point data set.

Main Steps

main_loop:

if (the unexplored basis plane stack is not empty) then

1. pop a basis plane with the weight set from the unexplored basis plane stack.
2. Max $w_1 Z_1 + w_2 Z_2 + w_3 Z_3$
   S.T. $X \in Fd$,
   where
   X is the vector of decision variables; and
   Fd is feasible domain.
3. if (a new point is obtained from step 2) then
   1. save the point with the basis plane used to generate the point into noninferior point data set.
   2. generate three basis planes, as planes DBC, DAB, and DAC shown in Figure 4.2.
   3. for each generated basis plane, compute its tolerance,
      if (tolerance is not acceptable) then
      1. compute the weights.
      2. add the basis plane with the weight set into the unexplored basis plane stack.
      end if
   end if
4. Go to main_loop.

else stop the procedure.

4.1.2. Complexities In Using The NISE Method

Two complexities exist in using the NISE method for a three-dimensional problem. The first complexity occurs if there are noninferior points beside or below a basis plane. Noninferior points beside or below a basis plane may not be located using the NISE method. For example, in Figure 4.3 the noninferior surface is the shaded plane with six noninferior extreme points (D, E, F, G, H, and I). In using the NISE method, no matter which three noninferior points are used to form the primary (or basis) plane, no new point can be located because no noninferior extreme point lies above the plane. Thus, the generating procedure would terminate and the other three noninferior points would be missed.

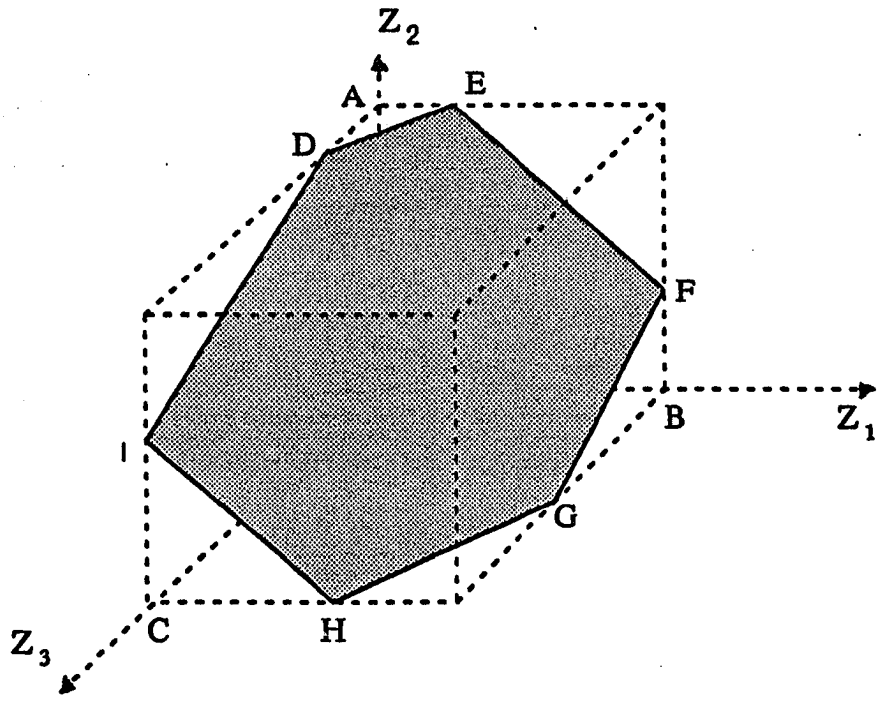


Figure 4.3 A Plate Noninferior Plane

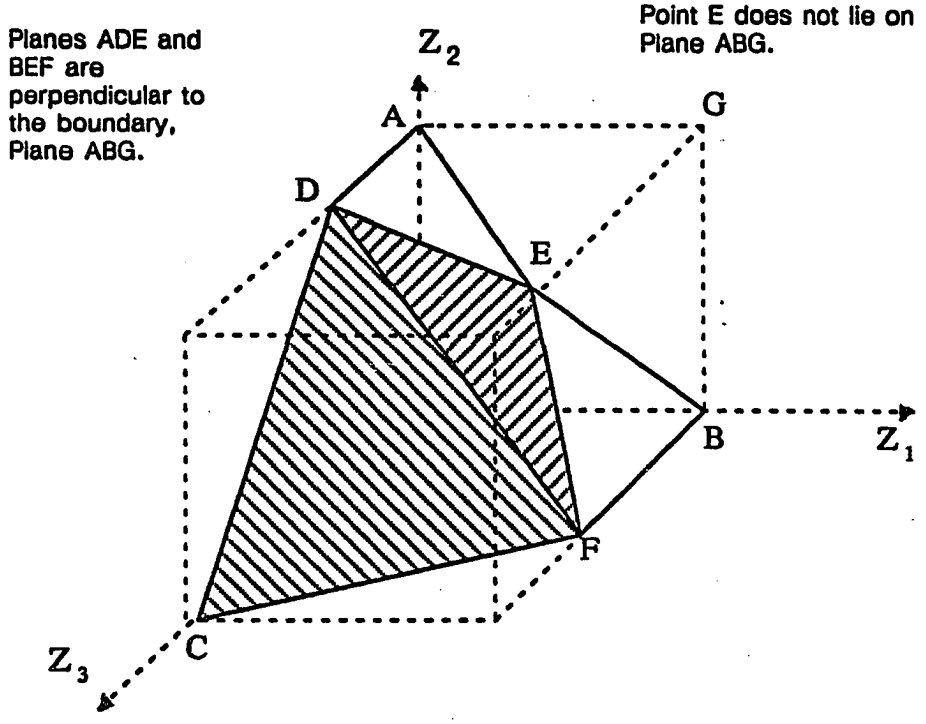


Figure 4.4 Two Noninferior Planes

In Figure 4.4, the noninferior surface is shown by two adjacent shaded planes. In using the NISE method, the plane CDF is the primary plane used at the first iteration. Since no noninferior point lies above this plane, the generating procedure would be terminated. There is, however, another noninferior extreme point, E, at the side and below the primary plane. Cohon [1978] suggested a boundary projection approach to check for the existence of any noninferior extreme point at the side of the basis plane. Many such points may exist, however, and many computations may be required since it is necessary to check beside every plane on the edge of the noninferior set.

The second complexity is that if there are other noninferior extreme points which lie outside-and-above the basis plane, then some redundant basis planes may be generated. In Figure 4.5, the noninferior surface is shown by three shaded planes. Plane CDB would be the primary plane used. Although another extreme point, F, can be located above this primary plane, one of the three generated basis planes for following iterations would be redundant, i.e. the plane BDF. From this redundant plane, the known noninferior extreme point, C, would be generated again.

4.1.3. The Vector Method

The Idea of the Vector Method and A Geometric Proof for a 3D case

The idea of the Vector Method is to start a procedure for generating the exact noninferior set in a bounded space by using a minimal approximate convex surface where no noninferior point can lie below this surface, or plane for a 3D case. Then, in each subsequent iteration a search vector, which points away from the origin and out of the current approximate noninferior surface, is used to locate a new noninferior extreme point. The approximate surface is thus extended and moved closer and closer to the exact noninferior surface. If the procedure is continued until no new point is located, then the final convex surface should be the exact noninferior surface in a bounded space. This general description of the Vector Method also provides a geometric proof of the completeness of the generated noninferior set based on the convexity of a linear space.

The main algorithm of the Vector Method for a 3D problem is shown below. Several important concepts used to determine the minimal approximate noninferior plane, to obtain search directions, to maintain convexity of the approximate surface, and to carry out other details of the method are described as follows.

Plane ADEF is perpendicular to the boundary plane, Plane ABG.

Points A, E, G, and B lie on the same plane.

Line DH lies on the plane ADC and is the intersection of planes BDF and ADC.

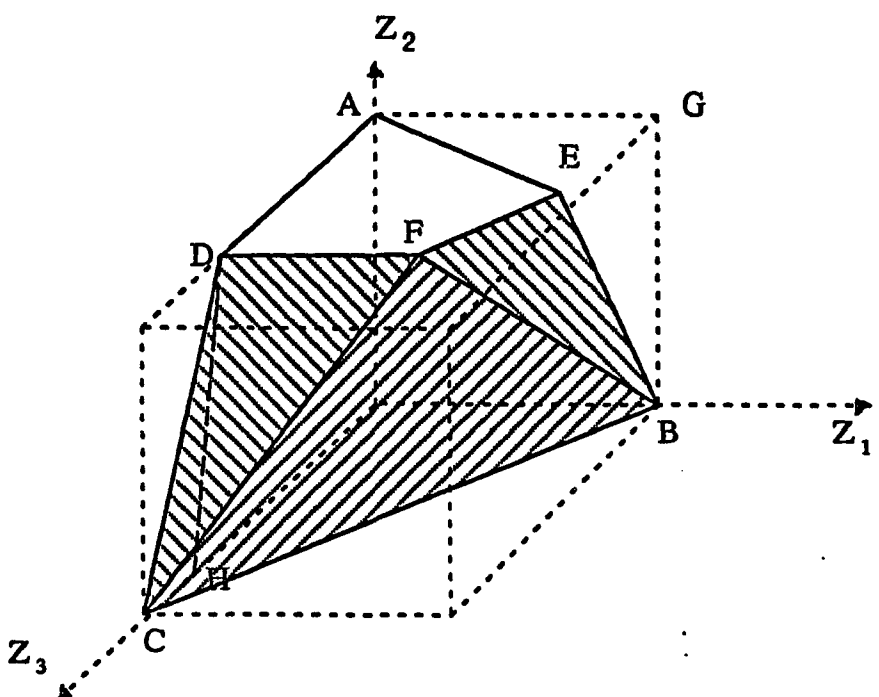


Figure 4.5 Three Noninferior Planes

Main Algorithm

A stack is used to keep unexplored basis planes.
A noninferior point data set is used to store information for all noninferior extreme points.
Initial Steps

1. set lower bounds for objectives and the tolerance, and empty the unexplored basis plane stack.
2. use unit vector (1, 0, 0) to maximize Z_1 to obtain (MaxZ_1, xx, xx)
   (0, 1, 0) to maximize Z_2 to obtain (xx, MaxZ_2, xx)
   (0, 0, 1) to maximize Z_3 to obtain (xx, xx, MaxZ_3)
   where xx's indicate whatever values are obtained for related objectives.
3. compute the unit vector and the perpendicular distance to the origin for the plane ABC, the minimal primary plane, (see equation 4–4 for computing the distance) where
   Point A = (MaxZ_1, 0, 0),
   Point B = (0, MaxZ_2, 0), and
   Point C = (0, 0, MaxZ_3). (Replace all xx's by zero.)
4. add the plane ABC with its unit normal vector (direction) and distance (from the origin to the plane) into the unexplored basis plane stack.

Main Steps

main_loop:
if (the unexplored basis plane stack is not empty) then
1. pop a basis plane with the unit vector, ($\cos\alpha$ $\cos\beta$, $\cos\gamma$) and the distance (p_{old} in equation 4–3) from the unexplored basis plane stack.
2. Max $p_{new} = \cos\alpha\ Z_1 + \cos\beta\ Z_2 + \cos\gamma\ Z_3$
   S.T. $X \in Fd$.
3. if (a new point is obtained) then
   (the objective value, p_{new}, is the vertical distance between the origin and the plane which includes the new point and is parallel to the basis plane)
   1. generate new planes (excluding redundant planes)
      (see Algorithm *NEWPLANE*)
   2. update the unexplored basis plane stack, if needed.
      (see Algorithm *UPDATE*)
   3. if ($(p_{new}-p_{old}) >$ tolerance)
      then add the generated planes into the unexplored plane stack
      else all generated planes are approximations for part of noninferior surface
      end if
   else {no point reached}
   if (any component of the unit vector of the basis plane is negative)
   then discard the basis plane (add it to the discarded plane data set).
   else the basis plane is an exact noninferior plane.
   end if
   end if
4. go to main_loop.

else stop the procedure.

Algorithm *NEWPLANE*

if (pnew > pold) then

1. compute the coordinate of the projection of the new point on the basis plane. Let the new point be (Zn-1, Zn-2, Zn-3), then the projection point, (Zp-1, Zp-2, Zp-3), is simply computed as (Zp-1,Zp-2,Zp-3) = (Zp-1,Zp-2,Zp-3)- (pnew-pold)* (cosα, cosβ, cosγ).
2. determine the location of the projection and delete any redundant basis planes (see subsection *Checking Redundant Plane and An Updating Procedure*).
3. add generated basis planes (after excluding any redundant ones) into the unexplored basis plane stack.

end if

Algorithm *UPDATE*

The generated point and the current basis plane are known before applying this algorithm.

Initial Step

add the current basis plane into the updating plane stack.

Main Steps

Update_loop

if (the updating plane stack is not empty) then

1. pop an updating plane from the updating plane stack
2. for each adjacent basis plane of the current updating plane, check
   if (the generated point is above the adjacent basis plane) then
   1. add the adjacent basis plane into the updating plane stack.
   2. delete the current adjacent plane from the unexplored basis plane stack.
   3. use algorithm *NEWPLANE* to generate new planes and the projection of the generated point on the adjacent basis plane (see discussion in subsection *Checking Redundant Plane and An Updating Procedure*).
   4. if the generated point is out-and-up relative to the adjacent basis plane, then delete all basis planes which include the side that is closest to the projection point of the adjacent basis plane from the unexplored basis plane stack.
      end if
   end if
3. go to Update_loop

else stop the procedure.

Search Box and Minimal Primary Plane

In using the Vector Method, a lower bound is set for each objective (assuming maximization) to limit the search space. Although these lower bounds are required for this method, there is no theoretical restriction on the values of these bounds. They can be set according to an analyst's judgment. The upper bound of each objective in search space is obtained from maximizing each objective respectively. The

search space can be viewed as the box shown in Figure 4.6. In the main algorithm, after the second step of *Initial Steps*, the search box is formed.

As in the case of the NISE method, this method starts from a primary plane, although it is determined in a significantly different way. The primary plane used is formed by three vertices of the search box, as illustrated by the plane ABC in Figure 4.7 and described in Step 3 of the *Main Algorithm* in the *Initial Steps* section. The three vertices may be inferior and sometimes infeasible rather than always noninferior and feasible as in the case of the NISE method. The major reason to use the plane ABC as the primary plane is to overcome the complexity of probably missing noninferior extreme points beside a basis plane in using the NISE method. Because lines AB, BC, and CA lie on the boundary of the box, any point below the plane ABC is dominated by one of the points on the triangle plane ABC. Thus, no noninferior extreme points can lie below this plane. This plane is therefore called the minimal primary plane. Furthermore, it is impossible to have any noninferior extreme points other than points A, B, and C that lie on this plane, and the minimal primary plane is always unique for a three objective problem.

By starting from this minimal primary plane, the first complexity in using the NISE method can be overcome. It is, however, still possible to generate redundant basis planes if a noninferior point exists outside and above a basis plane. This complexity is easy to overcome by detecting redundant planes and deleting them. The procedure to check out these redundant planes is discussed mathematically and shown graphically in the next subsection.

Checking Redundant Plane and An Updating Procedure

Figure 4.8 shows a noninferior surface formed by two adjacent noninferior planes, ABC and ACD. Line BD is inside the noninferior space. Assume the plane ABD is the basis plane used in the current iteration. The noninferior extreme point C is located next. Three new basis planes, ACD, DCB, and ACB, are generated. It is easy to see that point A would be located again by using the basis plane DCB which is redundant. For checking out this redundant plane, the location of the projection (Point E in Figure 4.9) of point C (the most recently generated noninferior extreme point) on the plane ABD (the plane used to generate the most recent noninferior extreme point) is computed. By expanding three sides of the triangle ABD, the basis plane is divided into seven regions (see Figure 4.9). Because of the convexity of the noninferior surface, the projection point cannot lie in region 4, 5, or 6. To figure out which of the other four regions includes the projection point, the following relationship among vertices of the basis plane and the projection point is used.

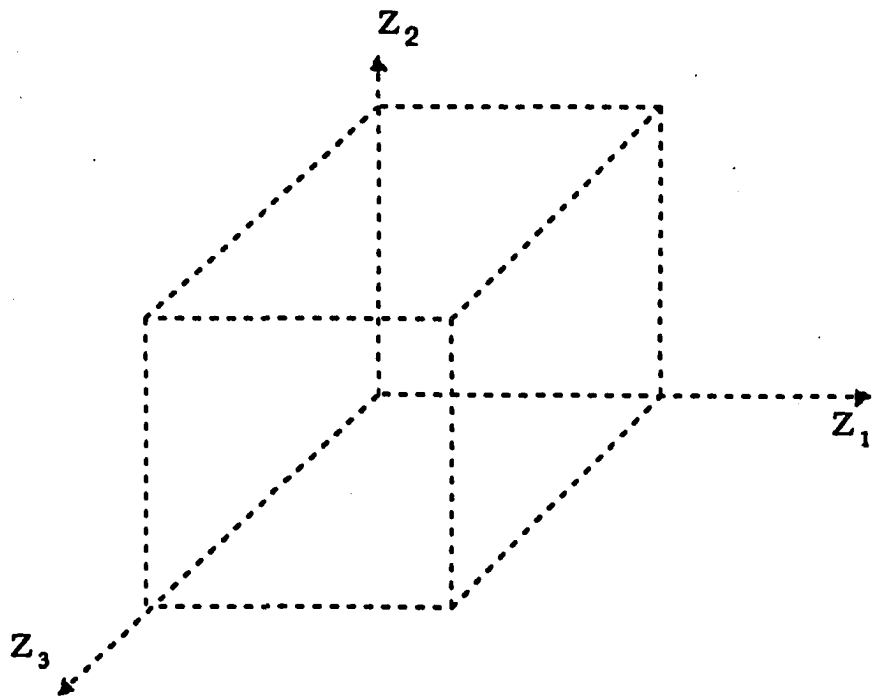


Figure 4.6 The Search Box

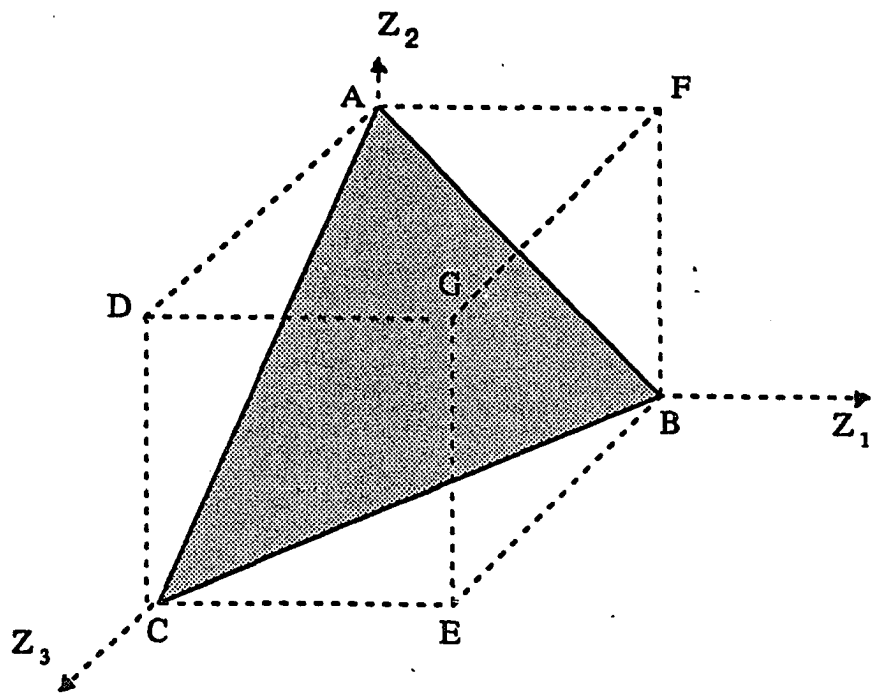


Figure 4.7 The Minimal Primary Plane

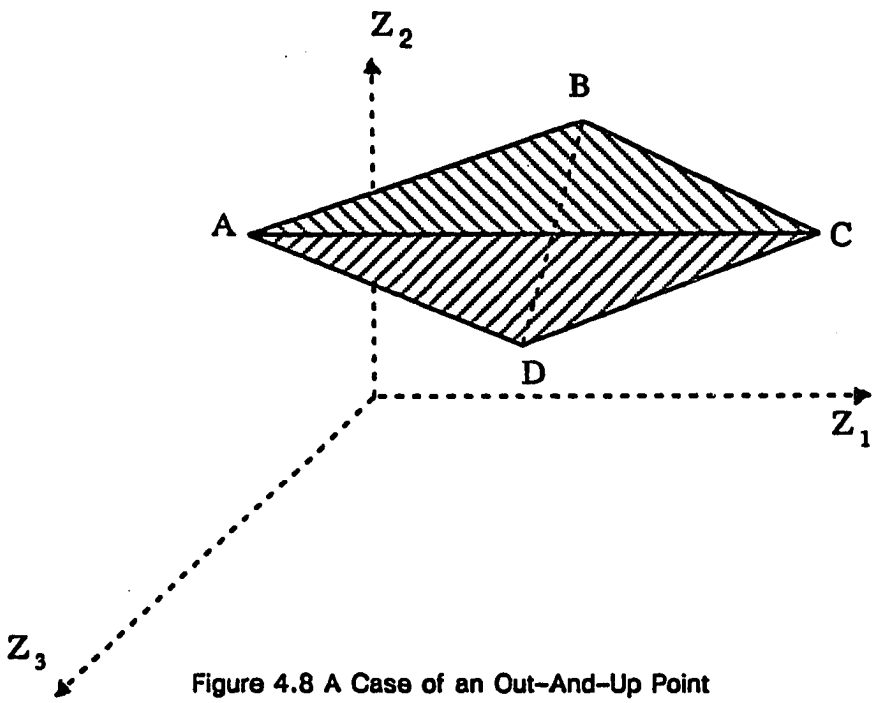


Figure 4.8 A Case of an Out-And-Up Point

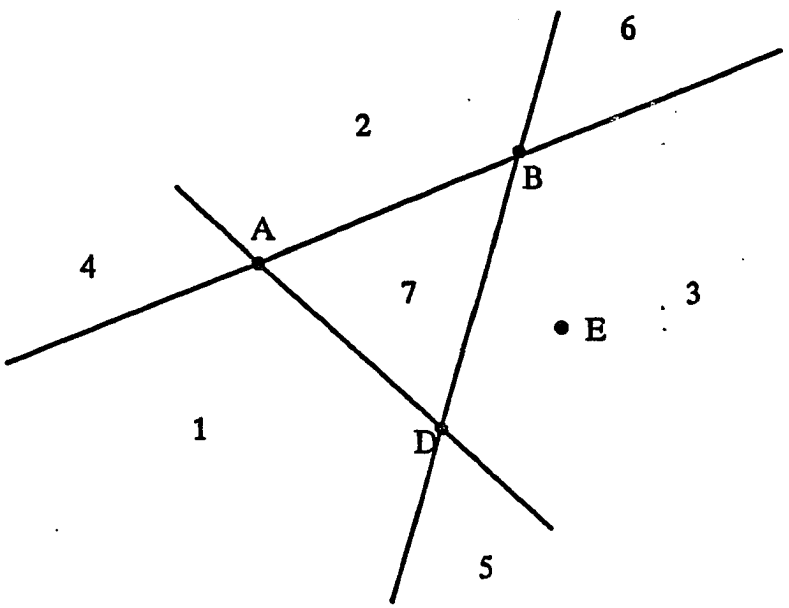


Figure 4.9 A Basis Plane with the Projection Point of the Point Generated Using the Basis Plane

$$\overrightarrow{AE} = \theta_1 \overrightarrow{AB} + \theta_2 \overrightarrow{AD} \qquad (4\text{–}1)$$

First, the above relationship for θ1 and θ2 is solved. Then, by using values of θ1 and θ2, the location of the projection point can be determined:

Point E is in:

region 1, if θ1 <= 0 and θ2 >= 0; and plane ADC is a redundant plane;

region 2, if θ2 <= 0 and θ1 >= 0; and plane ABC is a redundant plane;

region 3, if θ1 >= 0, θ2 >= 0, and θ1+θ2 >= 1; and plane DCB is a redundant plane; and

region 7, if θ1 > 0, θ2 > 0, and θ1+θ2 < 1; and there is no redundant plane.

(4–2)

If a point is obtained outside and above a basis plane (the projection point lies in region 1, 2, or 3), then the current approximate surface may be concave. To maintain the convexity of the approximate surface and to avoid using a redundant basis plane later, an updating procedure (see also Algorithm *UPDATE*) is required as follows. Those basis planes which may be required to be updated are the basis planes adjacent to the current basis plane. If the generated point lies above any adjacent basis plane, then the adjacent basis plane is needed to be updated. To determine if the point is above an adjacent basis plane or not, the perpendicular distances* between the generated point and each adjacent plane can be used as follows:

$$\text{distance} = p_{new} - p_{old} = \cos\alpha\, Z_{n-1} + \cos\beta\, Z_{n-2} + \cos\gamma\, Z_{n-3} - p_{old}, \qquad (4\text{–}3)$$

where

p_{new} is the distance between the origin and the generated point in the normal direction to the adjacent plane;

p_{old} is the distance between the origin and the adjacent plane in the normal direction;

$\cos\alpha\, Z_1 + \cos\beta\, Z_2 + \cos\gamma\, Z_3 = p_{old}$ determines the adjacent plane; and

$(\cos\alpha, \cos\beta, \cos\gamma)$ is the unit normal vector of the adjacent plane.

If the distance is positive, then the generated point lies above the adjacent basis plane, which must be updated to maintain the convexity of the approximate surface. The update can be done by the same procedure for generating a new basis plane (see Algorithm *NEWPLANE*) by treating the generated point as the point generated from using the adjacent plane as the basis plane. This procedure is continued until no adjacent plane needs to be updated (see Algorithm *UPDATE*).

---

* the distance in this context is the directed distance along the normal vector, $(\cos\alpha, \cos\beta, \cos\gamma)$; it is negative if opposed to the normal vector.

Vector, Distance, Tolerance, and Nondominating Plane

Although the weights for objectives used in each iteration of the Vector Method are similar to those generated using the NISE method, a theoretically different procedure for generating weights is used. For a three dimensional plane, it can be determined by either one of two general equations listed below.

$A Z_1 + B Z_2 + C Z_3 = D$, or

$\cos\alpha\ Z_1 + \cos\beta\ Z_2 + \cos\gamma\ Z_3 = p$ (4–4)

where

A, B, C, and D are constants;

Z_1, Z_2, and Z_3 are objectives;

$(\cos\alpha, \cos\beta, \cos\gamma)$ is an unit vector which is perpendicular to the plane;

p is the perpendicular distance between the plane and the origin.

The unit vector described above is called the unit normal vector to the plane. The direction of the normal vector points away the origin and out of the basis plane, and it can be easily determined by letting p and D be positive.

The first general equation can be normalized as

$a Z_1 + b Z_2 + c Z_3 = 1$ (4–5)

where $a = A/D$, $b = B/D$, and $c = C/D$.

For each iteration, there are three known points to determine the basis plane: $(Z_{1-1}, Z_{1-2}, Z_{1-3})$, $(Z_{2-1}, Z_{2-2}, Z_{2-3})$, and $(Z_{3-1}, Z_{3-2}, Z_{3-3})$. Then, based on equation 4–5, a matrix system can be formed as:

$$\begin{bmatrix} Z_{1-1} & Z_{1-2} & Z_{1-3} \\ Z_{2-1} & Z_{2-2} & Z_{2-3} \\ Z_{3-1} & Z_{3-2} & Z_{3-3} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

After solving this matrix system for values of a, b, and c *, the unit normal vector can be determined as follows.

Let $t = SQRT(a*a+b*b+c*c)$,

then $(\cos\alpha, \cos\beta, \cos\gamma) = (a/t, b/t, c/t)$, and $p = 1/t$.

*The vector, (a, b, c) can also be computed from the cross product of two different vectors on the plane determined by the three known points.

By using the unit normal vector, the perpendicular distance between the parallel plane on which the generated noninferior point lies and the origin along the normal direction can be easily read from the objective value (see the main algorithm). The distance is useful in determining the tolerance and the projection of the generated point on the basis plane.

The tolerance (see Balachandran, et al. [1985]) used by the NISE method is defined in terms of the perpendicular distance from the current basis plane to the intersection of the three basis planes used to locate the three vertices of current basis plane. A simpler tolerance, however, is used for the Vector Method. Since we push the basis plane out as far as possible at each iteration, the perpendicular distance between the basis plane and a new parallel plane on which the generated point lies can be used as the tolerance, because no noninferior point can lie beyond the latter plane. This tolerance can be easily computed because the distances of the current basis plane and the new plane to the origin are known in each iteration. Although this tolerance is simpler, the one used by the NISE method could also be used.

A nondominating plane is defined as a plane on which no point dominates any other point. A nondominating plane is a noninferior plane if no other feasible point dominates any of the points on the plane. A nondominating plane can be easily identified by using the unit normal vector described. Assume that two arbitrary points on a plane are $(Z_{1,1}, Z_{1,2}, Z_{1,3})$ and $(Z_{2,1}, Z_{2,2}, Z_{2,3})$, and the unit vector normal to the plane, which is directed away from the origin, is $(\cos\alpha, \cos\beta, \cos\gamma)$. Because the unit vector is perpendicular to any vector on the plane, the inner product of the unit vector and the vector determined by the two points should be equal to zero, i.e.

$$\cos\alpha\,(Z_{2,1}-Z_{1,1}) + \cos\beta\,(Z_{2,2}-Z_{1,2}) + \cos\gamma\,(Z_{2,3}-Z_{1,3}) = 0$$

If $\cos\alpha$, $\cos\beta$, and $\cos\gamma$ are all positive (or all negative), then no two points on the plane can meet the relationship shown above with one point dominating the other. On the other hand, if any component of the unit vector is negative while at least one other component is positive, it is always possible to find some other points to dominate any point on the plane except for those points on the boundary. Thus, a nondominating plane can be defined as a plane whose unit normal vector has positive components. Although a plane is still a nondominating plane if all components of the unit vector are negative, the direction of the unit vector is invalid in generating a noninferior extreme point and should not be used. Since the Vector Method may start from a plane formed by several inferior or infeasible extreme points, the final approximate surface may include some inferior planes which are close to a boundary plane of the search box. These inferior planes should be discarded from the final noninferior surface by checking the signs of all components of unit normal vectors.

4.1.4. A Sample Problem

A sample three objective problem which includes five variables and five constraints is used to demonstrate the use of the Vector Method. The noninferior set of this problem has been determined by checking all extreme points in the feasible space. This sample problem was tested to show the procedure graphically. There is no special meaning associated with any objective or constraint. A FORTRAN program was developed using the optimization package XMP [Marsten, 1984] to implement the Vector Method. A graphics package, GM 2-D Graphics Metafile [1985], designed for an Apollo™ Workstation was used to show the steps of the procedure on a high resolution monitor.

The objective functions and constraint set of this problem are listed below.

Max 1.20 X1 + 2.60 X2 - 1.10 X4 + 1.80 X5 + 130
Max 1.50 X1 - 1.50 X2 + 1.00 X3 + 1.60 X4 + 2.20 X5 + 250
Max -0.70 X1 + 1.40 X2 - 1.20 X3 - 1.60 X5 + 115
(Assume that all objectives are positive.)

S.T.

1.0 X1 + 2.0 X2 + 3.0 X3 + 4.0 X4 + 5.0 X5 <= 120.0
3.2 X1 - 3.3 X2 + 1.8 X3 - 1.0 X4 + 4.0 X5 <= 60.0
-1.2 X1 + 4.3 X2 - 3.0 X3 + 4.0 X4 + 3.0 X5 <= 400.0
2.0 X1 + 3.4 X2 + 2.4 X3 + 1.2 X4 - 2.0 X5 <= 40.0
-3.0 X1 + 5.0 X2 + 2.9 X3 + 1.0 X4 + 1.0 X5 <= 300.0
-405 <= Xj <= 400 for j= 1, 2, 3, 4, and 5

For ease of implementation of the Vector Method and computational efficiency (see the end of this subsection), the formulation is reconstructed as follows.

Max ($\cos\alpha$ Z_1 + $\cos\beta$ Z_2 + $\cos\gamma$ Z_3)

S.T.

Z_1= 1.20 X1 + 2.60 X2 - 1.10 X4 + 1.80 X5 + 130
Z_2= 1.50 X1 - 1.50 X2 + 1.00 X3 + 1.60 X4 + 2.20 X5 + 250
Z_3= -0.70 X1 + 1.40 X2 - 1.20 X3 - 1.60 X5 + 115
Z_i >= 0 for i= 1, 2, and 3
Original constraint set;

Figure 4.10 shows the final noninferior surface obtained using the Vector Method. All noninferior points and planes are listed in Tables 4.1 and 4.2, respectively. In the literature (e.g. Balachandran et al. [1985]), the noninferior set of a multiobjective problem is often presented by a set of noninferior points. From this set, however, it is difficult to visualize the complete noninferior set correctly. For example, from Table 4.1 it is difficult to perceive the exact noninferior surface. If we connect points 3, 6, and 10, the

Plane 1-5-6 and 2-5-10 are perpendicular to the boundary plane, Z3=0.
Points 4, 5, 6, 7, 9, and 10 lie on the same plane.
Points 3, 4, 8, and 9 lie on the same plane.

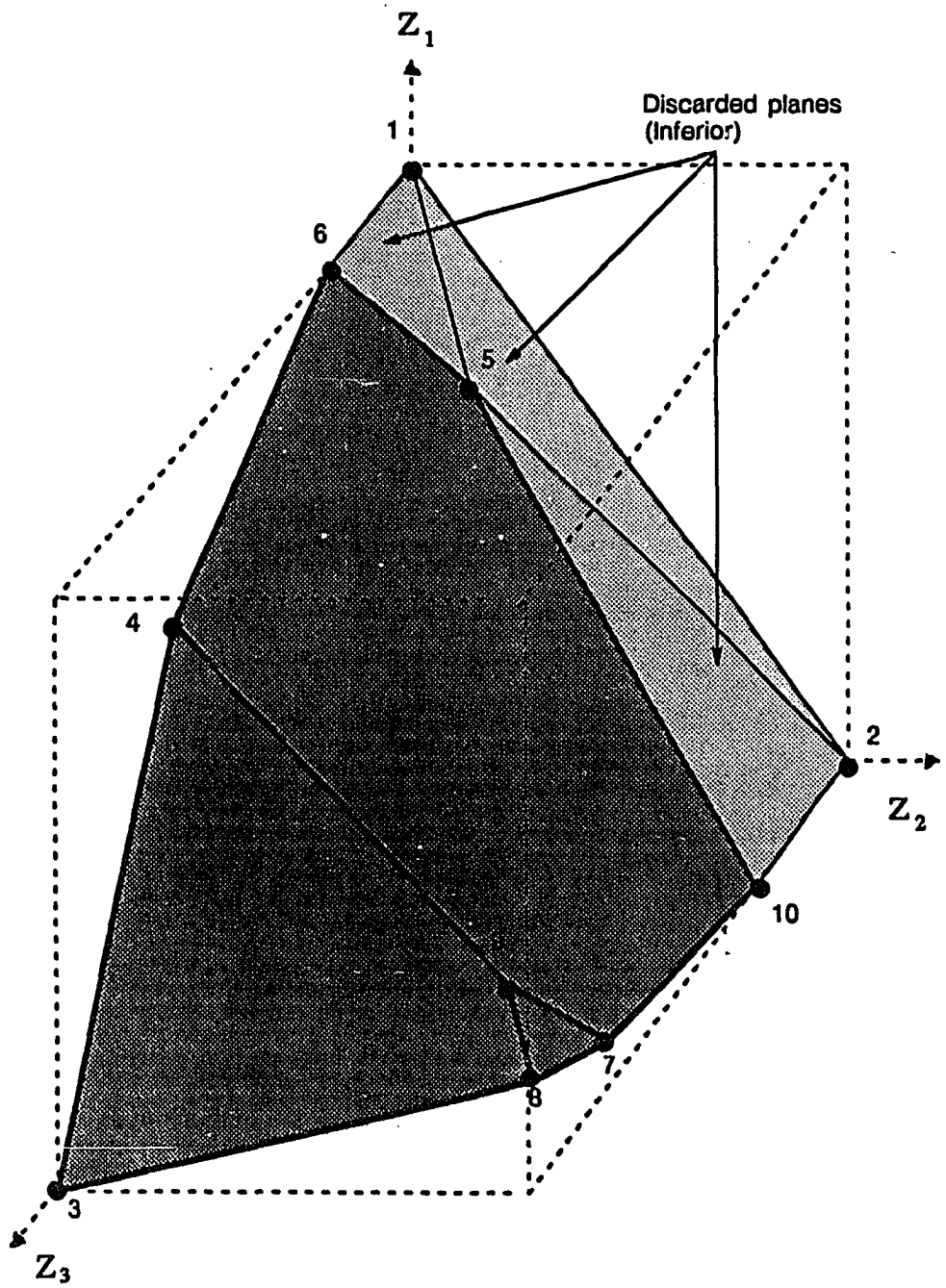


Figure 4.10 Noninferior Points and Planes for the Sample Problem

Table 4.1 Noninferior Points (except for point marked 'inferior')

Point	Values of Objectives (Z1, Z2, Z3)	Values of Decision Variables	Distance
1	(846.95, 0.00, 0.00)	unknown	846.95 *
2	(0.00, 488.34, 0.00)	unknown	488.34 *
3	(0.00, 0.00, 676.19)	(305.89, -23.95,-405.00, 65.09, -201.77)	676.19
4	(695.27, 0.00, 547.48)	(379.95, 59.07, -405.00, -92.45, -81.09)	584.80
5	(662.78, 113.53, 170.32)	(25.52, 102.61, -41.83, -90.58, 75.41)	448.70
6	(846.95, 0.00, 215.98)	(42.03, 135.171, -65.34, -146.05, 85.78)	569.80
7	(0.00, 406.69, 461.62)	(400.00, -75.41, -400.26, 118.81, -157.35)	458.37
8	(0.00, 386.81, 476.04)	(400.00, -73.65, -405.00, 116.44, -161.35)	593.63
9	(128.42, 330.59, 481.36)	(400.00, -51.09, -405.00, 79.874, -144.93)	597.56
10	(0.00, 488.34, 139.27)	(93.02, -32.38, -86.83, 111.90, -19.08)	449.61

* inferior point

Table 4.2 Noninferior Points (except for points marked 'inferior')

Plane	Vertices	Unit Vector (Z1, Z2, Z3)	Distance	
1	(1, 2, 5)	(0.499, 0.866,-0.037)	422.758	(inferior plane)
2	(1, 5, 6)	(0.525, 0.851, 0.000)	444.435	(inferior plane)
3	(6, 5, 4)	(0.473, 0.854, 0.216)	447.243	
4	(4, 7, 9)	(0.473, 0.854, 0.216)	447.243	
5	(4, 9, 8)	(0.162, 0.453, 0.876)	592.601	
6	(9, 7, 8)	(0.218, 0.573, 0.790)	425.075	
7	(4, 8, 3)	(0.162, 0.453, 0.876)	592.601	
8	(5, 2, 10)	(0.492, 0.870, 0.000)	425.075	(inferior plane)
9	(5, 10, 7)	(0.473, 0.854, 0.216)	447.243	
10	(5, 7, 4)	(0.473, 0.854, 0.216)	447.243	

(Note that planes 3, 4, 9, and 10 are the same plane, and planes 5 and 7 are the same other plane.)

triangle determined by these points is an inferior set rather than a noninferior set (except for the vertices). Furthermore, a point such as (64, 358, 478) in objective space may seem noninferior since no noninferior extreme points listed in Table 1 dominates this point. But it is actually an inferior point, which is below noninferior plane 6 listed in Table 4.2. Thus, to present a noninferior set, a list of planes (or surfaces) in addition to the list of extreme points should also be provided (see Table 4.2).

The chronological steps of the Vector Method are listed in Table 4.3, and the sequence of generating the ten final planes are shown graphically in Figure 4.11. The steps are briefly described as follows. First, three points (1,2 , and 3) are generated by maximizing each objective. Point 4 is then generated by using the primary plane 123. Three basis planes are generated: Plans 124, 134, and 423. However, since Plane 134 is redundant, only the other two planes are saved. Although Point 4 is out-and-above the basis plane, 123, used to generated it and an updating procedurce should be implemented, there is no unsearched adjacent basis planes to be updated because it is on the boundary of the search box. After Point 5 is generated, Plane 125 is used as a basis plane but no feasible point exists above this basis plane; it is a final plane. Since this final plane includes an inferior vertex, Point 1, it is an inferior plane and should be discarded (see (a) in Figure 4.11).

Generally, in each iteration three basis planes are generated for following iterations. For those iterations where fewer than three basis planes are generated, there are redundant planes which are deleted (see iterations 1, 4, 7, and 8). If no new point can be located, then the basis plane used in the current iteration is an exact noninferior plane, unless it is a discarded plane because a component of the unit normal vector is negative. Basis planes used in iterations 6, 10, 11, 12, 13, 16, and 17 are exact noninferior planes, while those used in iterations 3, 5, and 15 are discarded ones. A valid updating operation can be observed in iteration 7 where Point 7 is generated by the basis plane 524, and the adjacent plane 423 is updated.

Eight noninferior extreme points exist for this problem; one of them is used to form the primary plane in the first iteration, and seven are generated. In using the Vector Method, seven iterations were performed to generate the seven points, and ten additional iterations were performed to make sure no other noninferior point exists. There were no redundant planes used.

If the NISE method is applied for solving this problem, point 5, which lies on a noninferior plane with more than three noninferior extreme points (the first complexity), will not be located if it does not look beside of each plane, and some redundant basis planes may be used.

Table 4.3. The Steps of the Vector Method for a Sample Problem
Tolerance= 1.0E-04
*: pop a basis plane, an iteration
-: related to the updating procedure
@: a deleted plane
&: a final plane
Ptopp: indicates which vertex is not included in the region on which the projection point of the newly generated point lies (see Section 4.1.3)
Each 'Push' operation relates to a generated basis plane.

Point Index or Indices of Vertices of a plane	Description	Related Figure in Figure 4.11
1	<- A new point generated	
2	<- A new point generated	
3	<- A new point generated	
(1, 2, 3)	<- Push a plane	
* (1, 2, 3)	<- Pop a plane (Iteration 1)	
4	<- A new point generated	
(4, 2, 3)	<- Push a plane	
(1, 2, 4)	<- Push a plane	
- (1, 2, 3)	<- Push an updating plane, Ptopp= 2	
- (1, 2, 3)	<- Pop an updating plane, Ptopp= 2	
* (1, 2, 4)	<- Pop a plane (Iteration 2)	
5	<- A new point generated	
(5, 2, 4)	<- Push a plane	
(1, 5, 4)	<- Push a plane	
(1, 2, 5)	<- Push a plane	
*& (1, 2, 5)	<- Pop a plane (Iteration 3); a discarded plane	(a)
* (1, 5, 4)	<- Pop a plane (Iteration 4)	
6	<- A new point generated	
(6, 5, 4)	<- Push a plane	
(1, 5, 6)	<- Push a plane	
- (1, 5, 4)	<- Push an updating plane, Ptopp= 2	
- (1, 5, 4)	<- Pop an updating plane, Ptopp= 2	
*& (1, 5, 6)	<- Pop a plane (Iteration 5); a discarded plane	(b)
*& (6, 5, 4)	<- Pop a plane (Iteration 6)	(c)
* (5, 2, 4)	<- Pop a plane (Iteration 7)	
7	<- A new point generated	
(5, 7, 4)	<- Push a plane	
(5, 2, 7)	<- Push a plane	
- (5, 2, 4)	<- Push an updating plane, Ptopp= 1	
- (5, 2, 4)	<- Pop an updating plane, Ptopp= 1	
(4, 7, 3)	<- Push a plane	
(4, 2, 7)	<- Push a plane	
-@ (4, 2, 3)	<- Delete a plane	
-@ (4, 2, 7)	<- Delete a plane,	
- (4, 2, 3)	<- Push an updating plane, Ptopp= 1	
- (4, 2, 3)	<- Pop an updating plane, Ptopp= 1	
* (4, 7, 3)	<- Pop a plane (Iteration 8)	
8	<- A new point generated	
(4, 8, 3)	<- Push a plane	
(4, 7, 8)	<- Push a plane	
- (4, 7, 3)	<- Push an updating plane, Ptopp= 1	
(4, 7, 3)	<- Pop an updating plane, Ptopp= 1	
* (4, 7, 8)	<- Pop a plane (Iteration 9)	
9	<- A new point generated	
(9, 7, 8)	<- Push a plane	
(4, 9, 8)	<- Push a plane	
(4, 7, 9)	<- Push a plane	
*& (4, 7, 9)	<- Pop a plane (Iteration 10)	(d)
*& (4, 9, 8)	<- Pop a plane (Iteration 11)	(e)
*& (9, 7, 8)	<- Pop a plane (Iteration 12)	(f)
*& (4, 8, 3)	<- Pop a plane (Iteration 13)	(g)
* (5, 2, 7)	<- Pop a plane (Iteration 14)	
10	<- A new point generated	
(5,10, 7)	<- Push a plane	
(5, 2,10)	<- Push a plane	
(5, 2, 7)	<- Push an updating plane, Ptopp= 1	
(5, 2, 7)	<- Pop an updating plane, Ptopp= 1	
*& (5, 2,10)	<- Pop a plane (Iteration 15); a discarded plane	(h)
*& (5,10, 7)	<- Pop a plane (Iteration 16)	(i)
*& (5, 7, 4)	<- Pop a plane (Iteration 17)	(j)
STOP		

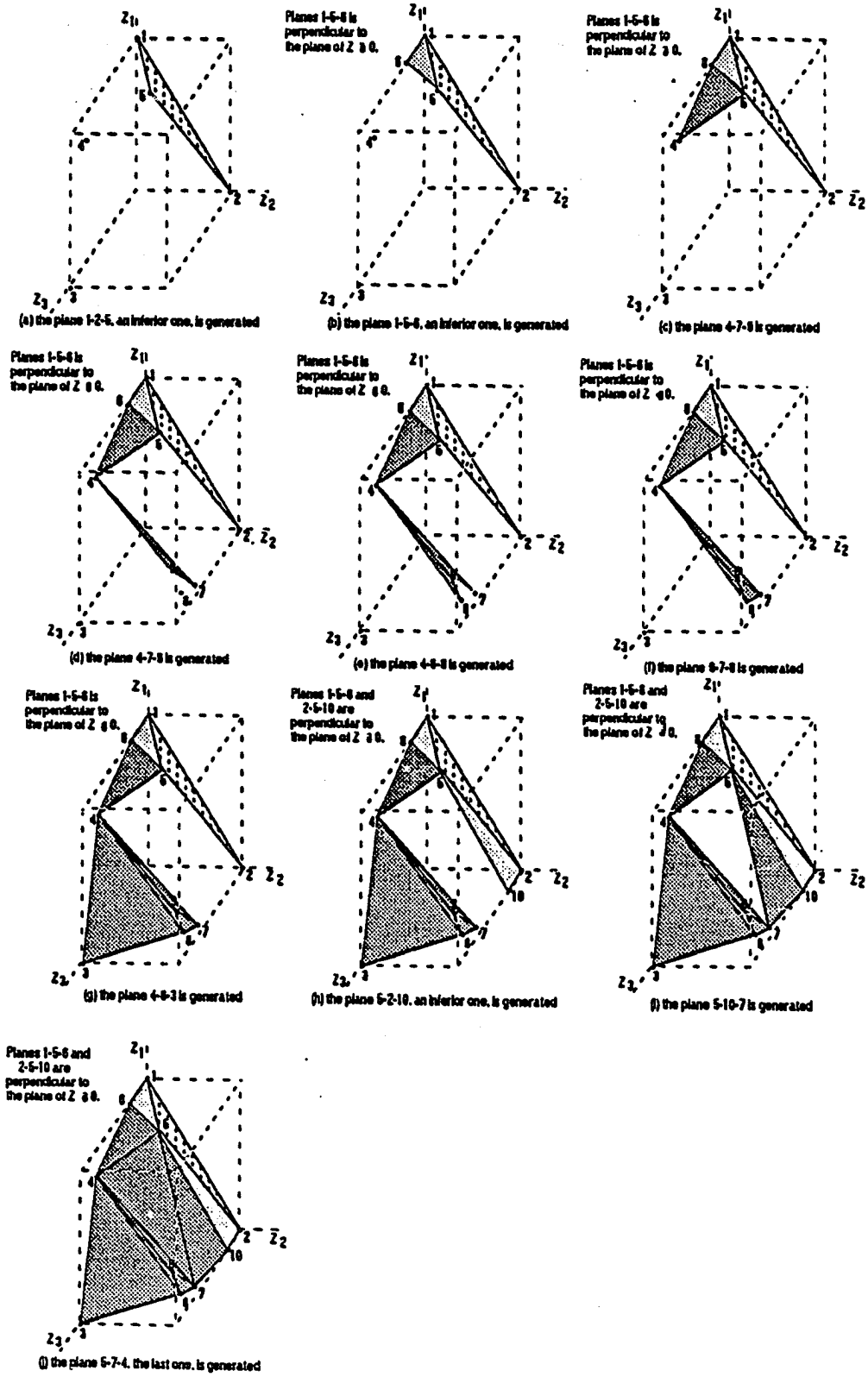


Figure 4.11

For this problem, five basis planes indicate the same plane and only one is really needed as a basis plane. To improve computational efficiency, checks can be made to eliminate duplicate basis planes. The checks can be easily done by comparing unit normal vectors and distances. Another procedure, Phase III based on an idea similar to the two phase procedure used with the Simplex method to obtain a feasible solution, can be used to increase computational efficiency. In any iteration the basis plane used is formed by at least one feasible point that is close to the one that will be generated. The feasible point can be thus used as the starting point in a linear programming algorithm such as one used by XMP to reduce searching time for optimum in next iteration.

4.1.5. N-dimensional Problems

The algorithm described above for a three-dimensional problem can be extended for an n-dimensional problem, although a proof has not been developed. The algorithm for an n-dimensional problem is similar to the main algorithm presented in Section 4.1.2 except for differences described as follows. For each iteration, a n-dimensional surface formed by n points is used as a basis surface. A unit normal vector is expressed as ($\cos\alpha 1$, $\cos\alpha 2$, ..., $\cos\alpha n$) and a surface is determined by

$$\sum_{i=1}^{n} \cos \alpha_i * Z_i = p$$

where p is the distance between the origin and the surface (step 3 in *Initial Steps*).

The way to determine on which side of a basis surface a projection point lies is to extend the procedure used for three dimensional problems. In Figure 4.9, if $\theta 1$ (for AB) is negative, then the projection point is in a region without point B. Similarly, if $\theta 2$ (for AD) is negative, then the projection point is in the region without point D. Point A is the origin point for all vectors used to determine the location of the projection point. Then, the location of the projection point can be determined:

if θ_i is negative, then the projection point, P_p, is in the region without point P_i,

where

$$\sum_{i=1}^{n-1} \theta_i * \overrightarrow{P_iP_n} = P_p$$

P_i's (for i=1,2, ..., n) are the points which form the basis surface;

P_p is the projection of the generation point on the basis surface;

P_n is used as the origin point (as Point A is used for the 3D case).

4.1.6. Phase III For Improving Computational Efficiency

In using the Simplex method to solve a linear programming problem, a two-phase procedure is usually first applied to deal with a starting point that is infeasible. In Phase I, a feasibility objective rather than the model objective is used to obtain a feasible point. After finding a feasible point, the objective function is replaced by the model objective, and Phase II is used to find an optimal solution. This replacement is valid since all the constraints are the same, and the only change is the objective function. Similarly, a multiobjective model can be reconstructed (see the reconstructed model for a sample problem in Section 5) such that at each iteration the constraint set is the same. Then, one of three extreme points of the current basis plane can be used as a feasible starting point for the next iteration (with changes only in the objective function). This procedure is called Phase III; it reduces computational time because the extreme points of current basis plane are the closest known points to the new noninferior extreme point. If no noninferior extreme point can be located from current basis plane, then the search will be terminated immediately.

4.1.7. Checking Inferiority

Although the inferiority of a point can be determined by using a standard LP optimization package two simple algorithms are provided in this section. All points covered by the final plane set (noninferior planes and the discarded planes) are inferior. Points lying above a noninferior plane are infeasible (except for those points that may lie the specified tolerance). Points lying above a discarded plane are either inferior or infeasible. With these observations, the inferiority of a point can be checked easily.

Algorithm Inferiority -1

Since inferior points are covered by the final plane set (the noninferior plane set and discarded plane set), the inferiority of a point can be determined by computing the distances to all planes on the surface using equation 4-3. The following algorithm shows the procedure.

Point to be checked (CP): (Z_{c-1}, Z_{c-2}, Z_{c-3})
Steps

1. use the equation 4-3 to compute the perpendicular distance between CP and each noninferior plane or discarded plane ($p_{new} - p_{old}$).
2. if any distance is zero, CP is a noninferior point if the projection is in region 7 (see Figure 4.9 in Section 4.1.3).
3. if all distances are negative
   then CP is inferior

else if any positive distance is associated with at least one noninferior plane
then CP is dominating but perhaps infeasible
if the distance is smaller than the tolerance, the inferiority is undetermined
else CP is inferior or perhaps infeasible
end if
end if
(Feasibility can be determined by checking all constraints.)

Algorithm Inferiority-2

According to the Krein-Milman theorem described in Lang [1971], a closed convex set can be expressed as :

$$t_0P_0 + t_1P_1 + \ldots + t_mP_m$$

where

$P0, \ldots, Pm$ are extreme points of the convex set;

$0 <= ti <= 1$ for i= 0,1, ... , m; and

$t0+t1+ \ldots + tm = 1$.

If $P0$ is the origin, then the above relationship can be modified as:

$$t_1P_1 + t_2P_2 + \ldots + t_mP_m \qquad (4\text{-}6)$$

where

$P1, \ldots, Pm$ are extreme points of the convex set;

$0 <= ti <= 1$ for i=1,2, ..., m; and

$t1+t2+ \ldots + tm <= 1$.

The final noninferior plane set, discarded plane set, and the origin form a closed convex set. Although we can solve the above relationship for all ti's to determine if a point is inside, on, or outside the convex set, the computations may be extensive for a convex set with many extreme points. A large matrix system would be required, and the solution of equation 4-6 may not be accurate since all ti's are small and roundoff error may be significant.

Instead of using the whole convex set, the set can be partitioned into several small convex sets, each of which is formed by the origin and a plane on the convex surface. Then, only a small matrix system, 3x3 for a 3D case, has to be solved at a time. For example, in Figure 4.12, Plane ABC is a noninferior plane. Any point inside of the shaded object OABC is dominated by one of the points on the noninferior plane. A simple geometrical interpretation of the Krein-Milman theorem for a point, d, which is inside the object is expressed as the following vectorial relationship.

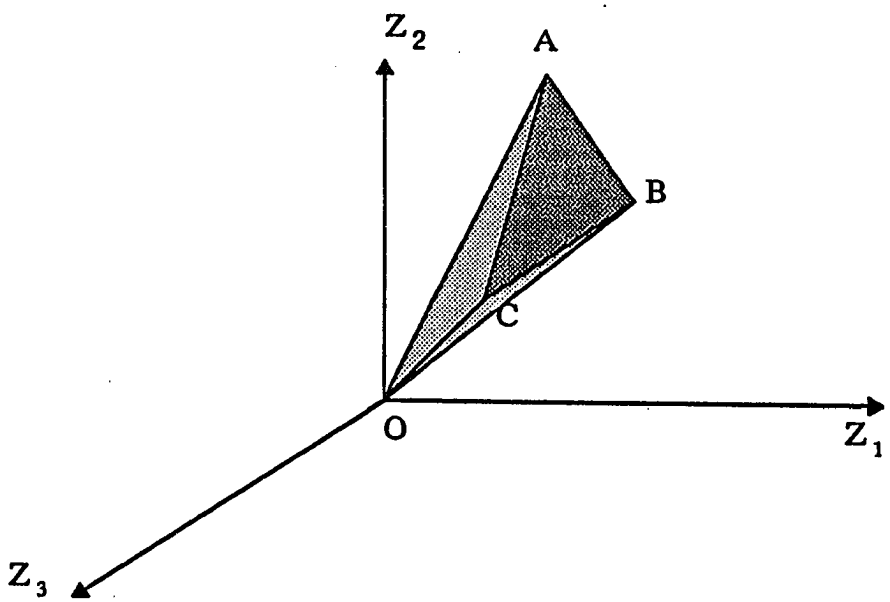


Figure 4.12 Inferiority to a Noninferior Plane

$$\overrightarrow{Od} = \theta_1 \overrightarrow{OA} + \theta_2 \overrightarrow{OB} + \theta_3 \overrightarrow{OC} \qquad (4\text{-}7)$$

where

$\theta 1$, $\theta 2$, and $\theta 3 >= 0$; and

$\theta 1 + \theta 2 + \theta 3 <= 1$.

By solving equation 4-7, the inferiority of a point (CP) can be determined as follows:

if all θ's are greater than or equal to zero and

$\theta 1 + \theta 2 + \theta 3 < 1$, then CP is inferior;

$\theta 1 + \theta 2 + \theta 3 > 1$, then CP dominates a point on the noninferior plane;

CP may be infeasible if not within the tolerance region.

$\theta 1 + \theta 2 + \theta 3 = 1$, then CP is noninferior and on the noninferior plane. (4-8)

If any θi is negative, then the inferiority of CP is undetermined based on the plane used. A negative θi does, however, indicate which adjacent plane should be used as the next checking plane. For example, if $\theta 1$ is negative, then CP is at the side of the plane OBC; and if $\theta 2$ is negative, the it is at the side of OAB. Except points below the lower bounds, all feasible points must satisfy one of the conditions listed above (4-8); and the algorithm will take a short path to find the covering plane. The details of the algorithm are listed below.

Point to be checked (CP): (Zc-1, Zc-2, Zc-3)

Steps

1.seclect any plane from the final noninferior or discarded plane set.
2.check the inferiority of CP using relationship (4-8)
3.if the inferiority is determined- all θi's are positive
then
1.if current checking plane is a noninferior plane, then the inferiority
of CP can be obtained based on relationship (8).
2.if CP lies on a discarded plane, then it is inferior.
3.if CP dominates one of the points on a discarded plane then
1.check the perpendicular distances of CP to all adjacent noninferior planes
or discarded planes.
2.if any distance is strictly greater than zero
then CP is dominating but perhaps infeasible
if the distance is smaller than the tolerance, the inferiority is undetermined
else CP is inferior or perhaps infeasible
end if
end if
4.stop the procedure.
else

1.select an adjacent plane (based on a negative θ_l) as the next checking plane.
2.go to Step 2.
end if

4.1.8. Summary

The main idea of the Vector Method is to approximate the noninferior set by a convex set and to start from a minimal one. At each iteration, the convex set is expanded to be closer and closer to the real noninferior set. This method can be used to obtain the noninferior surface for a multiobjective problem. The formulations and procedures are presented with enough details for an analyst to translate them directly into a computer program such as the one used in this work to solve a three objective problem. In addition, the method can be improved for computational efficiency as described in Sections 5 and 7 by checking duplicate basis planes and using Phase III.

The inferiority of a point can be easily determined using the algorithms described in Section 7. This capability provide answers to 'what-if' questions. Since the final set of planes provides a good approximation to the noninferior surface, it is very desirable to use the whole surface rather than only extreme points to aid decision-making. A graphic display such as that shown for the sample problem can be used for this purpose.

4.2. Modeling Language

Although many mathematical packages, such as XMP, have been developed to optimize a mathematical model, the interface between an analyst and a package is usually not straightforward. For example, the left column of Table 4.4 lists a typical XMP input data set [Marsten, 1984] in MPS format [CDC, 1979] for a resource allocation model. Although it is systematically formatted, understanding the model structure of this data set is not easy for someone who is not a regular XMP or MPS user. Moreover, for a complex problem, it is usually possible to understand only partially a given system at the beginning of an analysis. During the alternative exploring stage, it is very likely that other attributes, constraints, or objectives will be identified. A model may be dynamically changed, but the typical mathematical packages are often not easy to use to implement these changes. The new modeling language presented here is easy to understand and use and thus provides flexibility to extend an existing model.

The right column of Table 4.4 shows the improved form of the input data set using the proposed modeling language. With this kind of interface, which is shown in general modeling form, not only can a regular XMP user form a model easily, but also an occasional user can figure out the model directly. Additionally, if the modeling language can be standardized, it could serve as an interface among different packages. Many engineers have benefited from using the standardized FORTRAN language; its portability allows it to be used on different machines. Similarly, the portability of software packages can potentially be extended also. The modeling language suggested herein is to provide one easy-to-use interface among packages and users. This modeling aid can facilitate the tasks for building or modifying a model (stage A) and being an interface to mathematical packages (stage B). The syntax of the developed modeling language is described and discussed with several examples as follows.

4.2.1. General

The preliminary version of the new modeling language was divided into several sections, such as OBJECTIVES, CONSTRAINTS, and BOUNDS. Although this division is commonly seen in other modeling languages, it has the significant drawback of preventing a modeler from using his own modeling form. The current version has no section divisions, and the user can be free to use any form to organize a model as long as it is syntactically correct. Another significant difference between the new modeling language and the others is that several objectives can be defined for a multiobjective problem.

Another feature is that a constant can be defined where it is first used; many modeling and programming languages require users to define all constants at the beginning of a model or computer code. The advantage of this restriction is that all definitions are kept together. However, it is inconvenient

Table 4.4. A Sample Mathematical Model

(Extracted From XMP Manual [Marsten,1985])

```
COST  -1.0
 1 2 1
NAME          RADEX ELECTRONICS
ROWS
    N  COST
    E  LABOR1
    L  PROD1
    L  OVERLIM1
    E  LABOR2
    L  PROD2
    L  OVERLIM2
    E  BALANCE2
    E  CUM2
    E  LABOR3
    L  PROD3
    L  OVERLIM3
    E  BALANCE3
    E  CUM3
    G  LABOR4
    E  BALANCE4
    E  CUM4
COLUMNS
        NEW1          COST          1060.
        NEW1          PROD1         -3.0
        NEW1          LABOR2        -0.95
        EXP1          COST          860.
        EXP1          LABOR1        1.0
        EXP1          PROD1         -4.0
        EXP1          OVERLIM1      -35.
        EXP1          LABOR2        -0.95
        RADAR1        PROD1         1.0
        RADAR1        BALANCE2      -1.0
        OVER1         COST          7.5
        OVER1         PROD1         -0.023256
        OVER1         OVERLIM1      1.0
        NEW2          COST          1060.
        NEW2          PROD2         -3.0
        NEW2          LABOR3        -0.95
        EXP2          COST          860.
        EXP2          LABOR2        1.0
        EXP2          PROD2         -4.0
        EXP2          OVERLIM2      -35.
        EXP2          LABOR3        -0.95
        RADAR2        PROD2         1.0
        RADAR2        BALANCE3      -1.0
        OVER2         COST          7.5
        OVER2         PROD2         -0.023256
        OVER2         OVERLIM2      1.0
        SHIP2         BALANCE2      1.0
        SHIP2         CUM2          1.0
        SHIP2         CUM3          1.0
        SHIP2         CUM4          1.0
        INVENT2       COST          10.
        INVENT2       BALANCE2      1.0
        INVENT2       BALANCE3      -1.0
        LATE2         COST          50.0
        LATE2         CUM2          1.0
        NEW3          COST          1060.
        NEW3          PROD3         -3.0
        NEW3          LABOR4        -0.95
        EXP3          COST          860.
        EXP3          LABOR3        1.0
        EXP3          PROD3         -4.0
        EXP3          OVERLIM3      -35.
        EXP3          LABOR4        -0.95
        RADAR3        PROD3         1.0
        RADAR3        BALANCE4      -1.0
        OVER3         COST          7.5
        OVER3         PROD3         -0.023256
        OVER3         OVERLIM3      1.0
        SHIP3         BALANCE3      1.0
        SHIP3         CUM3          1.0
        SHIP3         CUM4          1.0
        INVENT3       COST          10.
        INVENT3       BALANCE3      1.0
        INVENT3       BALANCE4      -1.0
        LATE3         COST          50.0
        LATE3         CUM3          1.0
        SHIP4         BALANCE4      1.0
        SHIP4         CUM4          1.0
RHS
        RHS           LABOR1        90.
        RHS           CUM2          300.
        RHS           CUM3          700.
        RHS           LABOR4        200.
        RHS           CUM4          1000.
BOUNDS
ENDATA
```

(In the Developed Modeling Language)

```
(NAME: RADEX ELECTRONICS)
Min Cost = 1060 NEW1  + 860 EXP1 + 7.5 OVER1 + 1060 NEW2 +
860 EXP2 + 7.5 OVER2 + 10 INVENT2 + 50 LATE2 + 1060 NEW3
+ 860 EXP3 + 7.5 OVER3 + 10 INVENT3 + 50 LATE3;
LABOR1: EXP1 = 90;
PROD1:      -3.0 NEW1 - 4 EXP1 + RADAR1 -0.023256 OVER1 < 0;
OVERLIM1: -35 EXP1 + OVER1 < 0;
LABOR2:   -0.95 NEW1 -0.95 EXP1 + EXP2 = 0;
PROD2:      -3.0 NEW2 - 4 EXP2 + RADAR2 -0.023256 OVER2 < 0;
OVERLIM2: -35 EXP2 + OVER2 < 0;
LABOR3:    -0.95 NEW2  - 0.95 EXP2 + EXP3 = 0;
PROD3:      -3.0 NEW3 - 4 EXP3 + RADAR3 -0.023256 OVER3 < 0;
OVERLIM3: -35 EXP3 + OVER3 < 0;
BALANCE2: -RADAR1 + SHIP2 + INVENT2 = 0;
BALANCE3: -RADAR2 -INVENT2 + SHIP3 + INVENT3 = 0;
BALANCE4: -RADAR3 - INVENT3 + SHIP4 = 0;
CUM2:       SHIP2 + LATE2 = 300;
CUM3:       SHIP2 + SHIP3 + LATE3 =  700;
CUM4:       SHIP2 + SHIP3 + SHIP4 = 1000;
```

OR

```
(NAME: RADEX ELECRONICS)

Min Cost = sum(1060 NEWI + 860 EXPI + 7.5 OVERI with i= 1 to 3) +
        sum(50 LATEJ + 10 INVENTJ with j= 2 to 3);

Subject to (optional)

    LABOR1:    EXP1 = 90;
    LABOR2:    -0.95 NEW1 -0.95 EXP1 + EXP2 = 0;
    LABOR3:    -0.95 NEW2  - 0.95 EXP2 + EXP3 = 0;
    LABOR4:    -0.95 NEW3  - 0.95 EXP3 ? 200;
    BALANCE2: -RADAR1 + SHIP2 + INVENT2 = 0;
    BALANCE3: -RADAR2 -INVENT2 + SHIP3 + INVENT3 = 0;
    BALANCE4: -RADAR3 - INVENT3 + SHIP4 = 0;
    CUM2: SHIP2 + LATE2 = 300;
    CUM3: SHIP2 + SHIP3 + LATE3 =  700;
    CUM4: SHIP2 + SHIP3 + SHIP4 = 1000;

    repeat with i= 1 to 3
        PRODI: -3 NEWI - 4 EXPI + RADARI - 0.023256 OVERI < 0;
        OVERLIMI: -35 EXPI + OVERI < 0;
    end repeat;
```

for a modeler to add a definition in the middle of developing a model because he must go back to the top of a model. Furthermore, to define a constant before it is used would make a model (or a computer code) difficult to read. A reader must understand all definitions at the same time for understanding a model. If the set of definitions is large, it may be difficult to understand all definitions at the same time particularly before they are used. Logically, it is better to define a constant at its first appearance. This approach reduces the number of definitions which the reader should keep track at a time, and thus a model should be easier to read and maintain. The modeling language provides a way to define a constant locally. By localizing the definition of constants, it is easier for the user to keep track of them.

4.2.2. Syntax

The syntax of the new modeling language is as follows. For explanation purpose, a sample form is shown in Table 4.5, in which a water quality management problem including 7 water quality checkpoints and 4 dischargers is shown. However, the form can be reorganized, if preferred, using the flexibility of the modeling language.

Documentation and Annotation

Documentation and annotation are usually very important in setting up a model; they are important for presentation and for allowing others to understand the model. The modeling language allows a modeler to add comments for documentation or annotation. Comments are delimited by '{...}' (see label A in Table 4.5).

OBJECTIVES

The modeling language is designed for multiobjective problems or single objective problems. An objective is defined by the modeling language just as a constraint except it has a key word of 'Max', 'Maximum', 'Maximize', 'Min', 'Minimum', or 'Minimize' added to indicate the optimization direction (see label B in Table 4.5). An objective may be assigned a name. For example, the first objective in Table 4.5 (label B) is named Equity, while the third one (label D) is not assigned a name. An objective can be put anywhere in a model as long as a key word which indicates the optimization direction is added.

CONSTRAINTS

The type of a constraint is indicated by '>' (greater than and equal to), '<' (less than and equal to), or '=' (equal to) (see label E in Table 4.5). The left-hand-side of a constraint can be assigned a name, and the name can be used as a model variable (see labels H and J in Table 4.5).

BOUNDS

A bound for a model variable is a special constraint which has only one model variable. Since a variable bound definition is similar to a constraint definition, there is no need to distinguish it in a modeling form. However, since most software packages such as XMP distinguish bounds from constraints for computational reasons, the modeling language would convert all one variable constraints into bounds before using a package such as XMP. The bounds can be specified using '>' (greater than and equal to), '<' (less than and equal to), or '=' (equal to). The modeling language can be extended to allow two sided bounds, if desired. A bound for an objective can be set either with the objective function (see label B in Table 4.5) or as an one variable constraint (see label K in Table 4.4).

SUMMATION

Summations of sets of variables commonly appear in linear models. The key word 'Sum' is used to indicate a summation (see label B or E in Table 4.5). For example,

Sum(ui + vi with i= 1 to 4) is equivalent to

u1+v1+u2+v2+u3+v3+u4+v4.

REPLICATION

For a small problem such as the sample problem shown in Section 4.1, all constraints can be readily written one by one. For a large model, however, to enter all constraints one by one would be difficult and it would be easy to make mistakes. Thus, a repeat-loop is provided to make this effort easier if many constraints have a similar form. For example, the following sixteen constraints are similar.

e1-e11-e12=0
e1-emin > 0
emax-e1 > 0
e1+u1-v1-ea=0
e2-e21-e22=0
e2-emin > 0
emax-e2 > 0
e2+u2-v2-ea=0
e3-e31-e32=0
e3-emin > 0
emax-e3 > 0
e3+u3-v3-ea=0
e4-e41-e42=0
e4-emin > 0
emax-e4 > 0
e4+u4-v4-ea=0

These constraints can thus be simplified as:

```
repeat with k=1 to NumDischarge
   ek-ek1-ek2=0;
   ek-emin > 0;
   emax-ek > 0;
   ek+uk-vk-ea=0;
end repeat;  (see also label F in Table 4.5)
```

The modeling language also provides a shorthand for a single line repeat-loop. For example, the following eight constraints

```
1.367375 e1 > 0.467
1.175146 e1+ 2.473486 e2 > 2.465
0.997637 e1+ 4.276584 e2 + 0.24546 e3 > 4.538
0.822017 e1+ 5.190134 e2 + 0.390162 e3 + 0.42006 e4 > 5.373
0.742592 e1 + 5.28186 e2 + 0.419358 e3 + 0.529009 e4 > 5.355
0.705100 e1 + 5.266755 e2 + 0.426553 e3 + 0.565714 e4 > 5.274
0.669098 e1 + 5.219727 e2 + 0.429795 e3 + 0.592763 e4 > 5.157 and
0.585731 e1 + 4.997476 e2 + 0.42452 e3 + 0.626824 e4 > 4.745
```

can be expressed by

```
Sum(Aij*ei with i=1 to NumDischarge) > Bj with j= 1 to NumCheckpoint;
```

instead of

```
repeat with j= 1 to NumCheckpoint
   Sum(Aij*ei with i=1 to NumDischarge) > Bj;
end repeat;(see also label G in Table 4.5)
```

This repeat-loop capability saves time in entering and modifying many constraints which are similar to each other.

CONSTANT AND GROUPED VARIABLE DEFINITION

A model usually includes many numbers. The numbers themselves are often not meaningful, although they may be associated with meaningful items. To make a model easier to read, it is possible to use a meaningful name for a number. For example, the number 1317.85 itself is not very meaningful. If it is assigned a name such as FIXcost (see label C in Table 4.5), then it is easy to understand that the number is a fixed cost. Constant definitions can be used to improve model clarity.

In the modeling language, there are two types of constant definitions: global and local. FIXcost is only used in the second objective shown in Table 4.5 and is better defined as a local constant. The key word 'with' indicates an local constant definition. On the other hand, if a constant is used more than once, it may be better to define it as a global constant. An example is shown in Table 4.5 (see label B) where the constant 'NumDischarge' is defined as a global constant for the number of dischargers. It is used several times in the model. The key word 'where' is used to define a global constant. Sometimes an array of

constants may be used, such as the impact coefficients Aij used in Table 4.5, and their definition may be too long to put within a constraint. These constants may be defined separately somewhere in a model (see label M in Table 4.5). If these constants are defined separately from where they are first used, they should be defined as global constants by using the key word 'where'. If they are defined at the beginning of a model the alternative key word 'Assign' instead of 'Where' can be used. The constant definition not only promotes easy understanding of the model, but also allows easy changes in the model.

The other type of definitions is for grouped variables. For example, the monthly expenses variables, Jan, Feb, ...etc., are used in a group in the model shown in Table 4.5. They are thus grouped and expressed by 'Months[i]', where i = 1 to 12 (see also label N in Table 4.5). Then, Months[1] is for Jan, Months[2] is for Feb, ...etc. Thus, a summation of some of these variables can be expressed easily. For example, 'Sum(Months[i] with i= 1 to 11 by 2)' (see also label I in Table 4.5) is equivalent to 'Jan+Mar+May+Jul+Sep+Nov'.

OTHER SYNTACTIC RULES

The following list describes other syntactic rules for the modeling language:

(1) each objective or constraint can be continued over several lines, but ';' must be added at the end of the objective or constraint. Independent global constants or grouped variable definitions must be ended by a ';' too, however, several definitions may be linked by the key word 'and' (see label L in Table 4.5);

(2) the right hand side of a constraint must be a number or a defined constant;

(3) an index for a summation clause, repeat-loop, or constant definition is in the form of indexname= start-value to end-value by increment-value, where indexname must be a letter and start-value, end-value, and increment-value must be numbers or defined constants. Omission of an increment-value defaults to a value of +1;

(4) a constant can be nestedly but not recursively defined; (for example,
where AA= r1+r3 and r1= 33 and r3= r5+30 and r5= 90; but not
where AA= r1+r3 and r1= AA+r5;)

(5) a global constant can be defined anywhere in a model and it affects the model globally; thus a global constant can be defined after it is used.

The modeling language has been coded in PASCAL. So far it can handle a linear problem only. The modeling form in the modeling language are understandable by both the computer software and modelers. It can serve as an interface to the mathematical package(s) chosen. Furthermore, the modeling time can be significantly reduced for a large model which has many similar constraints. To build a model is thus much simpler and the model is easy to maintain.

Table 4.5. A Sample Water Quality Management Model

```
{ Model Name: Illini River Water Quality Management Model.
  Modeler: John, Smith.
  Date: xx/xx/xx                                                    }          (A)

Min Equity= Sum(ui + vi with i= 1 to NumDischarge) > 0 where NumDischarge = 4;  (B)
Min TotalCost= Sum(Ci1*ei1+Ci2*ei2 with i= 1 to NumDischarge) +
        Sum(Months[i] with i= 1 to 12) + FIXcost with FIXCost= 1317.85;         (C)
Min emax - emin;                                                                (D)

Subject to  {optional key words}

Sum(ei with i= 1 to NumDischarge) - NumDischarge*ea = 0;                        (E)
repeat with k=1 to NumDischarge                                                 (F)
        ek-ek1-ek2=0;
        ek-emin > 0;
        emax-ek > 0;
        ek+uk-vk-ea=0;
end repeat;
Sum(Aij*ei with i=1 to NumDischarge) > Bj with j= 1 to NumCheckpoint;            (G)

EvenMonth: Sum(Months[j] with j= 2 to 12 by 2) < 4000;                          (H)
OddMonth: Sum(Months[i] with i= 1 to 11 by 2) < 3000;                           (I)
Jan + May + Sep + Nov < 2500;
OddMonth: + EvenMonth: < 6000;                                                  (J)

emax < UpperE with UpperE = 0.95;
emin > LowerE  with LowerE = 0.3;

TotalCost < 9000;                                                               (K)

where
        NumCheckpoint= 8 and                                                    (L)
        Cij with i= 1 to NumDischarge and j= 1 to 2 = (  {Cost coefficients}
          842.6      1853.03
          876.53     2503.72
          269.13     1161.15
          1132.6     2877.47)  and
        Bj with j= 1 to NumCheckpoint = (    {use DO standard= 6.0 mg/l}
          .467       2.465      4.538      5.373      5.355      5.274      5.157      4.745)  and
        Aij with i= 1 to NumDischarge and j= 1 to NumCheckpoint= (               (M)
            {impact coefficient of discharger i on checkpoint j }
          1.367375   1.175146   0.997637   0.822017   0.742592   0.705100   0.669098   0.585731
          0.0        2.473486   4.276584   5.190134   5.281860   5.266755   5.219727   4.997476
          0.0        0.0        0.245460   0.390162   0.419358   0.426553   0.429795   0.424520
          0.0        0.0        0.0        0.420060   0.529009   0.565714   0.592763   0.626824) and
        Months[i] with i= 1 to 12= (       {Monthly expenses}                   (N)
          Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
```

4.3. MGA Methods

The MGA approach is designed to explore alternatives that are good with respect to mathematically modeled objectives but significantly different from each other. The purpose of generating good alternatives is to gain insight into a problem and to help the analyst consider alternatives while taking into account unquantifiable or unmodeled issues. The method is implemented in an iterative fashion. The differences among generated alternatives are defined based on a difference objective function. 'Good' solutions are obtained by setting a constraint on values of the original modeled objective(s).

In this section, the geometrical and mathematical expressions of the MGA approach are presented; the HSJ method is reviewed; and two modified HSJ methods are presented at the end, although they were not coded into the prototypes. For ease in implementing the MGA approach, difference functions used for the MGA method should be systematically determined. The HSJ method [Brill, 1979] is one of several choices, which are described geometrically in Section 4.3.1.

4.3.1. Geometric Expressions of the HSJ Method

The original HSJ method shown below is designed for a problem whose solutions generally have a significant number of variables that are equal to zero. The method was tested by Chang [1982] for a mixed-integer facility location problem.

$$Min \sum_{k \in K} x_k$$

$$S.T. \quad X \in F_d$$

where

x_k's are model variables;

K is the set of indices of non-zero variables for all previously generated alternatives;

X= (x1, x2, ... , xn);

n is the number of modeled variables; and

Fd is the feasible space of the model.

Since linear problems do not necessarily have a significant number of zero-valued variables, a generalized HSJ method was suggested by Kshirsagar [1984] and is expressed below; a simpler form is presented here, although the generalized HSJ method was originally interpreted as an inner product measure.

$$Min \sum_{i=1}^{n} h_i * x_i \qquad \text{(B-1)}$$

$$S.T.\ \ X \in F_d$$

where

n is the number of variables;

$$h_i = \sum_{j=0}^{p} C_{j,i}$$

where

p is the number of previously generated alternatives; and

Cj,i is the value of variable i in alternative j;

xi's are model variables;

X= (x1, x2, ... , xn);

n is the number of modeled variables; and

Fd is the feasible space of the model.

For a two variable problem, the generalized HSJ function can be expressed as:

$$Min\ h_1 * x_1 + h_2 * x_2$$

$$S.T.\ \ (x_1, x_2) \in F_d$$

where

$$h_i = \sum_{i=0}^{p} C_{j,i} \qquad \text{for i =1, 2;}$$

where

p is the number of previously generated alternatives; and

Cj,i is the value of the variable i for alternative j;

x1 and x2 are model variables; and

Fd is the feasible space of the model.

The geometric expression in a two-dimensional domain for the first iteration of using the generalized HSJ function can be viewed as in Figure 4.13. Point A, ($C_{0,1}$,$C_{0,2}$), is the optimal solution to the original model. Then, the difference objective function used at the first iteration by using the generalized HSJ method is

$$Min\ C_{0,1}x_1 + C_{0,2}x_2.$$

The search direction oriented by this difference function can be expressed as the vector AO shown in Figure 4.13.

Let the solution obtained from the first iteration be the point B, (C1,1, C1,2), as shown in Figure 4.14. Based on the HSJ function shown in equation B-1, the difference function used at the second iteration is

$$Min\ (C_{0,1} + C_{1,1})x_1 + (C_{0,2} + C_{1,2})x_2$$

The search direction at the second iteration can be expressed as the combined vector, CO, of vectors AO and BO (see Figure 4.14).

The procedure is continued until the desired number of alternatives are generated. The search direction at each iteration can be expressed as the combined vector of the previously used search vector and the vector originating at the last generated point and directed toward the origin. The search directions used are all toward the origin, O. However, for problems where most variables are non-zero (e.g. few nonbasic variables in an LP), it is usually difficult to produce widely different solutions using the original HSJ method. In such cases, it may be desirable to use a point (or different points) other than the origin to orient search directions. For illustration purposes such a point is called an orientation point in the next section, where two modified HSJ methods are introduced.

4.3.2.Two Modified HSJ Methods

Two alternative HSJ methods are described in this section. In each case, the first iteration is the same as in the original HSJ method. After the first iteration, several modifications have been developed to improve the ability to generate different alternatives.

HSJ Walk Method

Although the generalized HSJ method is systematic and simple, it may not generate "maximally" different alternatives since the orientation point is the same for all iterations. Thus, the first modified HSJ method, called the HSJ Walk method, is based on changing the orientation point at each iteration. A systematic procedure to change the orientation point is as follows. In Figure 4.15, Point A is the optimal solution to the modeled objective, and Point B is obtained by using the generalized HSJ method described in last section; the search direction is AO and the origin is the orientation point. Next, the previously obtained point, A, is used as the orientation point at the second iteration, and the new search direction can be expressed as the vector CA, which is the combination of vectors OA and BA. Point B is the orientation point used in the third iteration. The orientation point walks from the origin to Point A and then to Point B in

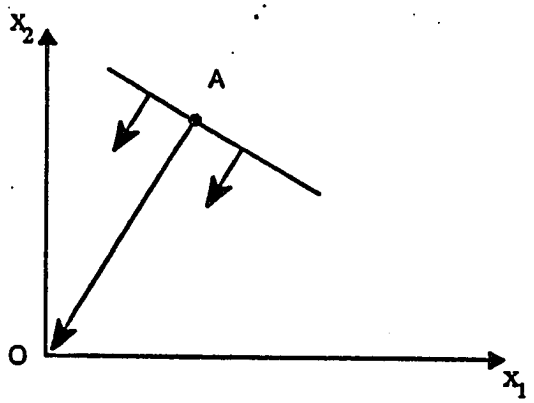


Figure 4.13 The First Iteration of Using the generalized HSJ Method– a 2D case

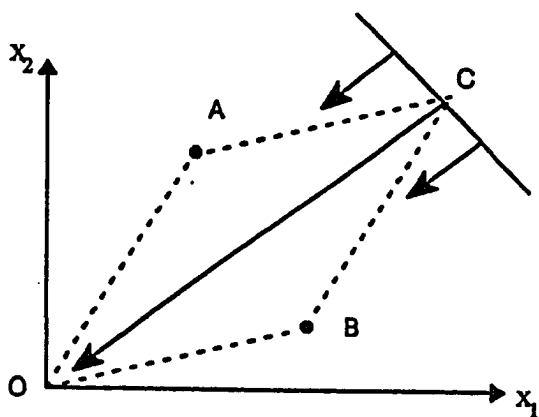


Figure 4.14 The Second Iteration of Using the generalized HSJ Method – a 2D case

three iterations. This procedure is continued until the desired number of alternatives is obtained. By using a variety of orientation points, the suggested method may perform better in generating different alternatives.

The formulation of the HSJ Walk function is:

$$Min \sum_{i=1}^{n} h_i * x_i$$

$$S.T. \quad X \in F_d$$

where

$$h_i = - C_{op,i} + \sum_{j=0}^{p} (C_{j,i} - C_{op,i}) = - (p+2) * C_{op,i} + \sum_{j=0}^{p} C_{j,i}$$

$C_{op,i}$: is the coefficient of variable i of the orientation point used in current iteration.

Line-Oriented HSJ Method

The search direction can also be based on a line by connecting two previously generated points. In Figure 4.16, Points A and B are the same as those obtained after the first iteration in using the HSJ method. The search direction at the second iteration in using the line-orientation HSJ method is the vector CO shown in the figure, which is perpendicular to the line AB. Note that the other direction, CD, which is opposite to the search direction CO, can be also used as a search direction, representing a variant method. Similar variations can be also applied to other HSJ functions. For example, in Figure 4.13 and 4.14, instead of using vectors AO and CO as search directions, vectors OA and OC can serve as search directions too. The orientation point used is still the origin, but it can be changed if desired.

The two new HSJ methods described above are designed for systematic implementation on a computer. There are many variations that can be derived from these two methods. For example, instead of using the search direction which is toward the origin, the opposite direction can be used, or an arbitrary point can be used as the orientation point at each iteration. These functions have not been tested, however, for any problem.

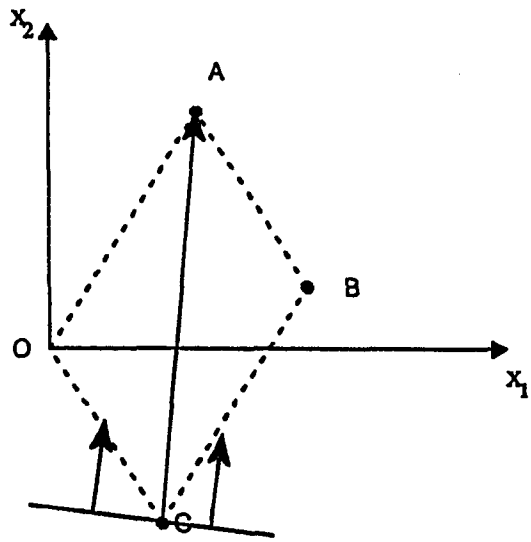


Figure 4.15 The Second Iteration of Using the HSJ Walk Method – a 2D case

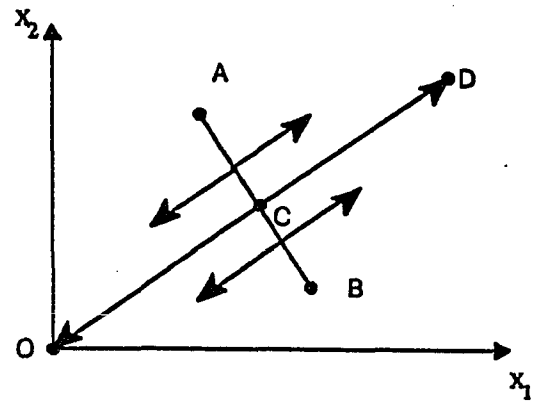


Figure 4.16 The Second Iteration of Using the Line-Oriented HSJ Method – a 2D case

CHAPTER 5

COMPUTER AIDED SYSTEM FOR WASTEWATER TREATMENT PLANT DESIGN

5.1. Wastewater Treatment Plant Design (WTPD)

The base WTPD model and analysis program adopted in this research were developed by Tang [1987] and modified by Geselbract (see Kao, et al.[1989]). The base model is for a complete secondary wastewater treatment plant, including sludge processing and disposal. In addition to individual unit process performance, the model considers the interactions among various unit processes. Since the mathematical development of the model and analysis program was provided in detail elsewhere (Tang [1987] and Kao, et al. [1989]), only a brief overview is presented below.

Figure 5.1 provides a typical process flow sheet showing the individual unit processes and various connecting flows. The unit processes included in the model are primary clarification, activated sludge with final clarification, gravity thickening of mixed primary and waste activated sludge, primary and secondary anaerobic digestion, vacuum filtration, and final sludge disposal via a sanitary landfill. Since a plant scheme can be dynamically changed based on design needs, the combination of unit processes may be different from the one shown in Figure 5.1.

5.2. Design Approach

A major design question is how does (or should) an engineer design (determine the sizes of the units in) a wastewater treatment process (see also Kao, et al.[1989]). It is desirable to design a processing scheme and to size units so that the complete system works, is efficient, and performs reliably.

Conventional Approach

A conventional approach is described in the wastewater engineering text by Metcalf & Eddy (M & E) [1979] for designing an activated sludge process. The design is initiated by giving the influent conditions as well as two process design variables (sludge age and mixed liquor volatile suspended solids, MLVSS) and the return sludge concentration. Thus, from a "design analysis" point of view, the problem is already solved; all that remains is to use these variables to calculate the resulting state variables. Aeration volume is conservatively determined based on the soluble BOD removal required under the condition of high effluent suspended solids (probably the effluent standard). Checks are then conducted on the resulting hydraulic retention time and the volumetric loading rate. The text guides the user in selecting those design

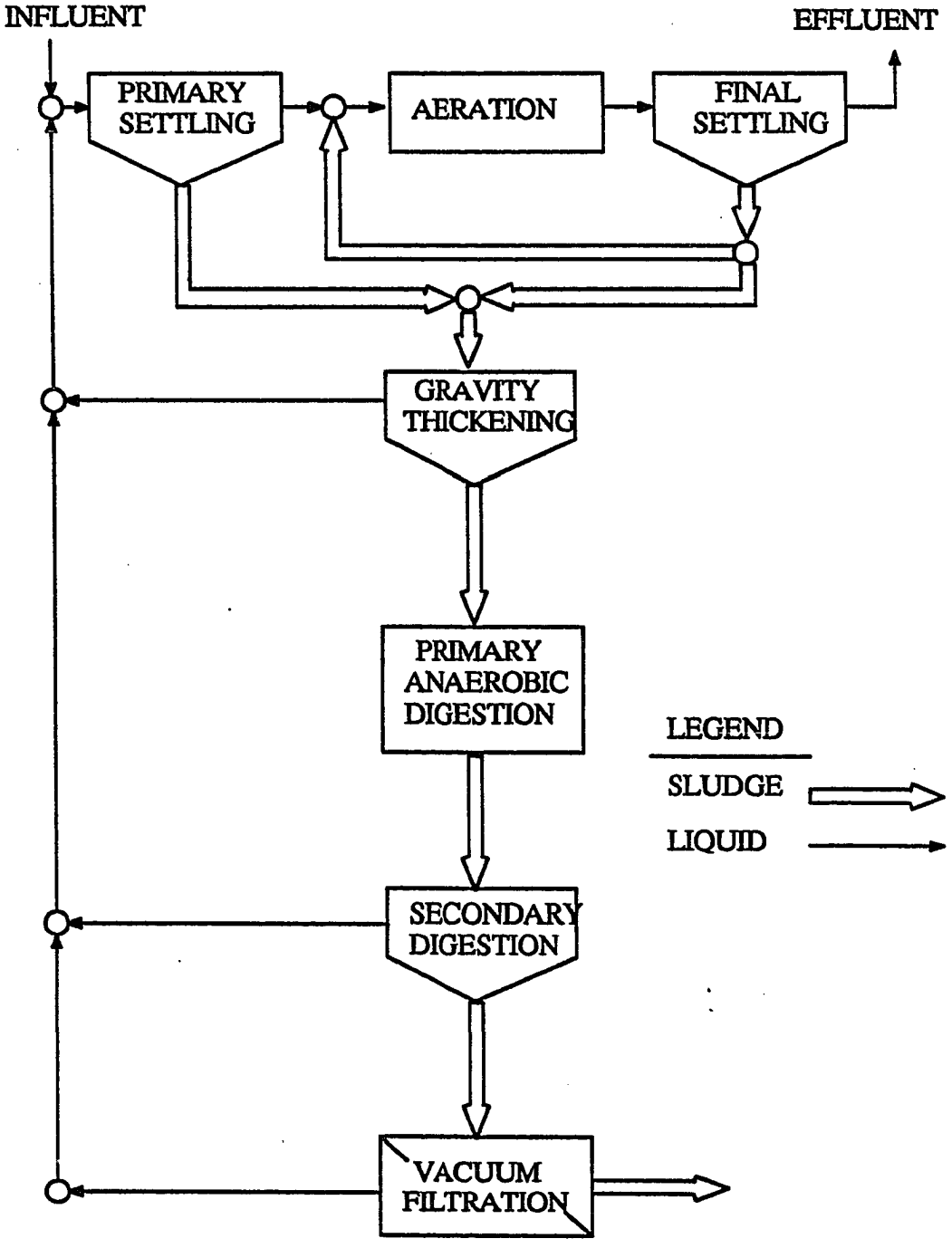


Figure 5.1 Wastewater Treatment Plant Flow Diagram

variables by providing recommended ranges. It also warns of other factors which must be considered during the design, including cost. Unfortunately, the nature of the interactions between the design variables and the resulting cost and reliability of the design is not explicit.

Computer Aided System Approach

The preliminary design of an activated sludge plant includes sizing the aeration basin and final clarifiers. Feasible sizes for those units are constrained to a large degree by allowable loading rates specified by State or Public Health standards. For example, Table 5.1 presents some typical recommended loading rates for these units.

Table 5.1. Recommended Activated Sludge Loadings

Criterion	10 State	V & H	M & E
Maximum HRT (hrs)	—	7.5	8
Minimum HRT (hrs)	—	6.0	4
Min Sludge Loading (1/day)	0.2	0.2	0.2
Max Sludge Loading (1/day)	0.5	0.5	0.4
Max Vol. Loading (lb/1000 cu.ft day)	40	40	37.5
Min Vol. Loading (lb/1000 cu.ft day)	—	30	18.7
Maximum MLSS (mg/L)	3000	—	3000
Minimum MLSS (mg/L)	1000	—	1500
FST Max Hyd. Loading (gpd/sq.ft)	1200	800	
FST Max Solids Loading (lbs/day sq.ft)	50	—	

HRT: hydraulic retention time;

10 State: the Great Lakes and Upper Mississippi River Basin States Design Standards (10-state standards);

V & H: Viessman et al. [1985] page 498.

Other constraints on the design are the performance criteria which are specified by the effluent standards. The loading rates probably have no intrinsic meaning by themselves but have been used as rules of thumb by engineers. The volumetric loading generally recognizes that oxygen transfer in the aeration basin becomes limiting when the aeration density becomes high. Sludge loading appears to influence the dominant organism type (filamentous vs. floc-forming) in the basin. To illustrate the decision making flexibility which remains for the engineer, the Ten-State Standards recommended loading rates were used with design conditions as shown below.

Flow = 10 MGD

Influent Soluble BOD5 = 150 mg/L

Influent TSS = 150 mg/L

The design variables used are the mixed liquor volatile suspended solids (MLVSS), aeration tank volume, and the final clarifier area. For a MLVSS of 1250 mg/L there is an acceptable aeration tank volume range of 2.9 to 7.3 million gallons with an associated annualized cost range of $950,000/yr to $1,100,000/yr (15% difference) (see Kao et al.[1989]). Design considerations such as cost, performance, and reliability can be used to narrow this range.

The analysis program can be used to solve the mass balances rapidly and to determine the cost for a given design. The first step to the problem is to formulate the influent conditions and effluent requirements for which the plant must be designed. Next, the average influent conditions are used with acceptable average loading criteria to determine unit process sizes. Those loading criteria may come from applicable State or regional design standards or from the consulting firm's company policy. Next the performance of the resulting design (specific unit sizes) should be checked under peak loading and adverse temperature conditions. From this point on, the user can iteratively delete processes or change unit process sizes to see the effect on cost and performance by the computer aided system.

5.3. Computer Aided System

5.3.1. General

The prototype computer aided system is an interactive system. Although the following demonstration illustrates how the system works, it is much easier to understand from a videotape or a live demonstration. The general characteristics of the prototype have been discussed in Chapter 3. The following discussions focus on the characteristics specific for a WTPD model and a demonstration of the general and specific characteristics.

As mentioned in Chapter 1, a wastewater treatment plant design problem is usually complex. The complexity is caused not only by the mathematical difficulty of obtaining a numerical solution, but also by the presentation of the design data, manipulation under different design conditions, generation of potential alternatives, and interaction in a trial-and-error or solution selection procedure. The goal of the prototype is to provide an efficient, accurate, creative, user friendly, and easy-to-use system for use in the design of wastewater treatment plants.

The prototype was first developed on an IBM PC AT. Since the IBM PC AT has limited capacity and screen resolution, the PC version was complex to use. The PC version was thus converted to an Apollo workstation environment. The user interface of the prototype on the Apollo workstation now is much simpler and easier to use.

For data entry, data must be manipulated in the prototype to define the problem which is to be solved and to describe performance and constraints for the unit processes. The approach taken here is to allow the user "form fill-in" of the table displayed on the screen. Such a format allows the user to make changes quickly to a data set. When the user is confronted with a table of data, the entries may be changed by directly typing in the new value on the corresponding input field.

A variety of process performance models are available in the prototype. This flexibility is considered important for allowing the engineer to explore the impact of research results or specific plant operating data on the design and performance of a plant. Those performance models are presented to the user by way of a two-dimensional plot of the performance parameter (solids concentration, fractional removal, etc.) vs. a significant design parameter (overflow rate, underflow rate, etc.). All of the available models are plotted on the same scale. The user selects one of the models, that curve is highlighted, and an abstract of information (under what conditions it was developed, the equation form of the model, etc.) is presented. Whichever model is selected when the user leaves the selection menu is chosen for the analysis. This presentation approach works well, and the presentation of the available models together on

the same plot provides interesting comparisons. However, many of the models have more than one dependent variable and thus the plot does not tell the whole story. For example, the overflow solids concentration model for the final clarifier may depend on overflow rate, unit feed rate, and/or feed solids concentration. A two-dimensional plot requires one of these variables to be fixed.

Model selection screens will display the curves of the models available to specify process performance and a menu of model authors. The currently chosen model will be highlighted on the graph. A short description of each performance model is presented when that model is highlighted on the screen. The description includes the model's equation and the position of the equation parameters (indicated as c1, c2, c3, ...).

A number of checks are made while the problem is being solved. For example, the aeration tank volume is determined as the minimum volume that satisfies the maximum loading and minimum detention time values. If the volume violates the minimum loading rate or maximum detention time, there is no solution for those design conditions. In instances where the design proves infeasible, or a violation of a design condition or standard has occurred, a warning message will be shown on the screen.

The data that are used to formulate a problem, processing scheme, performance models, etc. may be stored and recalled as designs. The number of designs that can be stored is limited only by the memory capacity on the workstation used.

The cost equations have been modified to utilize a generic function. Generally, the cost equations are piecewise non-linear curves of the form:

$$COST = a\ (X)^{b}$$

where a and b are modeling parameters and X is the relevant sizing variable. The program allows up to five curve segments for each cost function. Each curve segment is defined by specifying its upper bound (the lower bound is the previous segment's upper bound or zero), a, and b. The cost parameters are loaded into the program from a data file and they may be modified without changing the source code.

The display of cost curves is very important for a designer to size a process or determine capacity of a facility unit. However, as mentioned in Chapter 3 the display of cost curves is not easy. For the PC version, the cost curves are displayed by semi-log plots. Although a semi-log plot can cover the whole range of parameter values, it is hard to see the approximate value of cost. A semi-log curve gives only the shape of curve and is not much help for a designer in evaluating the cost region of interest. For the Apollo version, a normal scale plot is used to show the cost curve. By using the normal scale, the cost curve can be more easily understood and used by the designer. The range of parameter values to be

displayed can be provided as input by the designer in two data fields, one for a reference point and the other for a spanning range. The designer can select a desired range by entering values into the two fields, and a cost curve, centered at the reference point and span backward and forward by the span range specified, would then be shown with the cost associated with the reference point. The flexibility of showing different ranges of the cost curve and the cost for a particular parameter value is very useful for the designer to select an appropriate design under a cost constraint.

The program's interface is written with DIALOG. The user controls program flow using the computer's mouse. Menu options are presented as a set of boxes on the screen. The user moves the mouse cursor into the box of the option desired and clicks the left button. This will activate that capability. When popup windows appear, they can be deactivated (popped down) by clicking the middle mouse button. A general description of the program structure is provided in Appendix B.

Capabilities

The following capabilities are currently functional on the Apollo workstations:

- construct interactively activated sludge models of any combination of a given set of unit processes and solve the mass balances and find the cost and the likelihood of bulking for that treatment scheme.
- specify a processing scheme graphically;
- specify unit process sizes;
- change interactively baseline model parameters and plant design conditions (flow, waste strength, etc.);
- view details of mass balances throughout plant and details of system capital and O&M costs with data presented in either a tabular or graphical format;
- display output data graphically;
- save and load an unlimited number of design cases;
- receive further explanation regarding the values of model parameters and the conditions under which they were developed.

These capabilities are demonstrated in the next subsection.

The following description presents features of the prototype by graphical demonstrations. The demonstrations simulate the interactive environment. However, as mentioned above the system is easier to understand from a videotape or live demonstration.

Introduction

Upon initiation of the prototype, an introduction is displayed as shown in Figure 5.2. After any mouse button is pressed, the program options which are present in that full environment are briefly explained below. In the initial screen shown in Figure 5.3, a default set of unit processes is shown. The unit processes shown on the screen provide an interface for various manipulations, e.g. configuration, editing parameters, and examining results. The designer can simply click the mouse button on the desired unit process to make a selection and then manipulate any necessary action. Although the prototype so far does not allow the designer to add interactively an arbitrary number of unit processes, this default group can be reset to any set of unit processes presented. The default set gives most unit processes used for an activated sludge system; it should be suitable for most cases.

The top row of menu options is for editing design parameters, selecting flow models and design approach, solving a design, reporting likelihood of bulking, showing cost figures and related information, and quitting the program, respectively. These options and those which are going to be presented can be easily selected by clicking the mouse button on the desired menu item. The ability to make selections without typing from the keyboard is one of the major characteristics of the friendly interface. This row of options controls the major activities in the design session and are demonstrated in more detail after Figure 5.8.

The second row is a message area which is used to report a short response, warning, or error messages. If the message is long and/or important, a popup window will be shown to bring it to the designer's attention instead of showing it on the message area. The messages, message area, popup window, and beeper establish the feedback system to avoid mistakes made by the designer and guide the designer in exploring good alternatives.

The third row of options is for configuration of the process schematic. The first option is a name field where the designer can type a name. If the design name exists in the database, then the design will be opened. Otherwise, a default set of unit processes without any linkages will be shown. An exception is that the activated sludge and final clarifier are treated as an individual process, and the recycle flow linkage cannot be changed. The name provides an identification for a design. The next options in the third row are used to show a list of names of created designs, re-configure a design, fix a configuration, and

select flow types (under or over flow), respectively. Below the three rows of options is the working area to configure a process schematic. After a schematic is configured, it serves as an interface for manipulating related information of the schematic.

The design procedure used in this prototype has two steps. In the first step the configuration is "free" to have changes or modifications in its linkages. In the second step the configuration (or process schematic) is "fixed." There are two reasons for using the "fix" and "free" options: (1) to avoid confusion in using the configuration because the configuration is also used as an interface for other tasks; and (2) to avoid inadvertently changing a configuration in editing or doing other tasks because any configuration change would change the set of parameters and model equations. Options are also grouped based on the condition (free or fixed) of the current configuration. The free group of options includes Re-Configure, Fix Configuration, and Under- and Over- flow, and the fixed group of options is the top row of options except Quit. If working on a free configuration, the fixed groups of options will be deactivated and will not respond to the designer's selection; and vice versa. This limitation reduces the chance of inadvertently selecting an undesired option by the designer. In Figure 5.3, the configuration named New is free and the fixed group of menu options is deactivated. The texts associated with deactivated options are turned gray, so the designer can distinguish them from active techniques.

Create, Configure, and Open a Design

Creating a new design is easily done by typing a new design name in the name field. After typing a name, the default set of unit processes would be shown. On the screen with the default set shown, by interactively clicking the mouse button on a process and drawing the flow lines to another process, a process schematic can be configured. Although a designer can create a process schematic of any combination of unit processes, the schematic may be infeasible based on design constraints or mass and flow balance conditions. In Figure 5.4, a process schematic would be set up after drawing, by moving the mouse, the underflow line from the final clarifier to the gravity thickener.

After creating a design, the designer should fix the configuration by selecting the "Fix Configuration" option. A fixed configuration is not allowed to change. In Figure 5.5, the design New is fixed. The fixed group of options is then activated and the free group is deactivated.

The designer can open an existing design by typing a name such as "test" in Figure 5.6 into the name field. The desired design is then opened. Note that the free group of options is deactivated because the 'test' design is an existing design and therefore fixed.

The designer can open an existing design by typing a name such as "test" in Figure 5.6 into the name field. The desired design is then opened. Note that the free group of options is deactivated because the 'test' design is an existing design and therefore fixed.

The designer can construct a different design by typing a new name into the name field and by using the procedure described (see design "partial" in Figure 5.7). A list of design names can be shown by selecting the option 'Configuration List' for review (see Figure 5.8).

The interactive approach of using the mouse to specify flow linkages is very convenient for setting up a process schematic. Each time the designer types a new name, a new design is created. After the option "Fix Configuration" is selected, the design will be stored in the computer memory. Those unit processes which do not have any linkage to or from any other process(es) will be automatically deleted. Thus, no options are needed to SAVE and DELETE individual processes.

View and Edit the Design Parameters

The characteristics of design parameters and results obtained from solving a design are shown in the forms shown in the next figures. A form can be selected for viewing and editing by clicking the mouse cursor on the desired position in the process schematic. For example, by clicking the "influent," the form for the feed characteristics will be displayed as shown in Figure 5.9. The editable fields are highlighted by using bold character display (see the number on the right of the form shown in Figure 5.9). To edit, the designer can click the mouse on a desired field. Then, a small triangular cursor will be shown to indicate the typing position (see the second number, 100.00, in the form shown in Figure 5.9), and the designer can then enter a new number. Since each field is self-explained, no more explanation for each field is provided in the following descriptions. Figures 5.10, 5.11 and 5.12, 5.13 and 5.14, 5.15 and 5.16, 5.17, and 5.18 show the input and output forms for the primary clarifier, activated sludge system, gravity thickener and secondary digester, primary digester, vacuum filter, and effluent conditions, respectively. The input (editable) fields are highlighted by bold character display. The output fields are those numbers which are not highlighted; it shows the information which is not editable and only for reference purpose. The output fields are discussed again later after the option Solve is introduced.

Since the information for some unit processes exceeds the display capacity of the screen, it is divided into two separate windows, one containing output and frequently used input fields and the other containing the less frequently modified parameters (see Figure 5.11 and 5.12, 5.13 and 5.14, and 5.15 and 5.16). The second window can be shown by selecting a menu option on the first window. By the

"form-in" approach demonstrated above, the design parameters related to a specific unit process can be easily examined and modified.

Flow Models

A variety of process performance models from the literature is available in the prototype. The performance models can be displayed by first selecting the option "Flow Model" and the desired flow type (under- or over flow), and then clicking the mouse on the desired unit process. For example, Figure 5.19 shows a popup window in which two available underflow models for the primary clarifier are displayed and the Dick model is selected with the description and the associated curve highlighted. The selection can be made by clicking the mouse on the checkboxes provided on the right of the popup window. Various performance models for different unit processes are shown in Figures 5.20, 5.21, 5.22, 5.23, 5.24, and 5.25.

Solving and Results

After the values of all design parameters are determined and all performance models are selected, the designer can solve the design model by selecting the option "Solve" on either the top row or the top right corner of an editing window. Although the Solve option is duplicated, the one in the editing window makes it easy to see the changes immediate after some modifications are made to the design parameters; this interactive ability has been found to be very useful for exploring alternative designs. For example, Figure 5.26 shows a solution with the cost of $2,190,266, shown at the bottom of the output form. The designer may want to change the value of a design parameter. For instance, the maximum of sludge loading may be changed from 0.5 to 0.45 lb BOD_5/lb MLVSS day. A new solution can be obtained by re-solving the design (see Figure 5.27). The new solution has the cost of $2,191,346.

After creating a feasible design, the designer may want to examine the likelihood of experiencing sludge bulking. By selecting the Bulking option on the top row, a popup window for the probability of bulking based on the design conditions will be shown (see Figure 5.28).

One of two options can be selected to specify the method used for solve the design model: fixed process sizes or specified loadings. The two options are shown on the top row. Each option has a group of editable fields. If the fixed sizes option is selected, all editable fields related to the fixed loadings method will not be highlighted; and vice versa (See Figure 5.29).

Cost Information

mentioned, the plot may not be useful. The following figures demonstrate how to improve the presentation of the cost related information.

The cost related information includes a cost summary table, cost parameters, cost curves, and cost coefficients. First, the "Cost" option in the top row should be selected for showing the cost related information. Upon selecting the Cost option, a list of sub-options is shown as in Figure 5.30. The cost summary table can be displayed by selecting the sub-option "Cost Summary" (see Figure 5.31). Cost parameters, average wage rate, electricity cost, capital recovery factor, methane value, and sludge disposal cost, can be modified from a popup window shown after selecting the sub-option "Cost Parameters" (see Figure 5.32).

Five types of cost information are provided in this prototype: capital, operations, maintenance, supplies, and power. Other than specific unit processes, there are several other components in a design (e.g. return sludge pumping) that impact the cost. Instead of using the configuration as the interface for displaying cost curves, a list is provided and selections are implemented by the checkbox approach. Figures 5.33, 5.34, 5.35, and 5.36 show the capital, operations, maintenance, and supplies cost curves for primary clarifier, and Figure 5.37 shows the power cost for return sludge pumping.

On the curves, the cost associated with the value of a design parameter in the current solution is indicated by a circle; the parameter value is shown in the field 'Ref. Point=>'; and the cost is shown in the message area. The field 'Span' indicates the range around the current value to be shown. And the fields 'Coeff a' and 'Coeff b' are the cost coefficients for the cost function associated with the displayed range. For example, Figure 5.38 shows the capital cost curve of return sludge pumping with pumping capacity = 72.39 cu.m/hr indicated and the cost shown in the message area. The value of pumping capacity can be changed and the new cost is indicated and shown. (see the field 'Ref. Point=>', the circle on the cost curve, and message area in Figure 5.39). The span range and cost coefficients can also be changed as shown in Figures 5.40, 5.41, and 5.42, respectively.

Design Violation and Standard Processing Report

After a design is solved and the solution has been checked against the design conditions or standards to determine if any violation occurs, a popup window with a warning message would be shown to tell the designer of violation(s). Although only one figure, Figure 5.43, is used to illustrate this ability, this kind of warning message may frequently appear in a real design session by checking feasibility and using

kind of warning message may frequently appear in a real design session by checking feasibility and using the standard processor described in Kao et al. [1989]. This feedback capacity is very important for guiding the designer to the design of a sound plant.

Comparison

In Figure 5.44, the design "partial", in which no primary and secondary digesters are used, is solved and the cost summary is shown. By comparing the cost table with the one shown in Figure 30, the effect of deleting secondary and primary digesters on the cost can be observed. Also, the duplicate "Solve" option can be used to compare the results obtained from using different values of a design parameter. This comparison function is primitive, and a much better comparison mechanism is demonstrated for the GRM prototype.

5.4. Summary

This chapter presents the features of the prototype developed for a WTPD problem. The prototype monitors actions that the designer selects and performs the action selected. The menu options are designed to be as simple as possible. Most options are shown in one window and are laid out to avoid complexity in selecting menu options. The deactivation of unnecessary options reduces the chance of inadvertently choosing undesired options. For the PC version of the prototype, more levels of popup windows are needed and thus complexity increases. To avoid confusion, at most two levels of popup windows are shown on the screen for the Apollo prototype. The second level of popup windows shows parameters which are infrequently modified.

The Solve option is provided in each editing screen, and the designer can see a new solution immediately after changes are made. Since the interactive response time is quick, the conventional trial-and-error procedure can be used efficiently. The ability of the prototype to solve a mass balance on virtually any unit process combination is a great aid to the designer when searching for a good system design or when he wishes to put together a model of a processing scheme quickly.

The feedback system provided in the prototype is intended to guide the designer in a design session. This prototype is expected to aid a process designer in shortening the time for producing a feasible design and to provide functions to assist the exploration of better designs. The designer can take into account other issues and modify the design by trial and error.

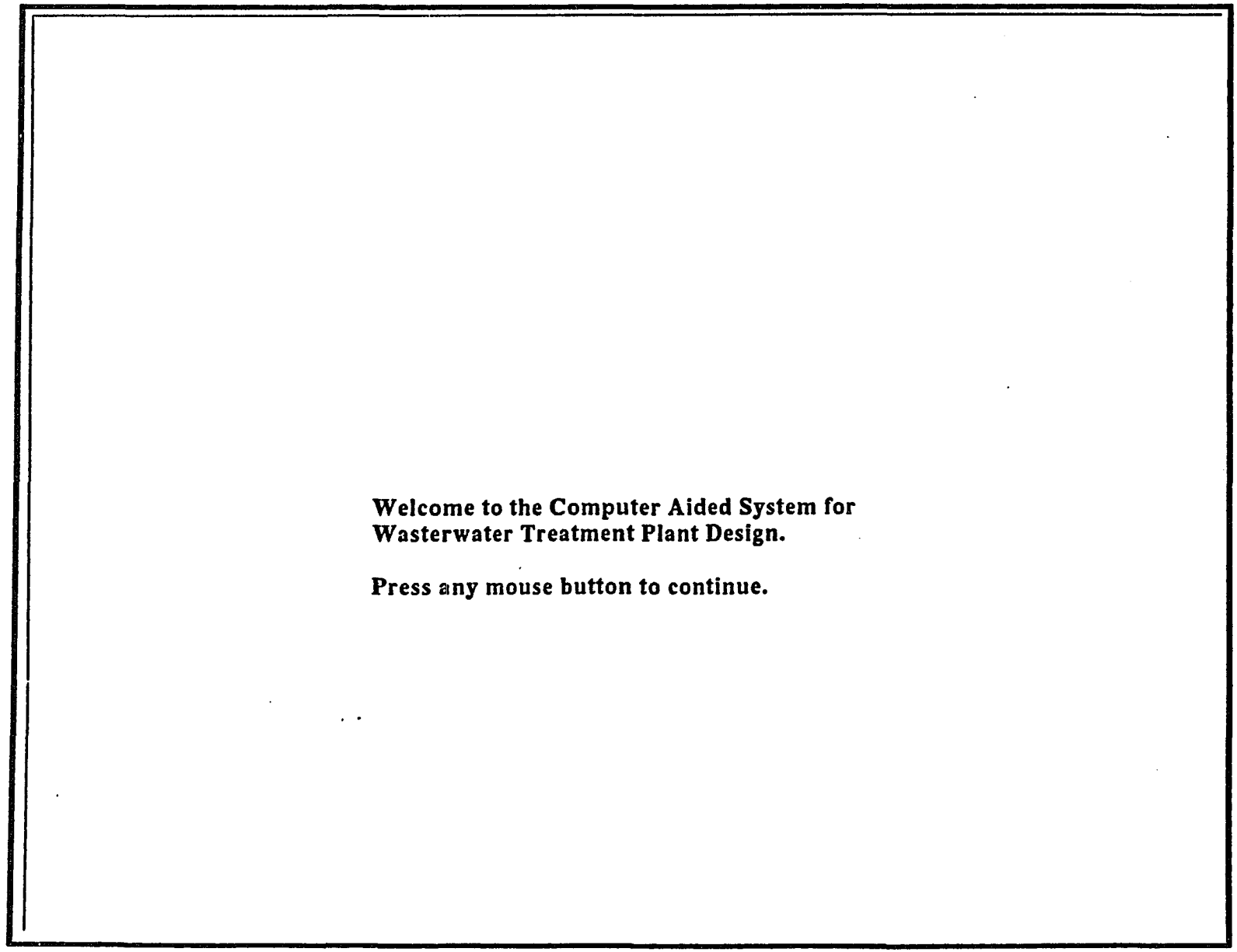


Figure 5.2

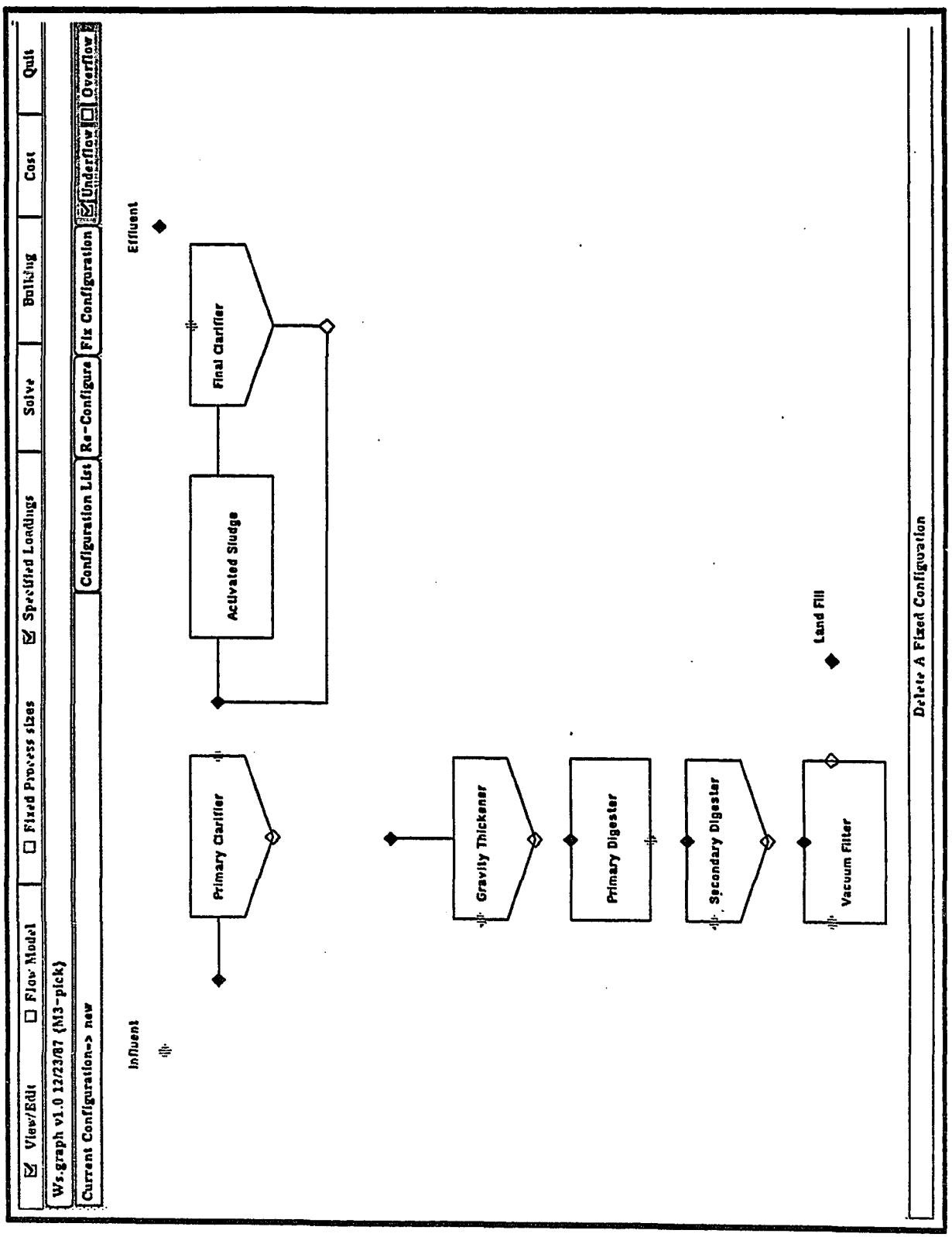


Figure 5.3

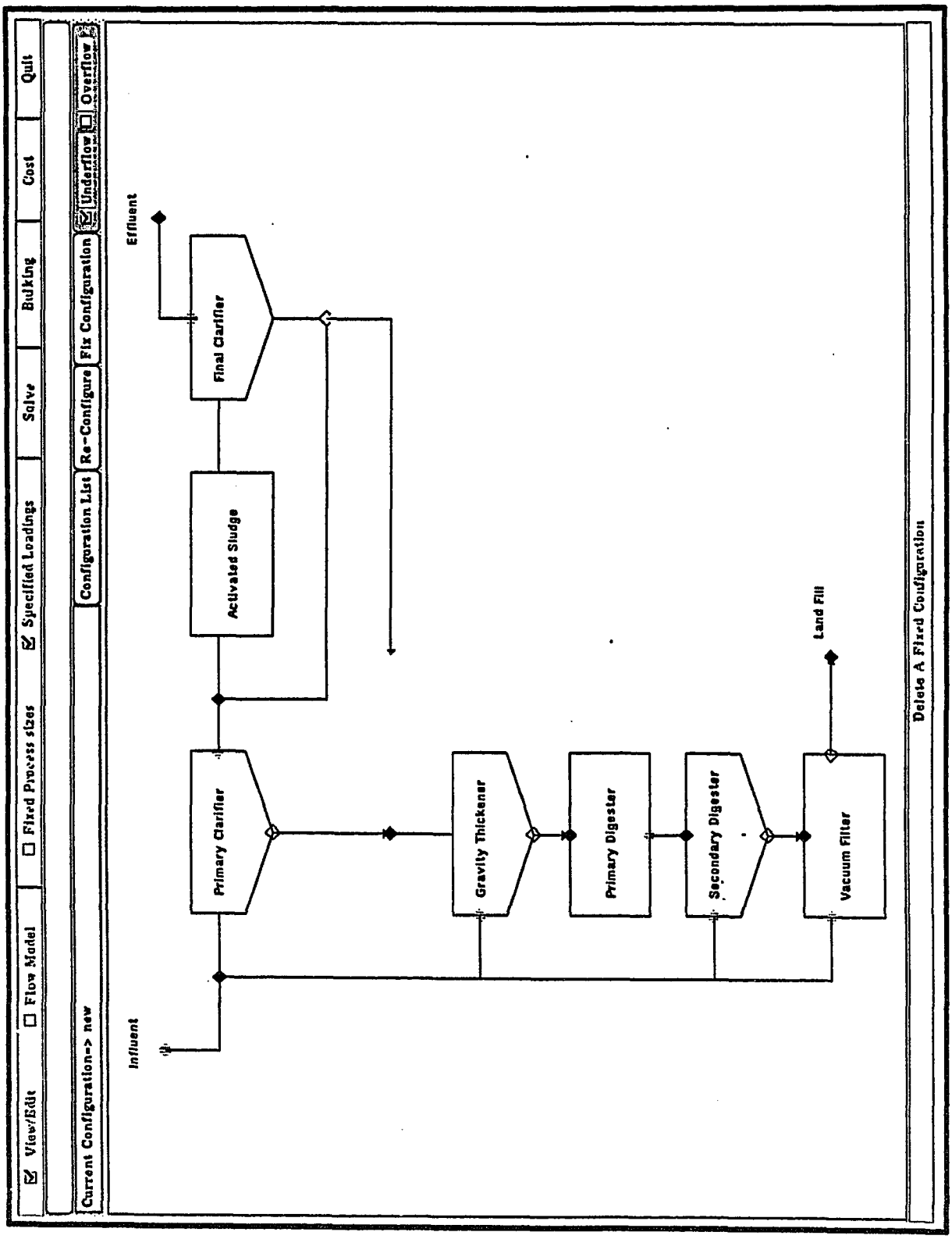


Figure 5.4

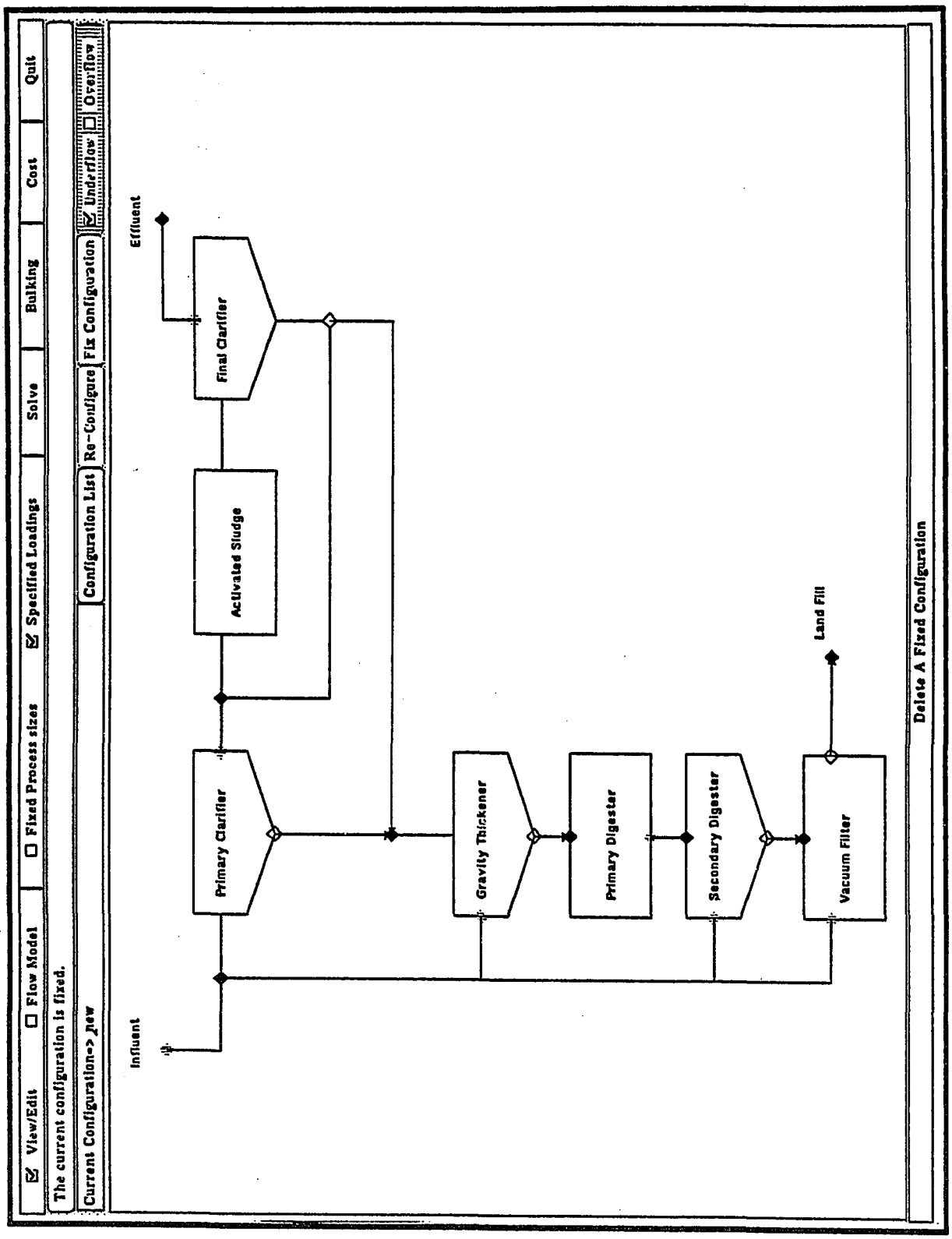


Figure 5.5

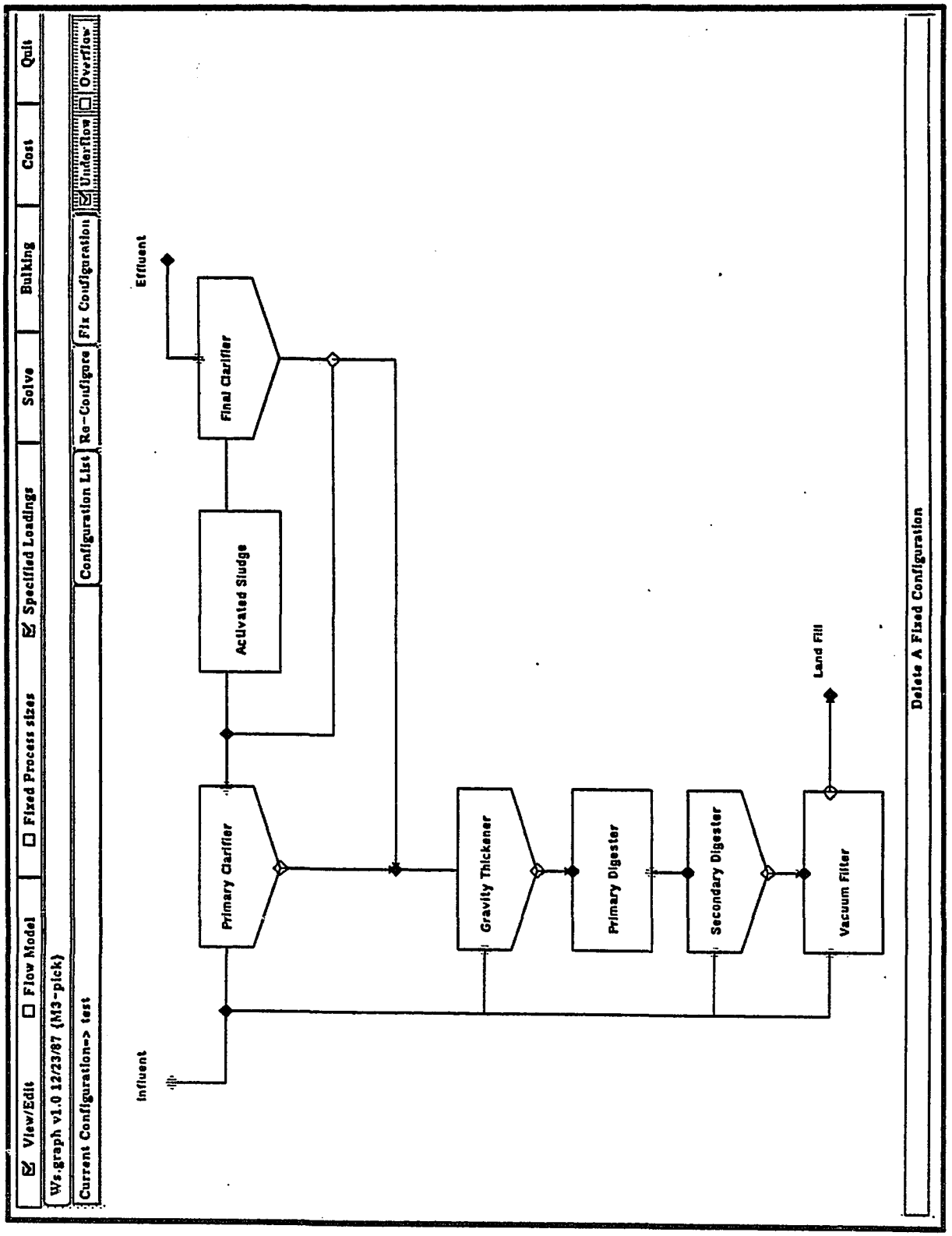


Figure 5.6

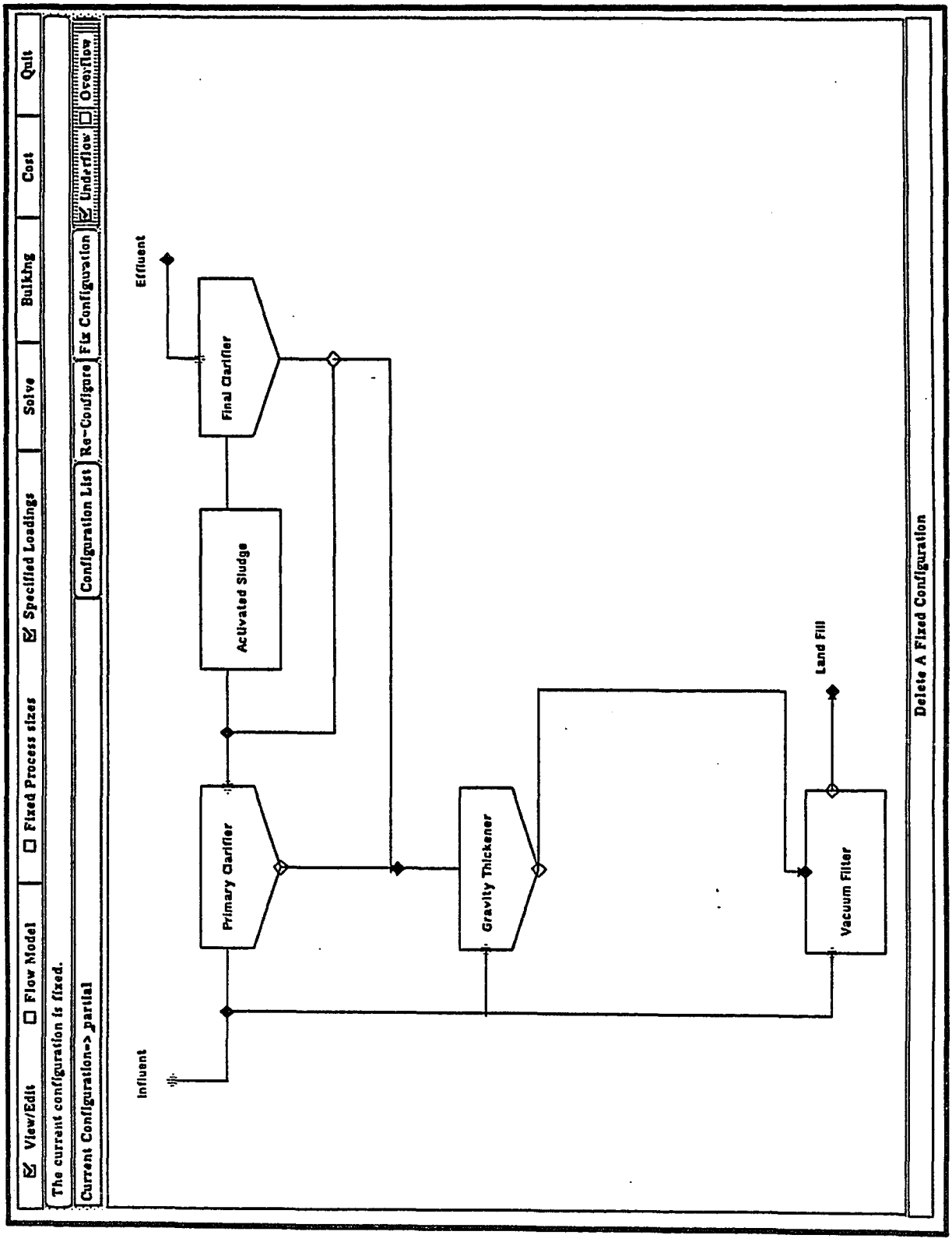


Figure 5.7

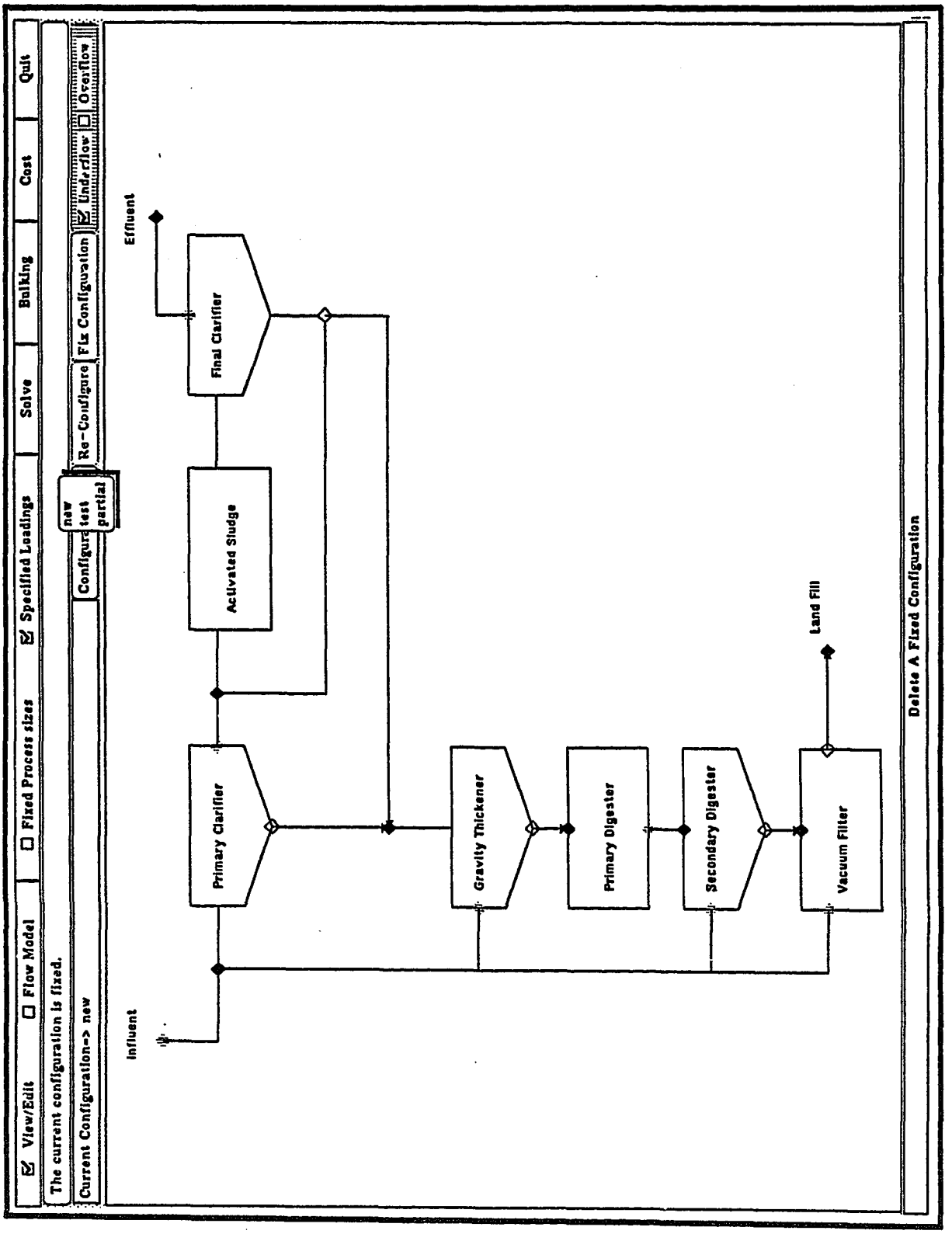


Figure 5.8

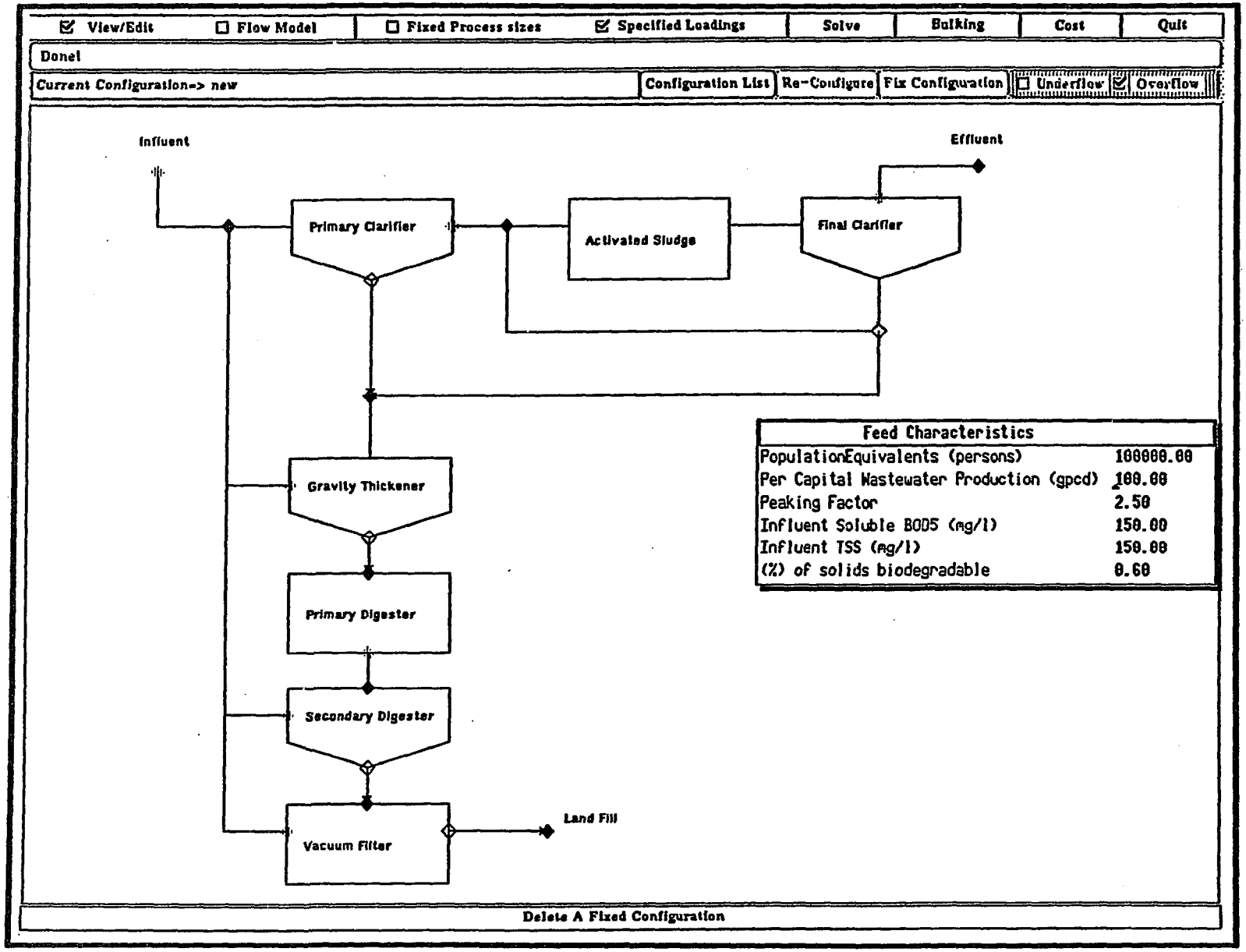


Figure 5.9

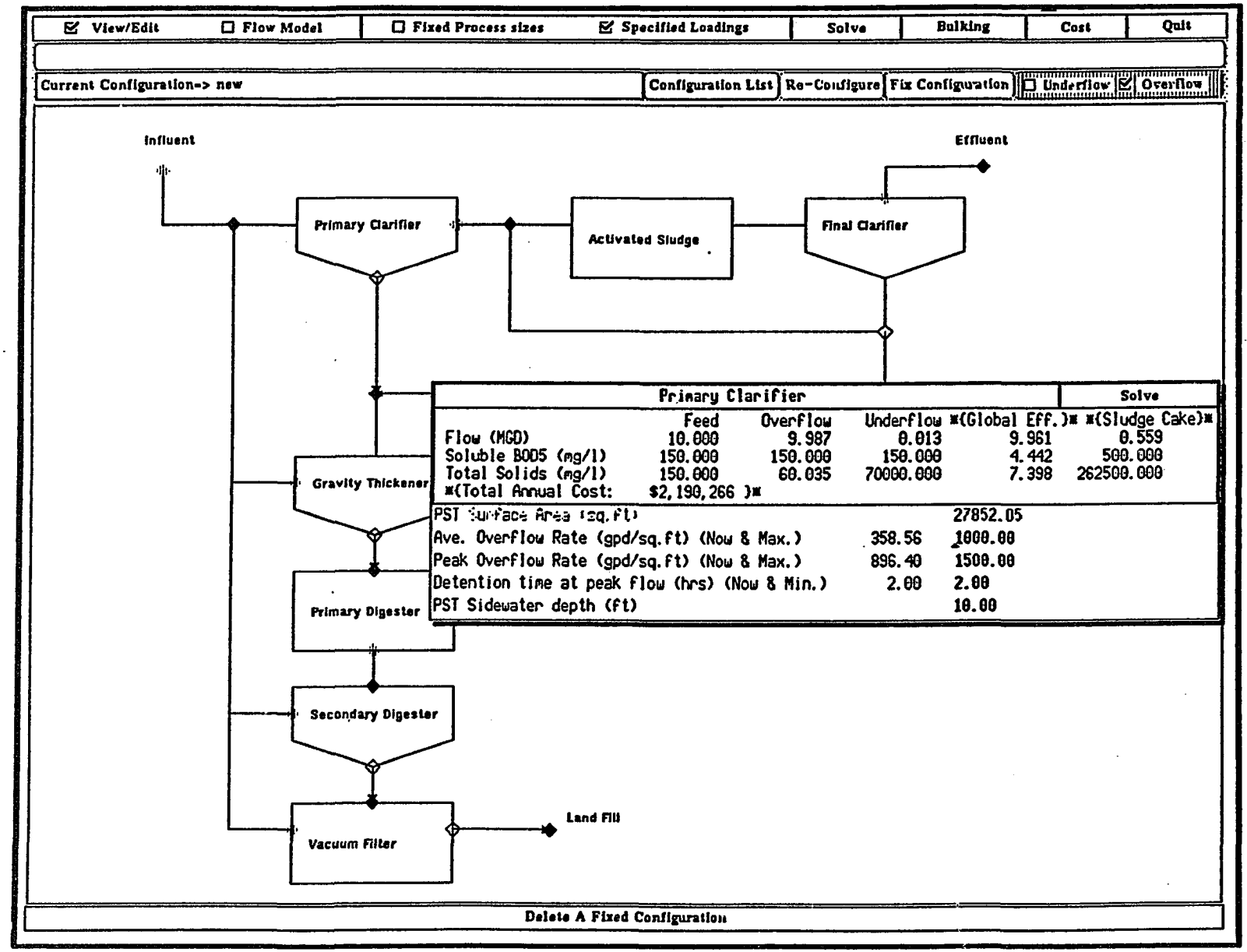


Figure 5.10

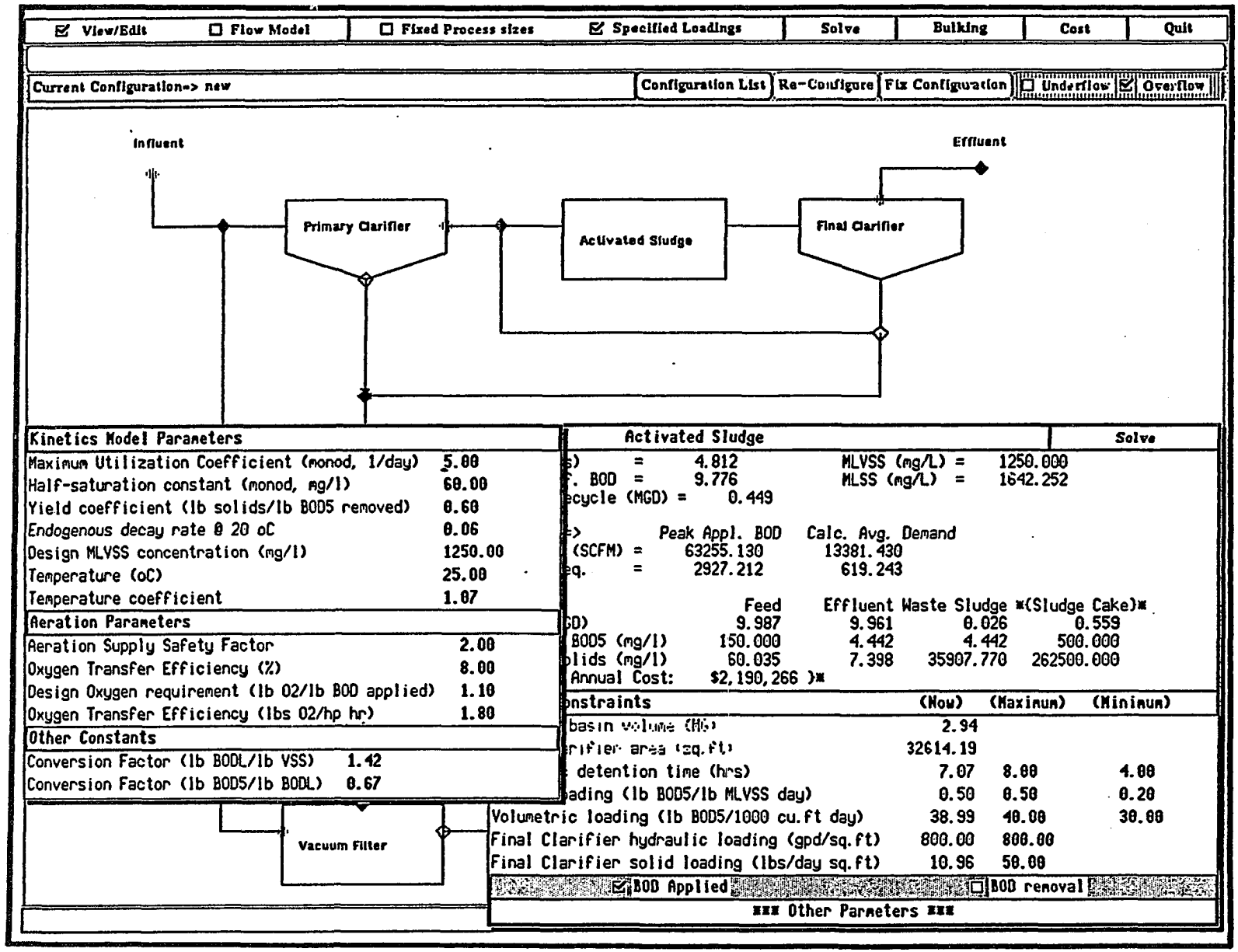


Figure 5.11

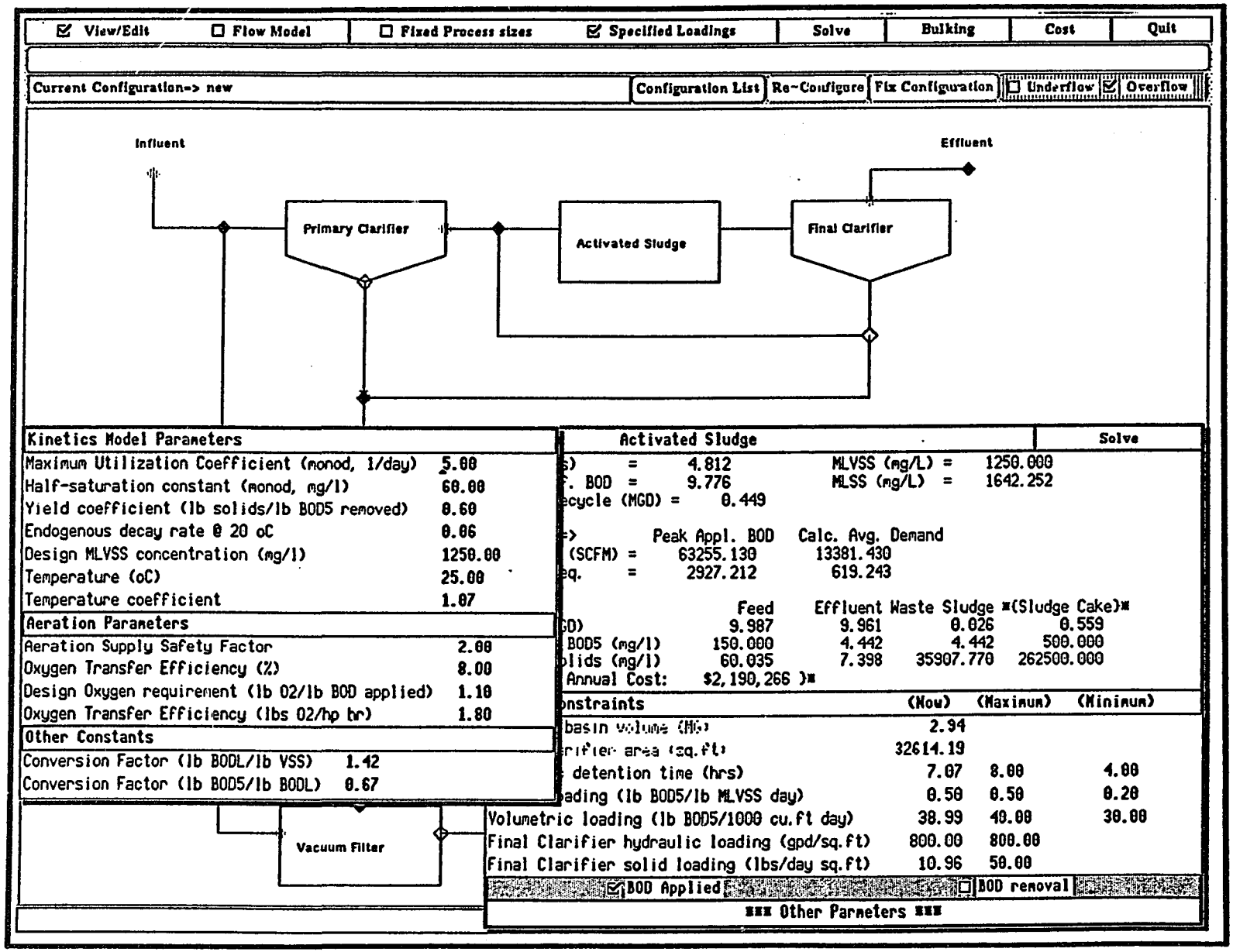


Figure 5.12

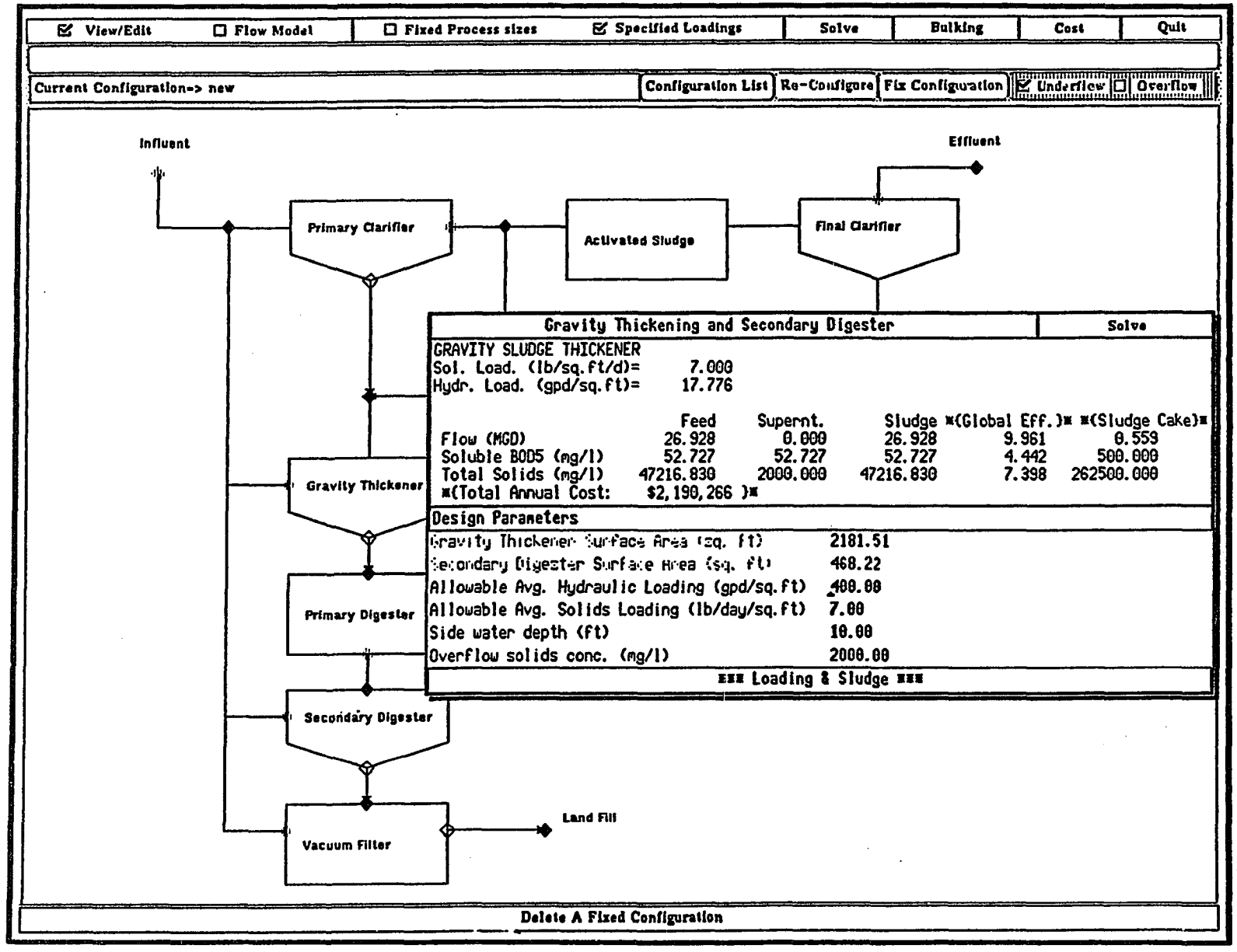


Figure 5.13

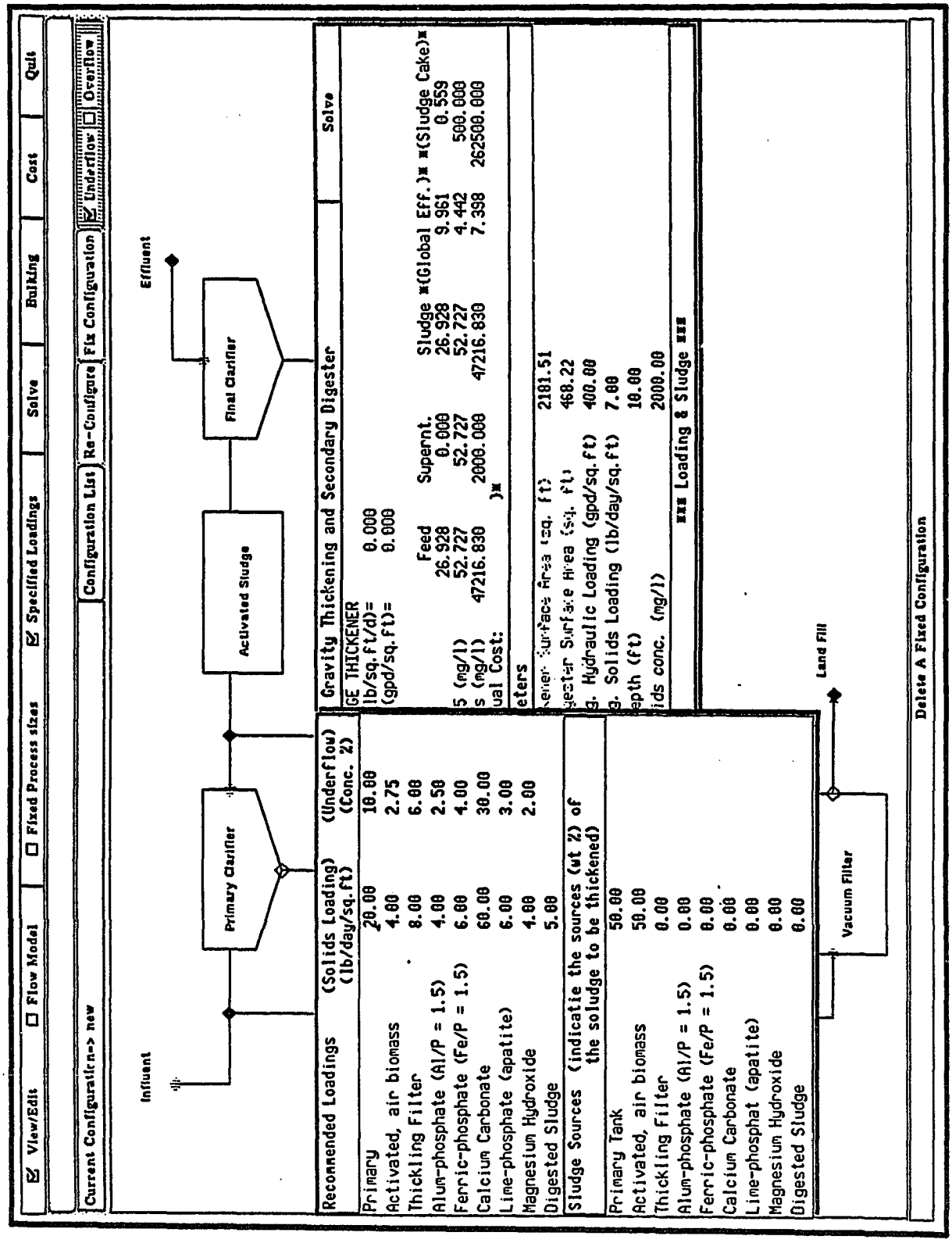


Figure 5.14

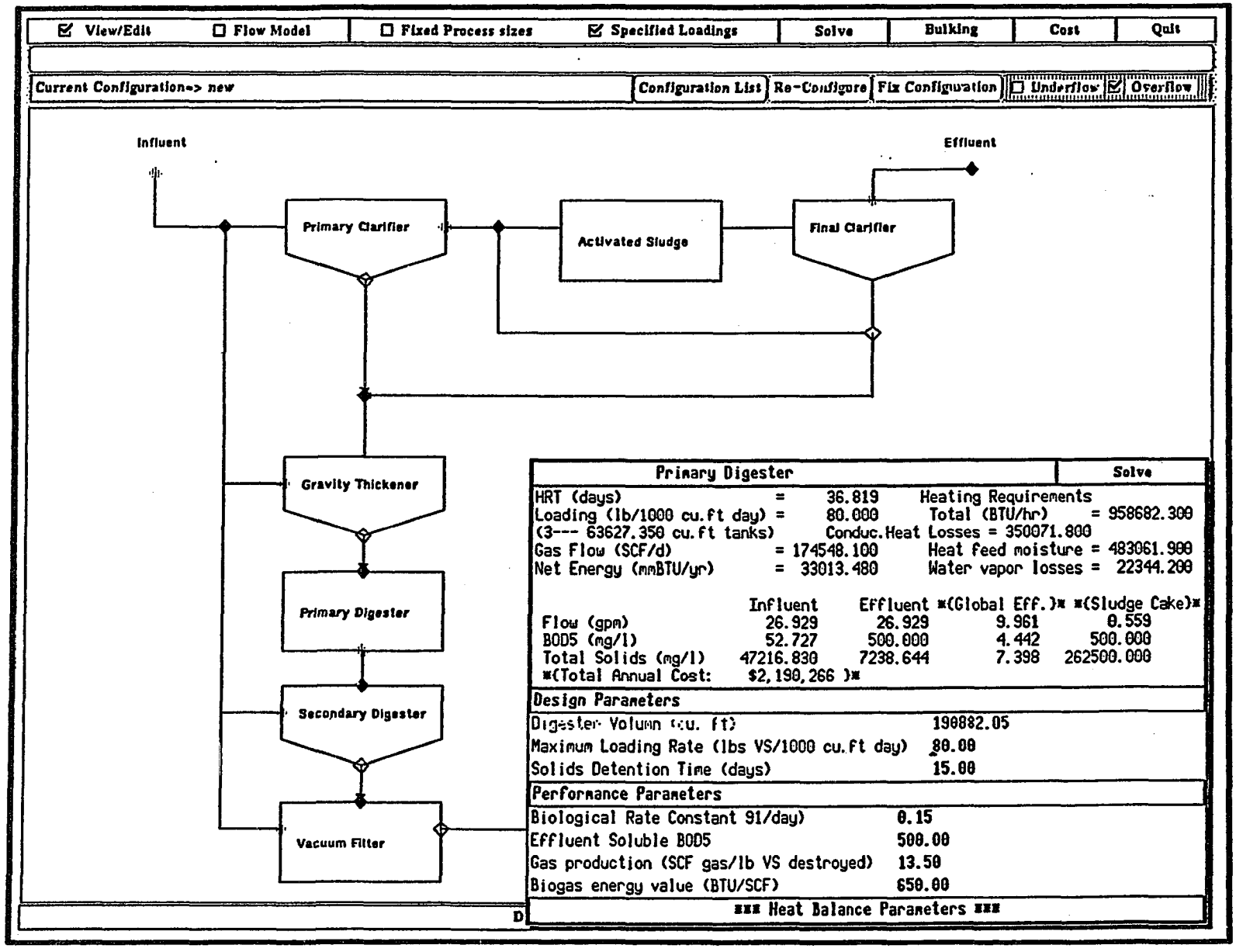


Figure 5.15

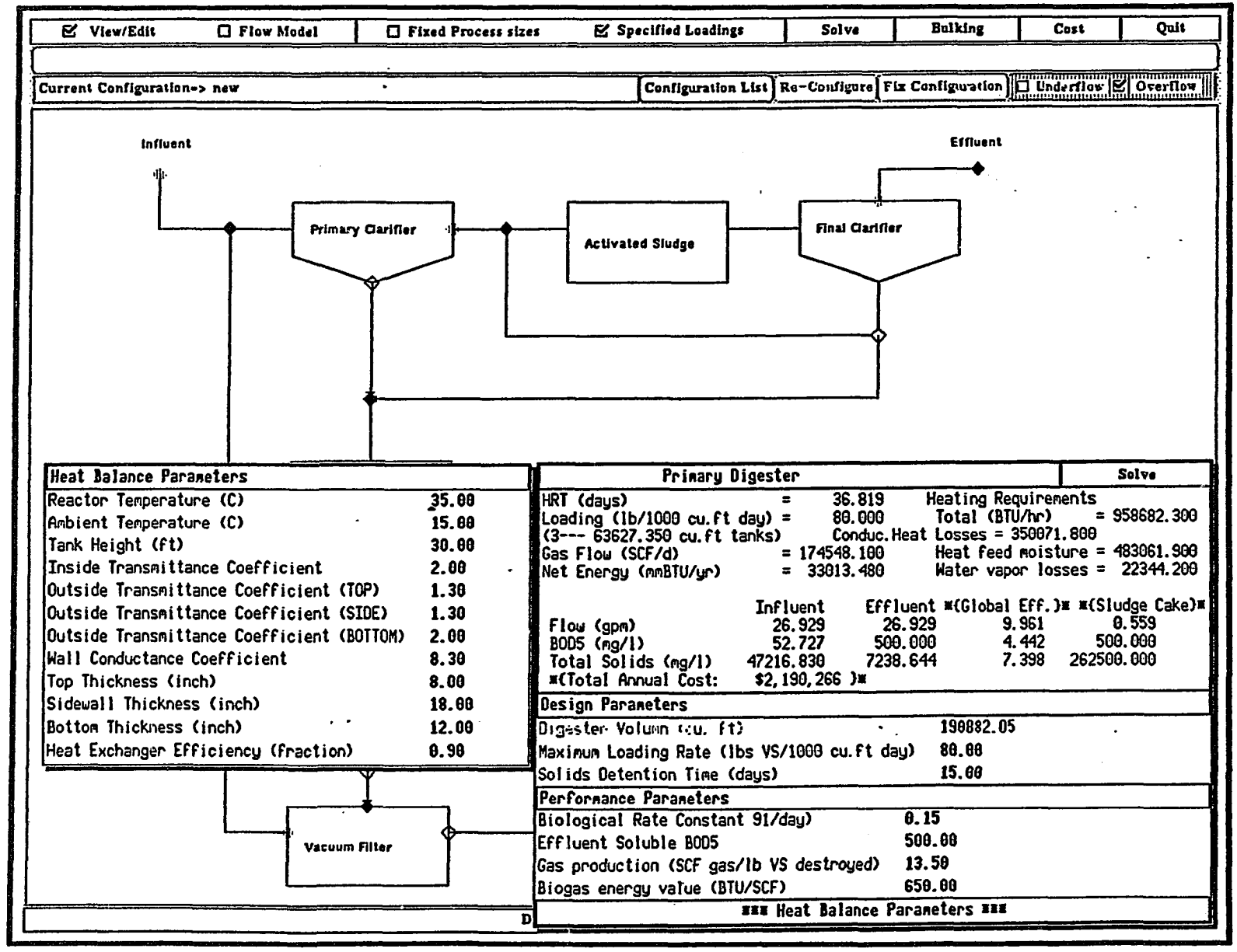


Figure 5.16

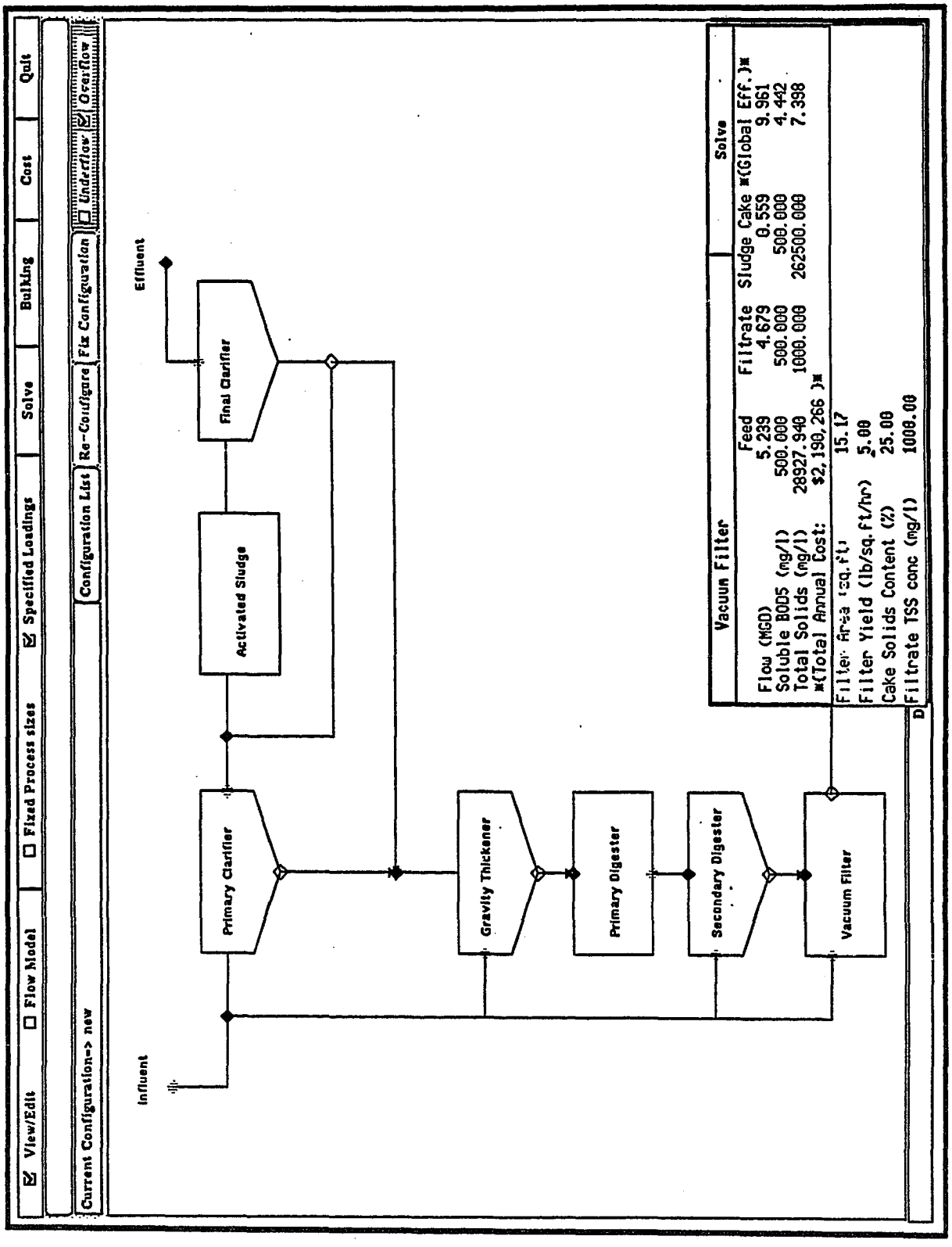


Figure 5.17

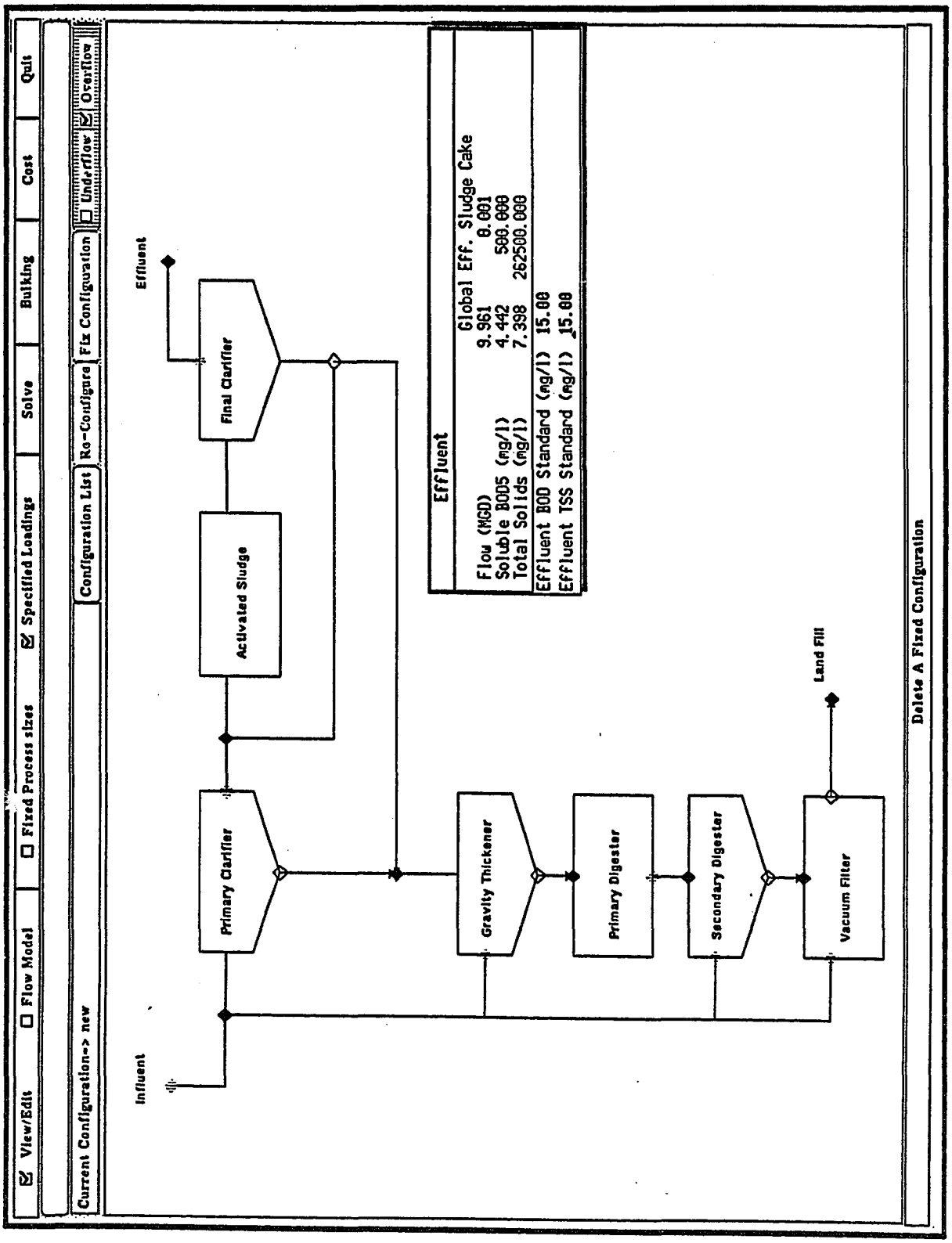


Figure 5.18

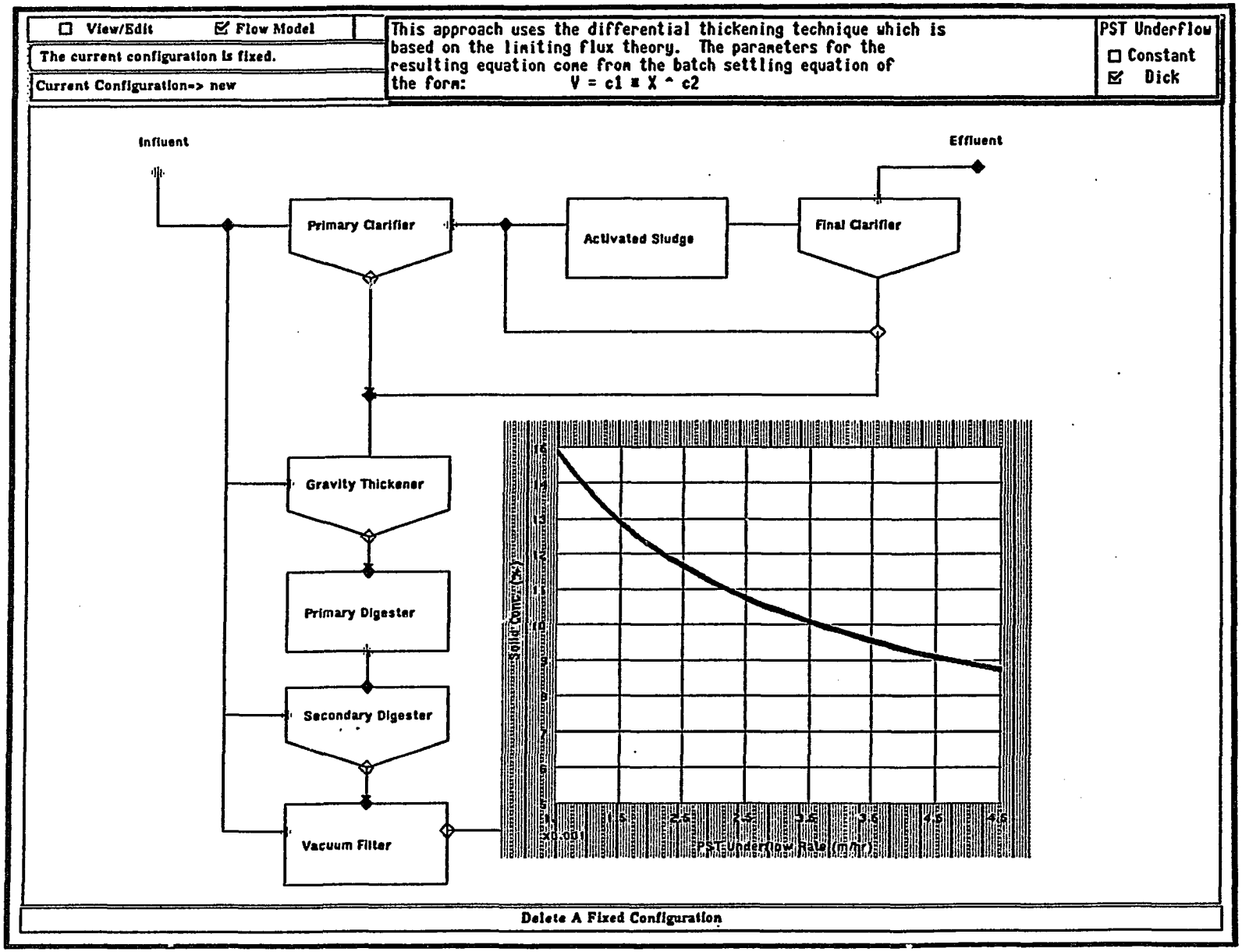


Figure 5.19

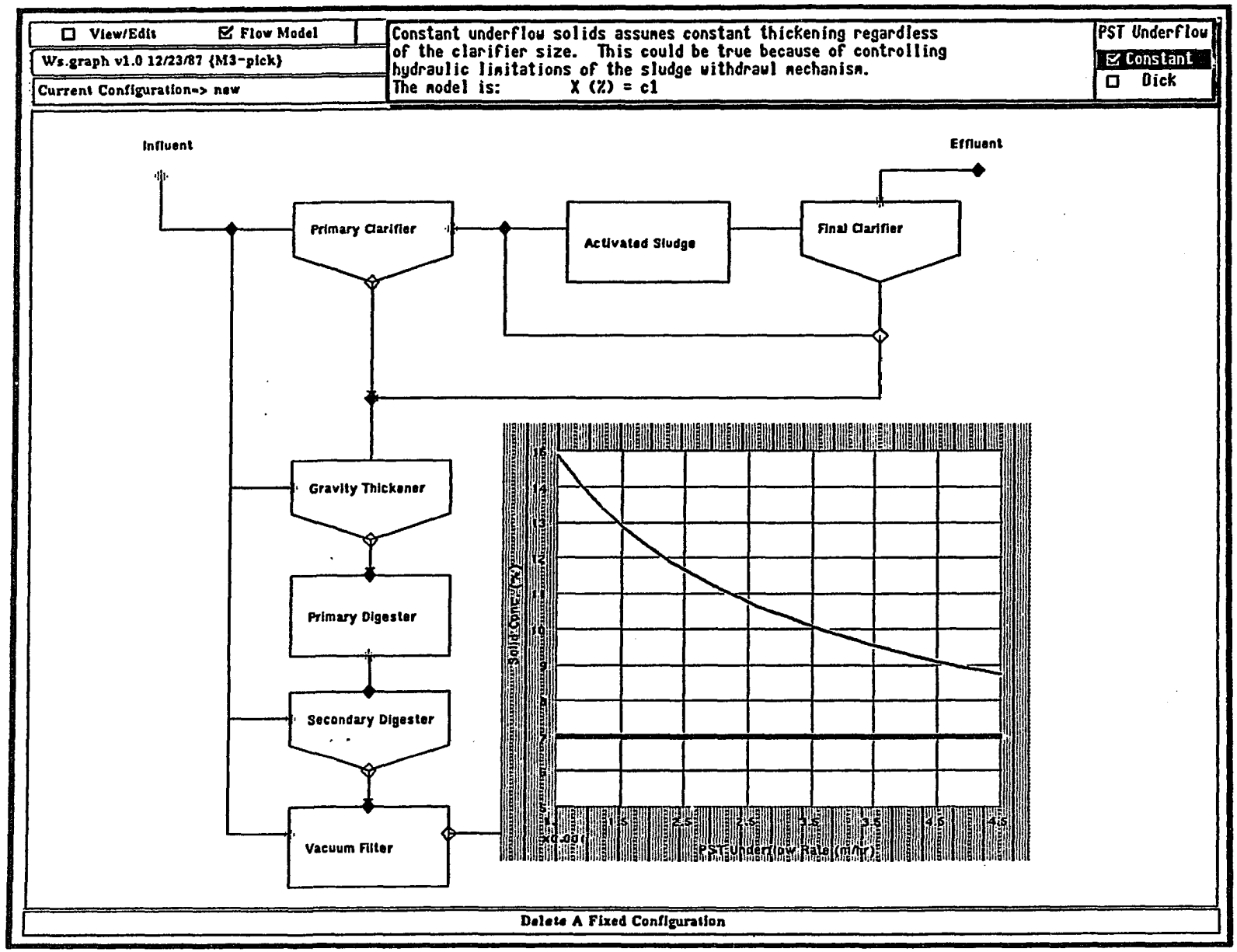


Figure 5.20

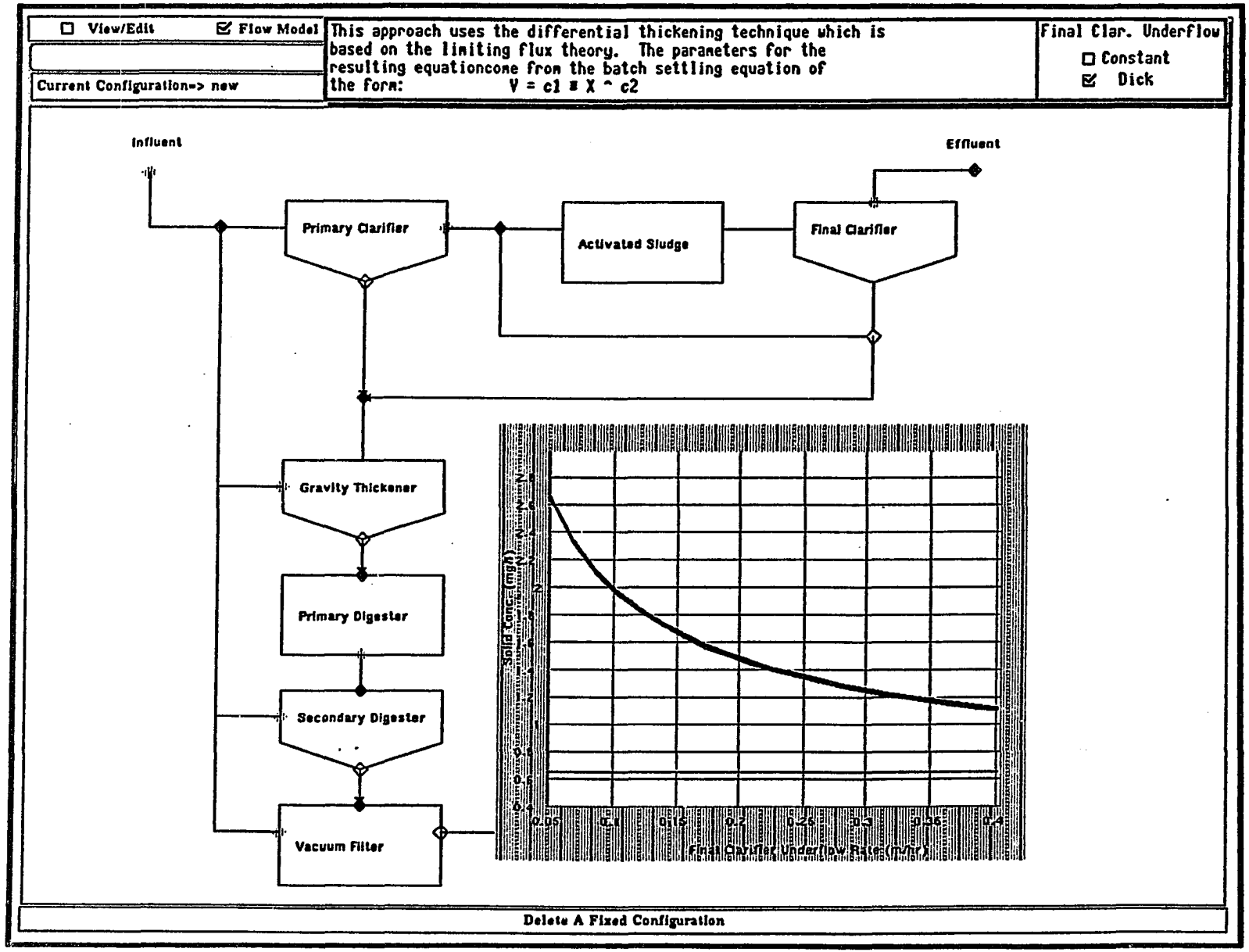


Figure 5.21

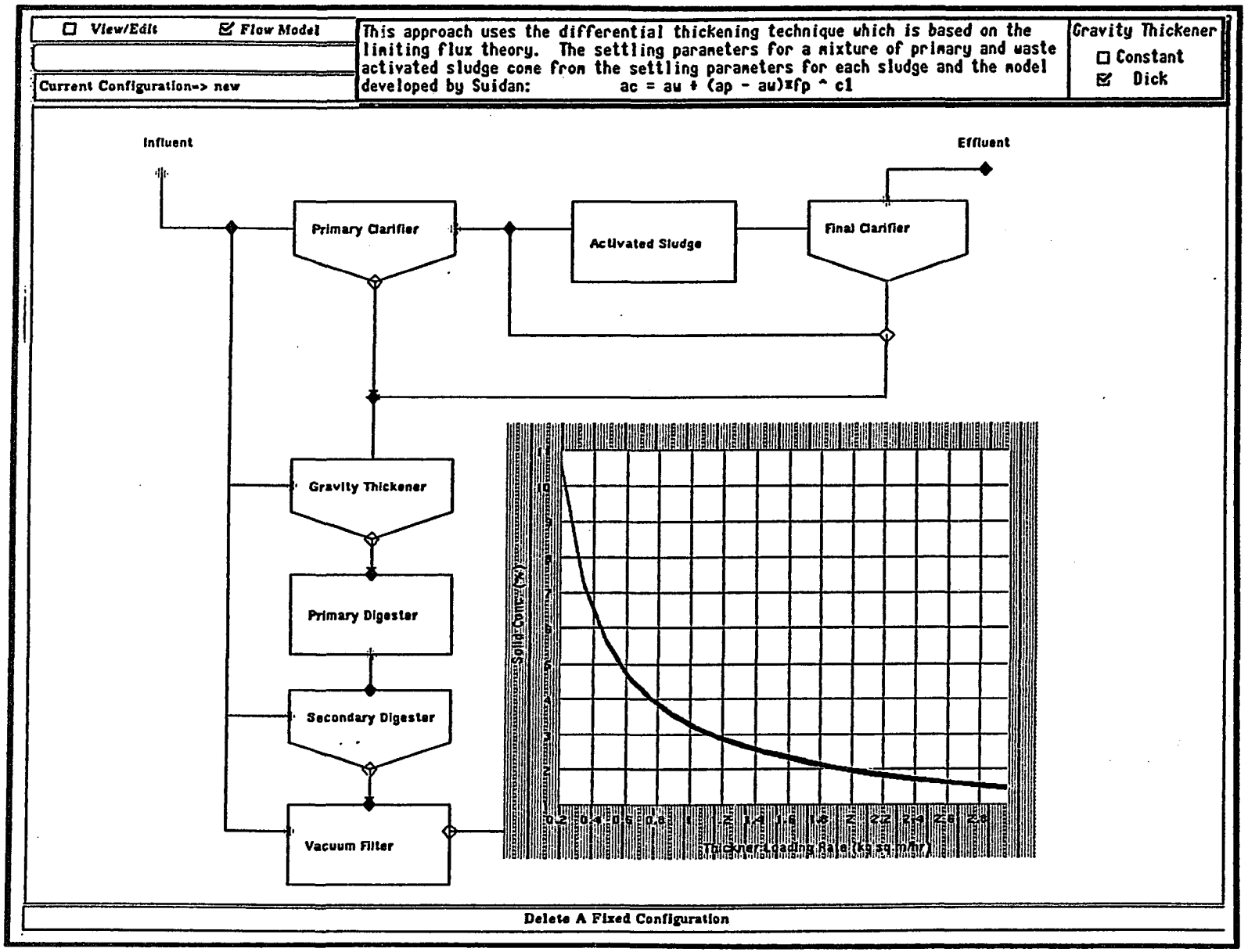


Figure 5.22

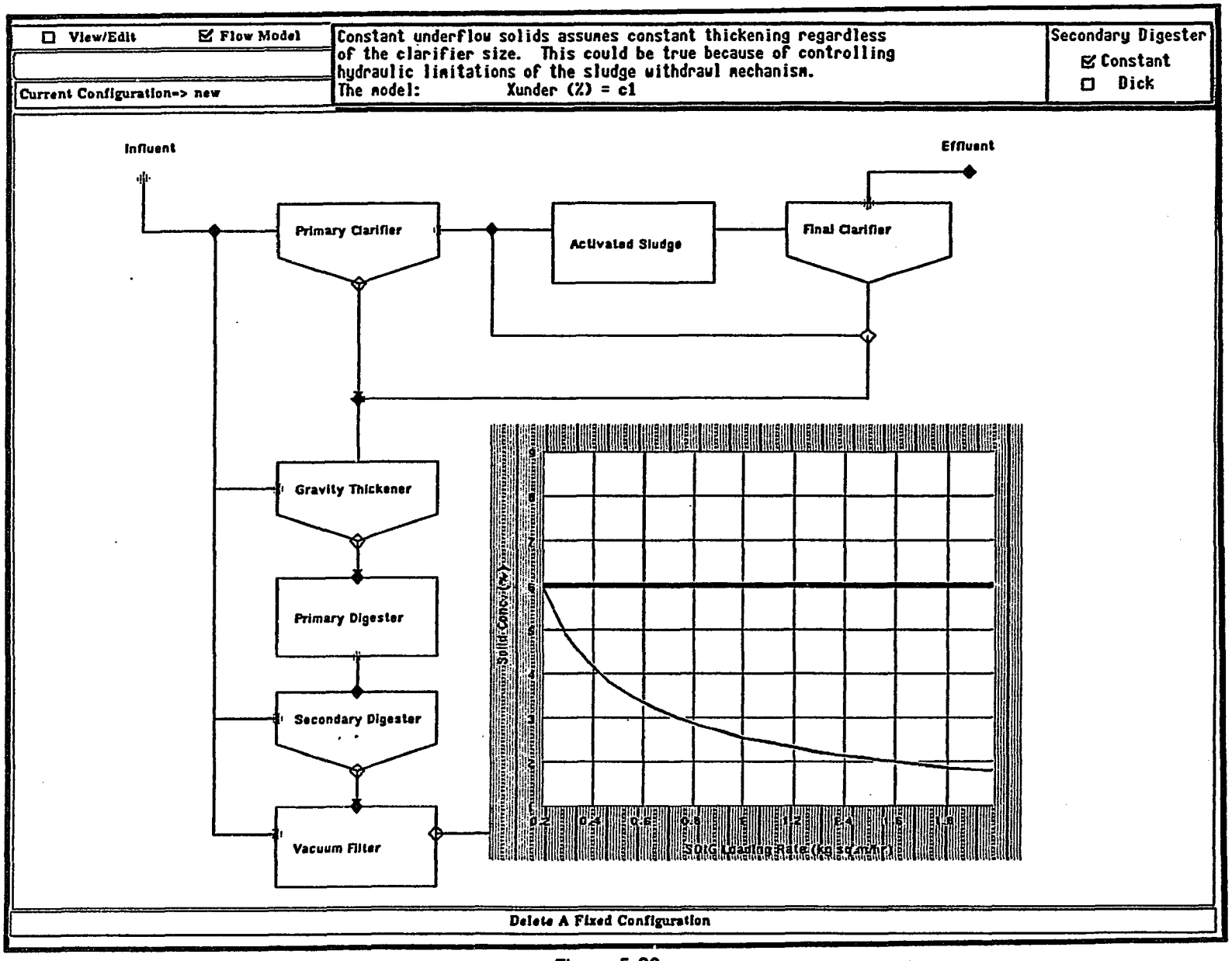


Figure 5.23

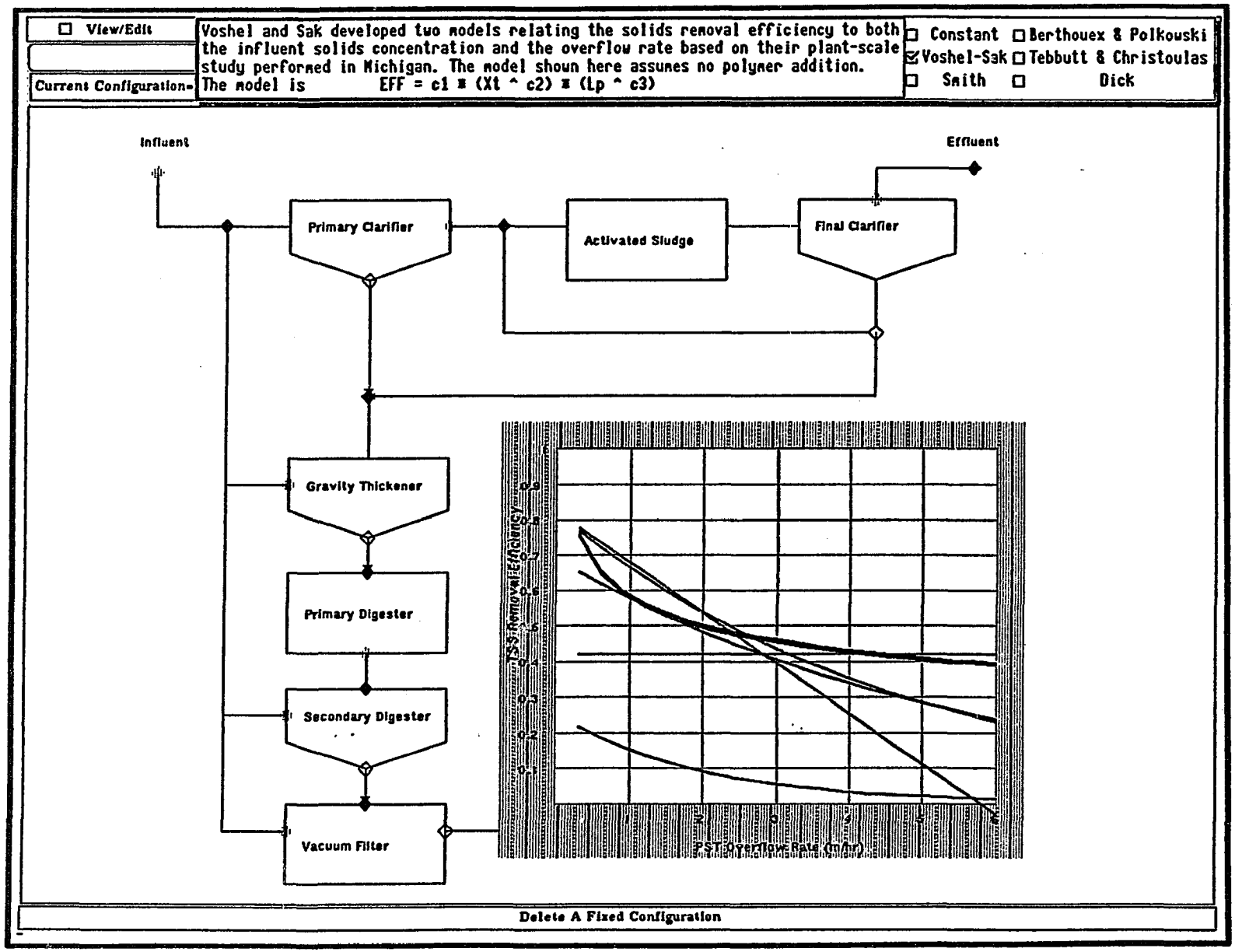


Figure 5.24

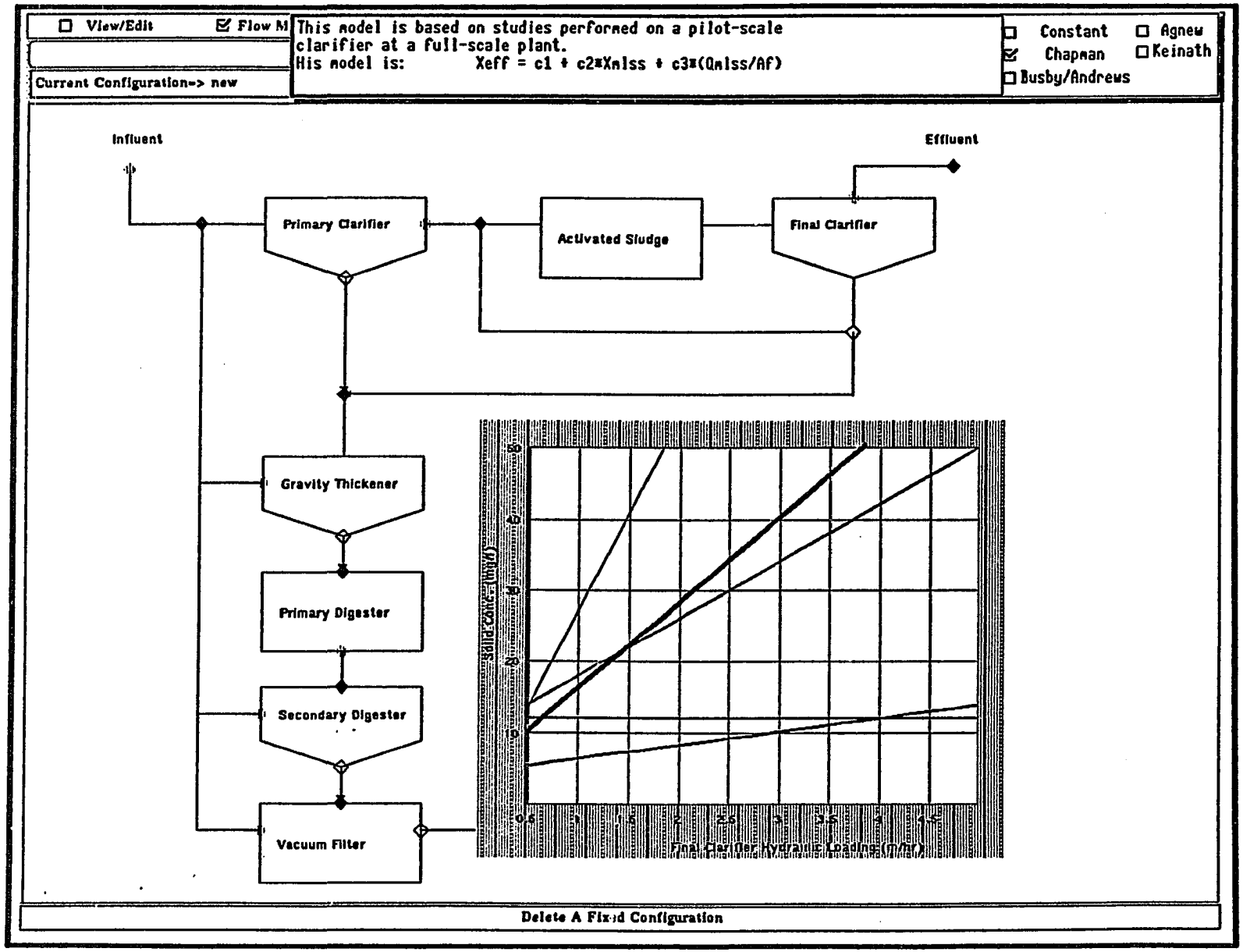


Figure 5.25

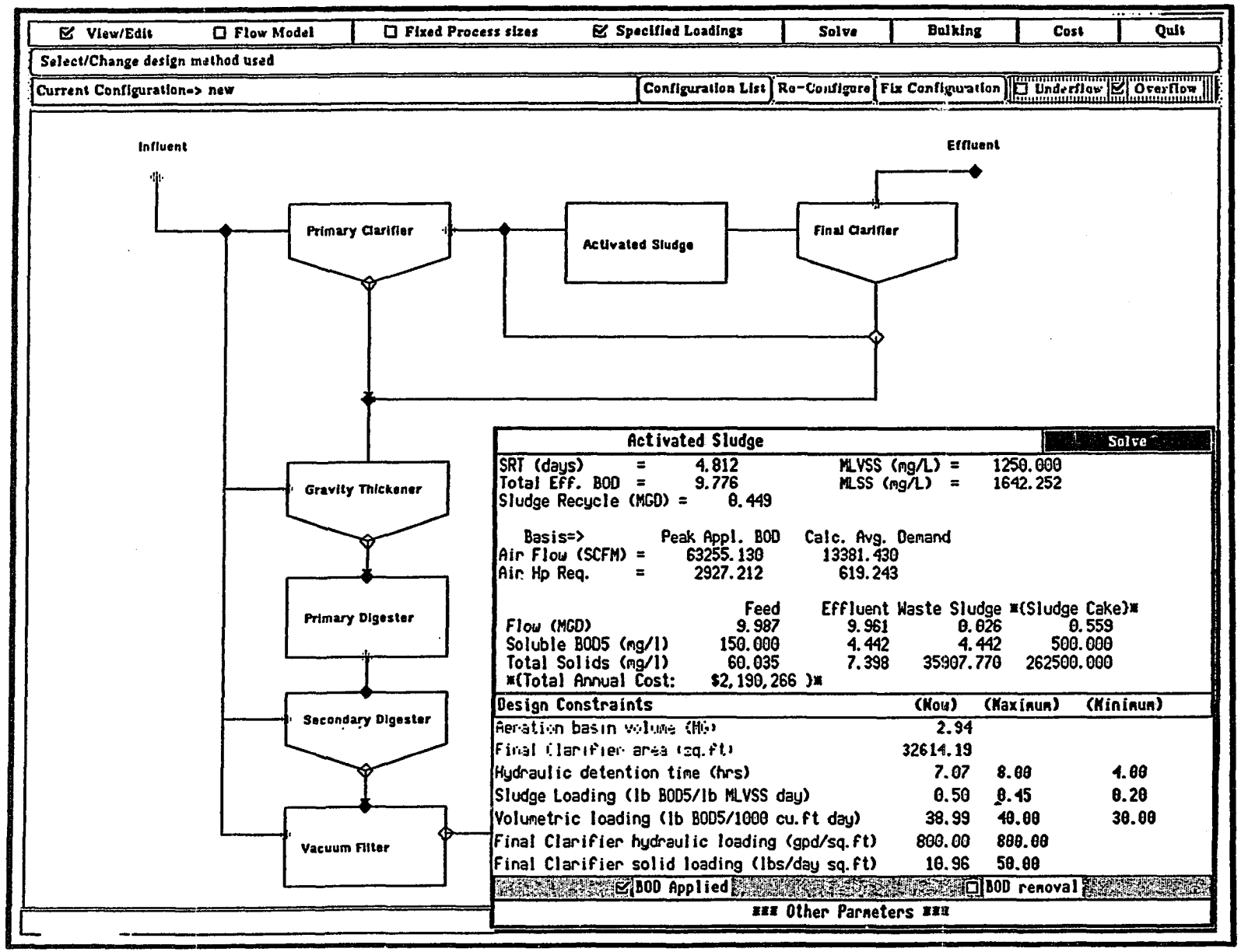


Figure 5.26

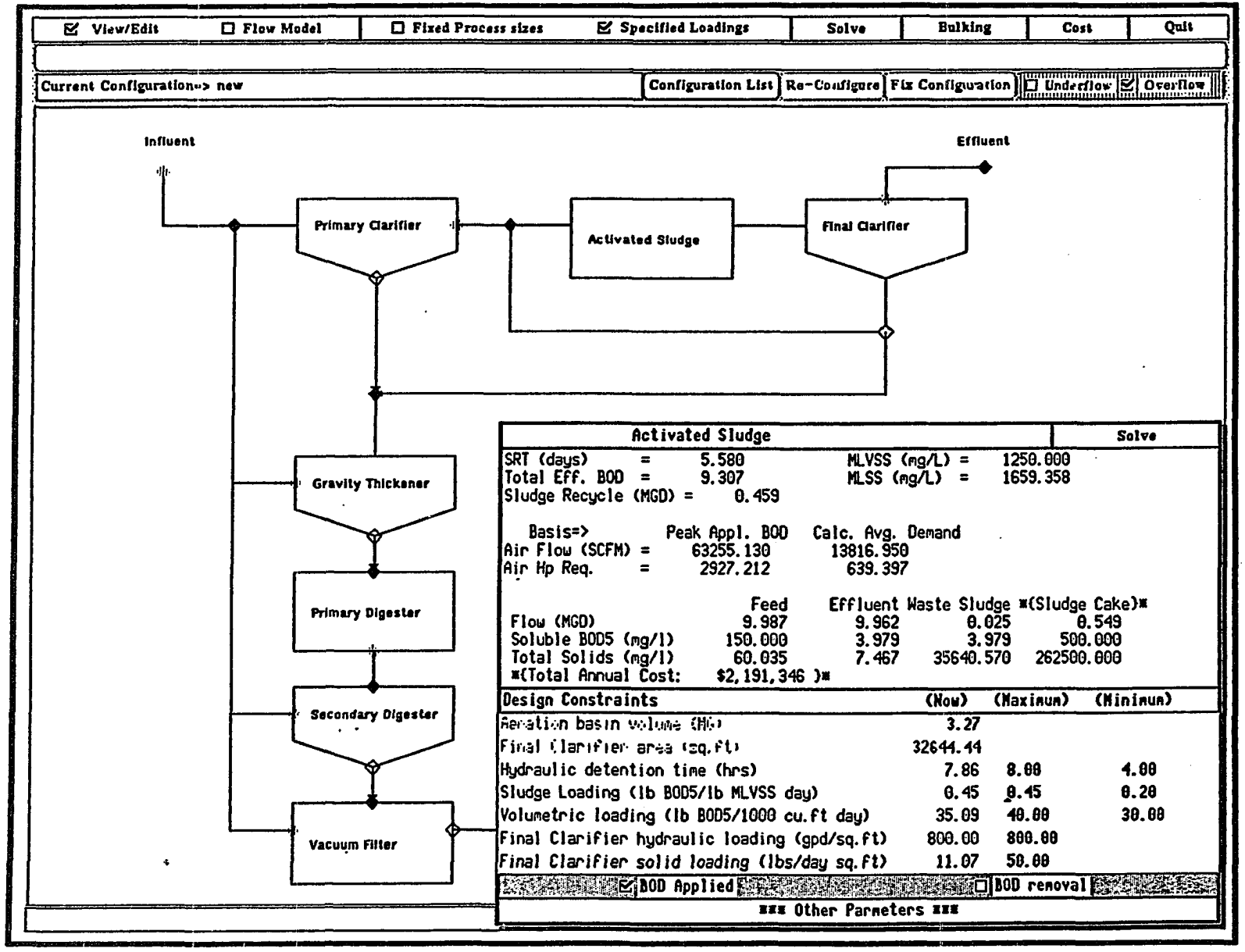


Figure 5.27

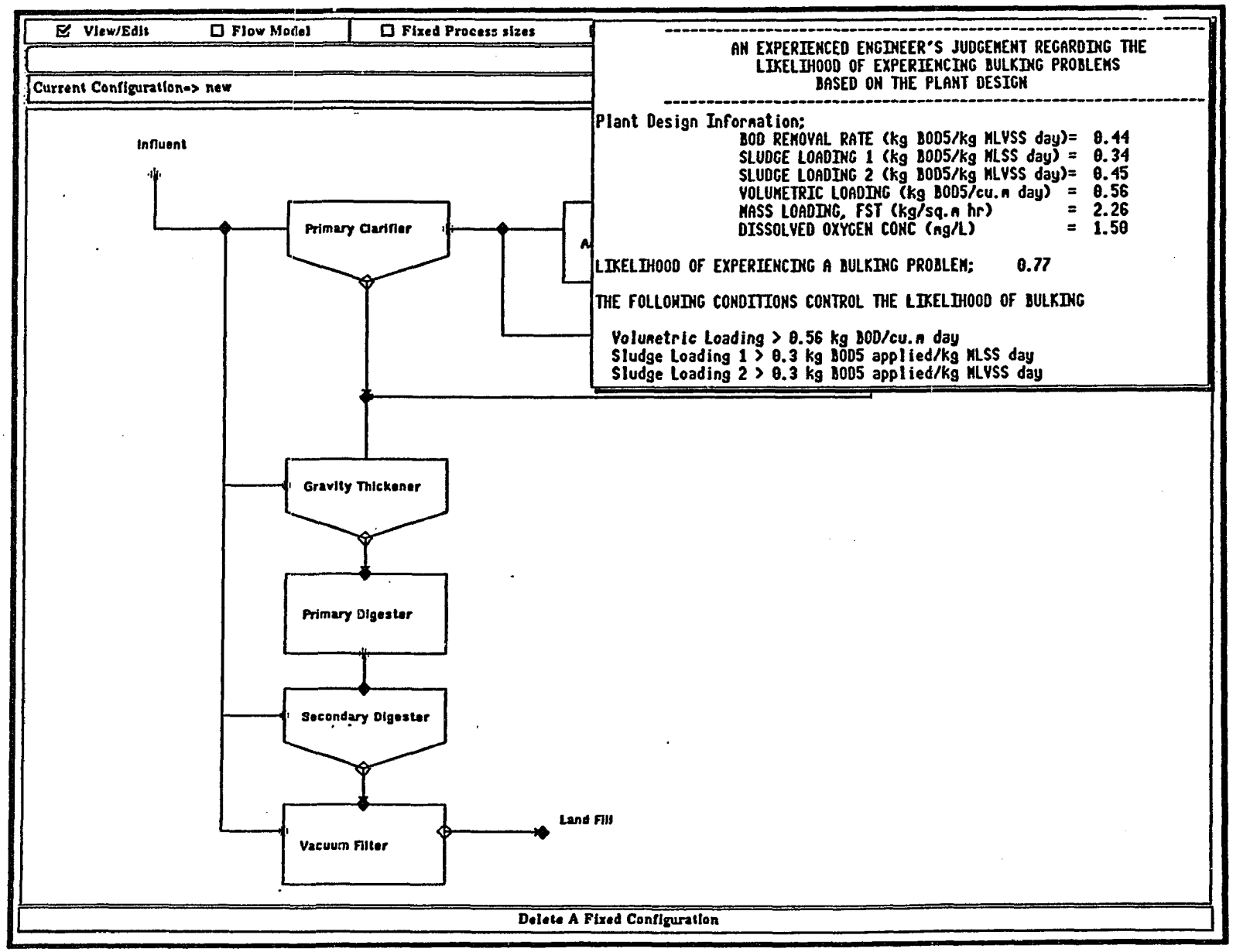


Figure 5.28

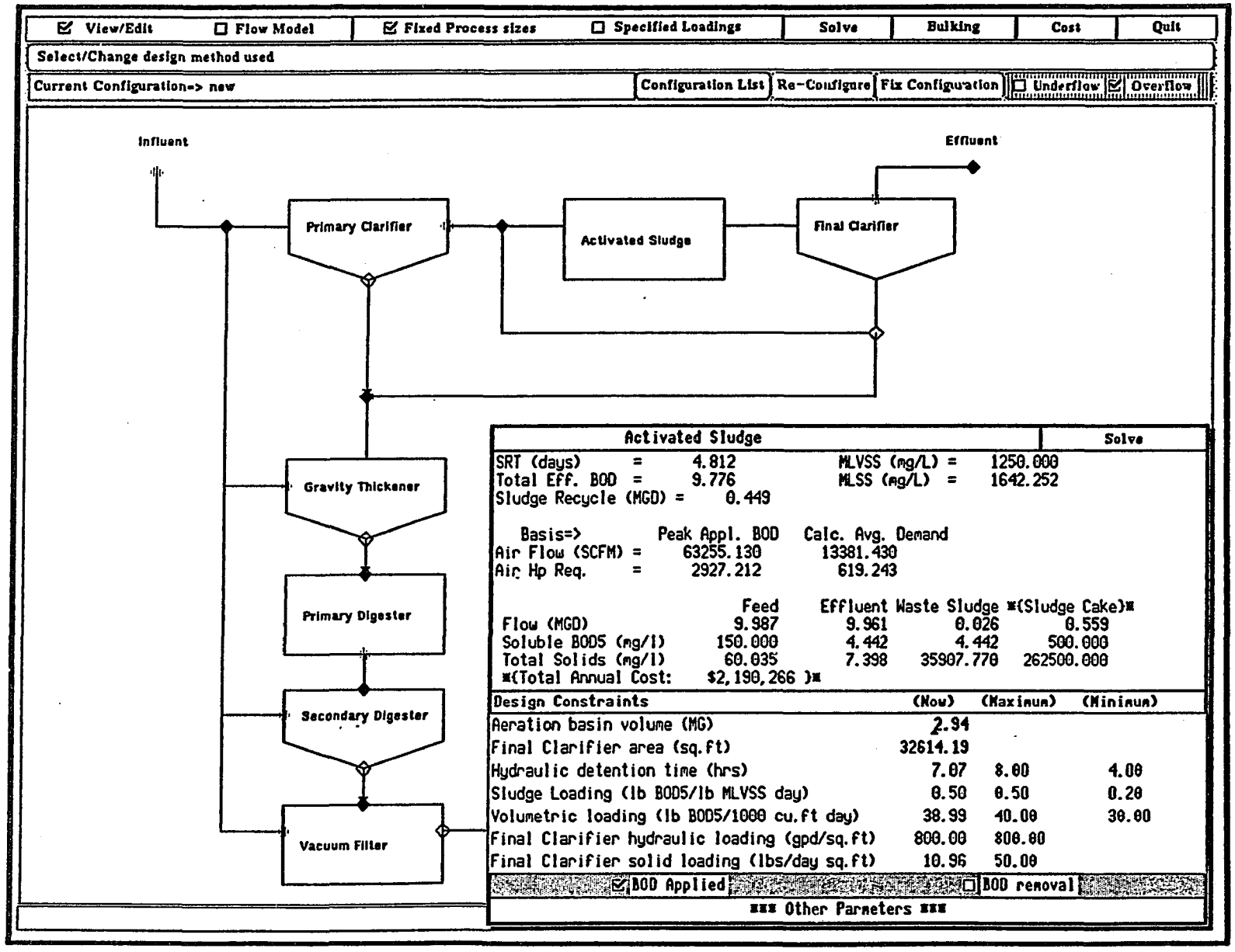


Figure 5.29

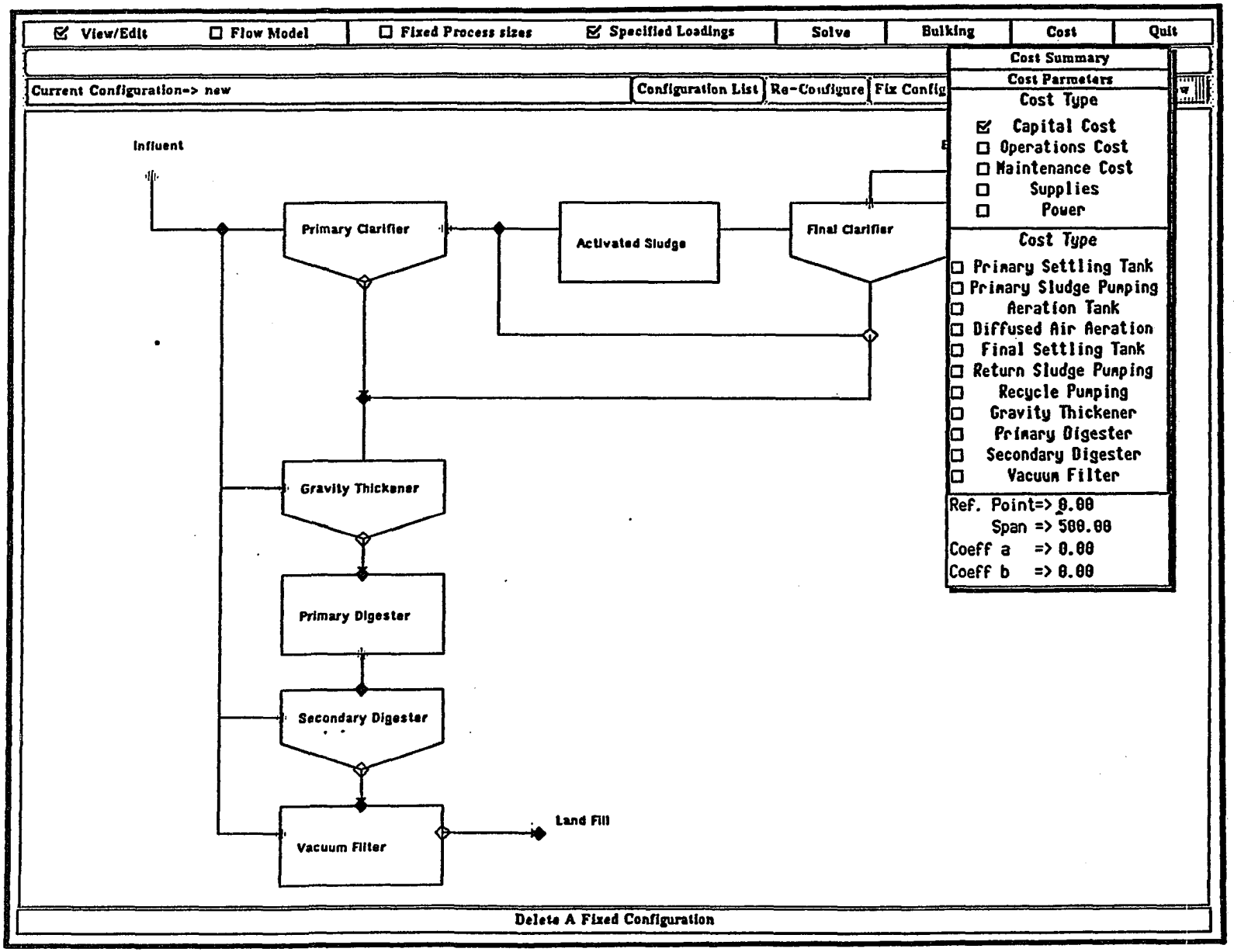


Figure 5.30

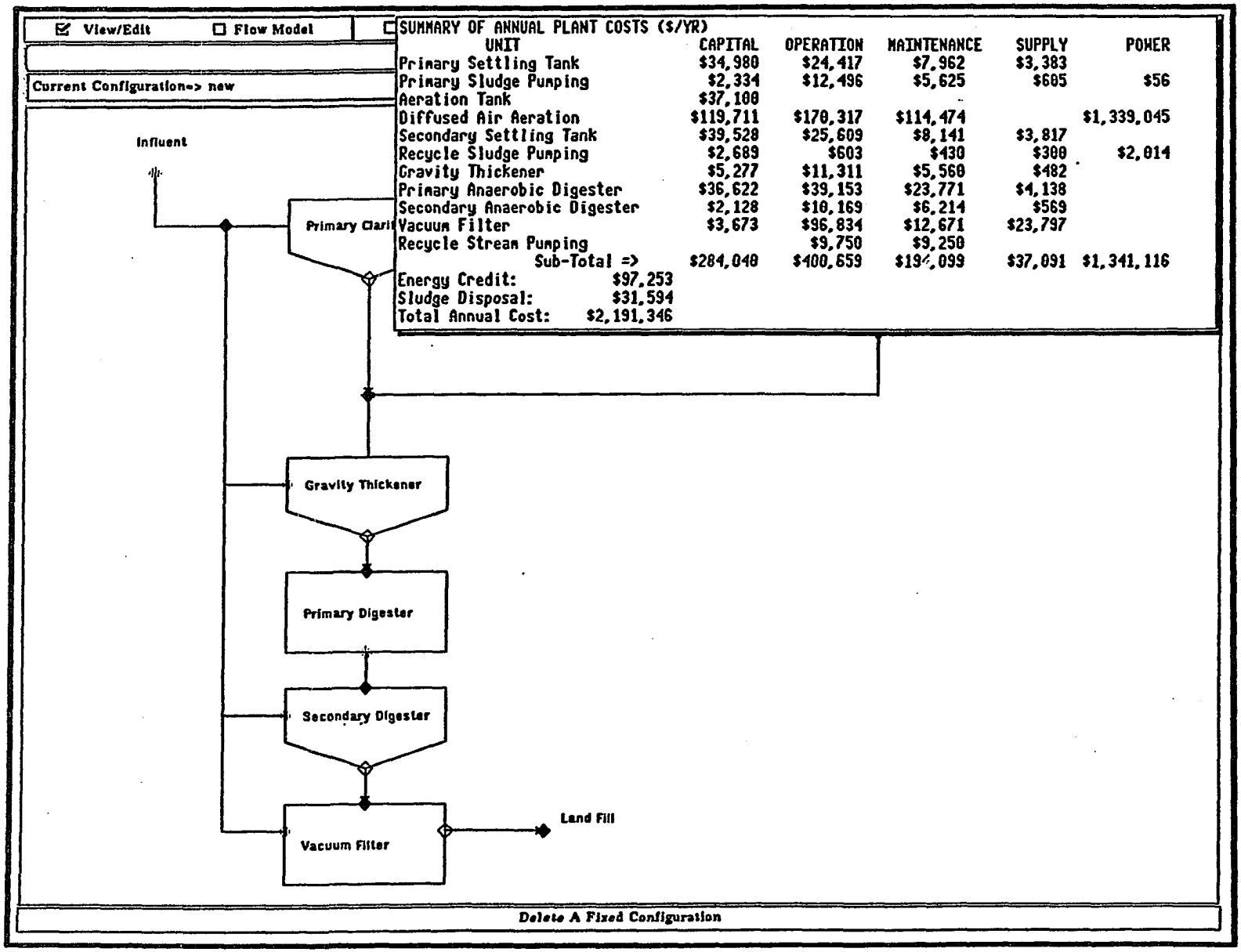


Figure 5.31

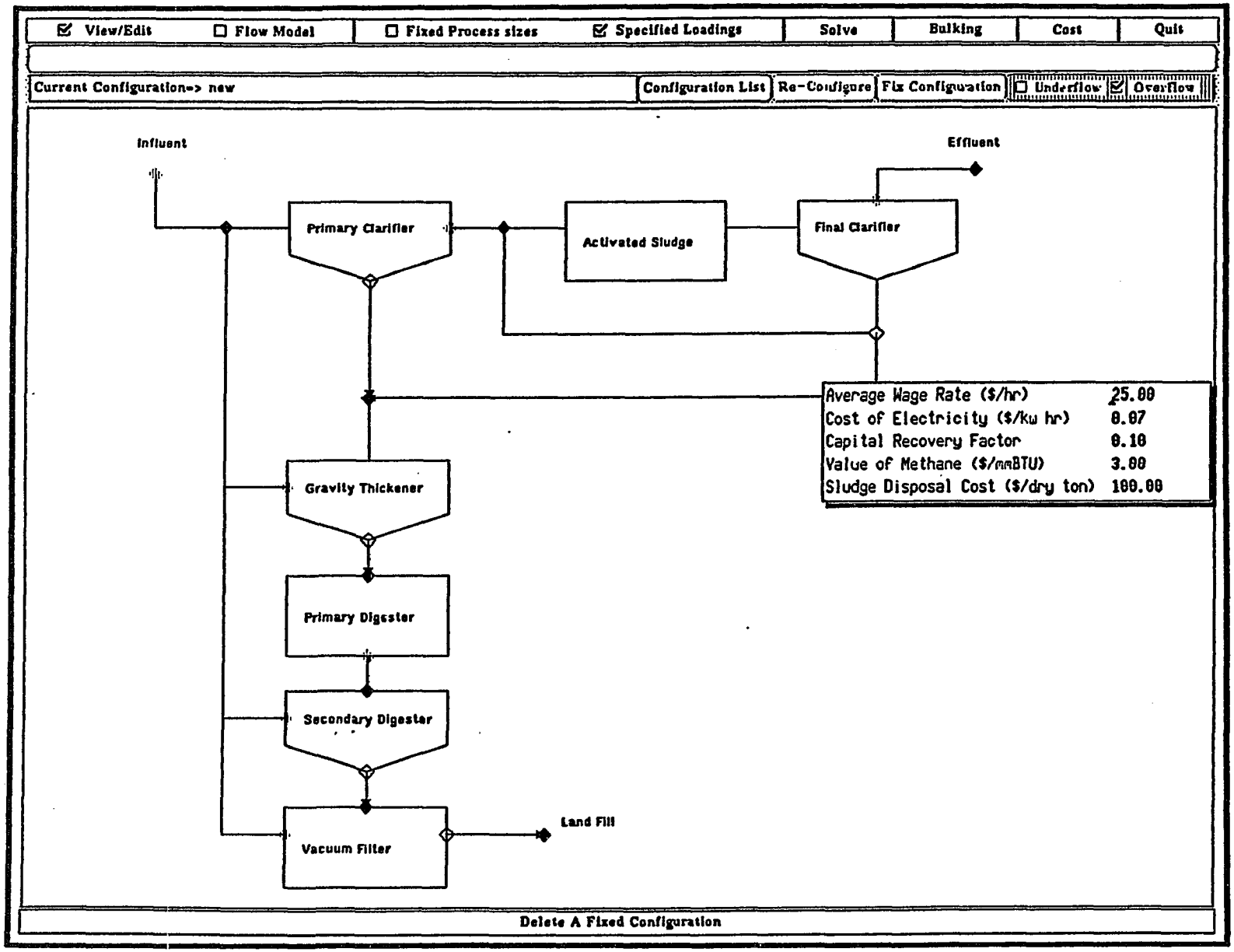


Figure 5.32

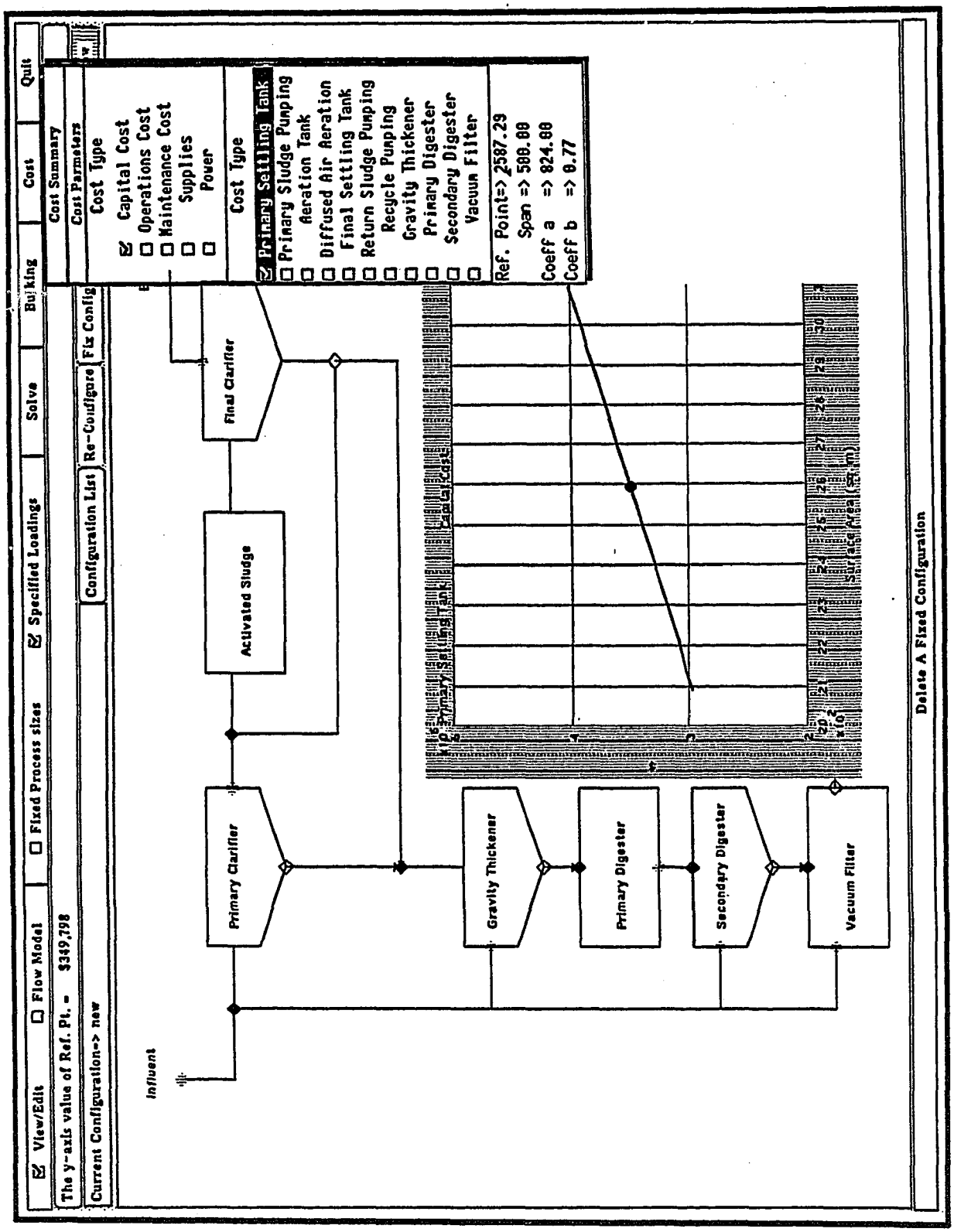


Figure 5.33

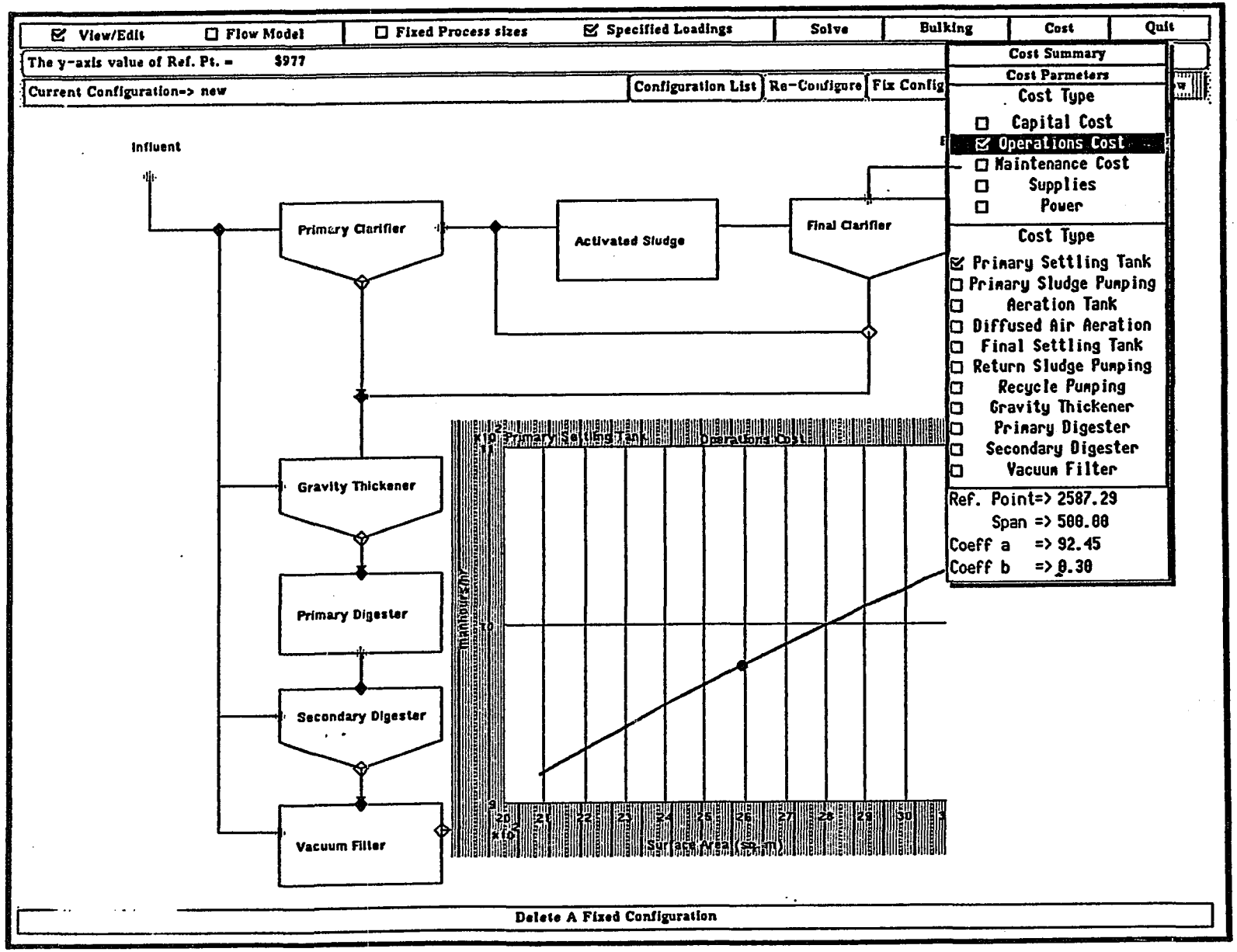


Figure 5.34

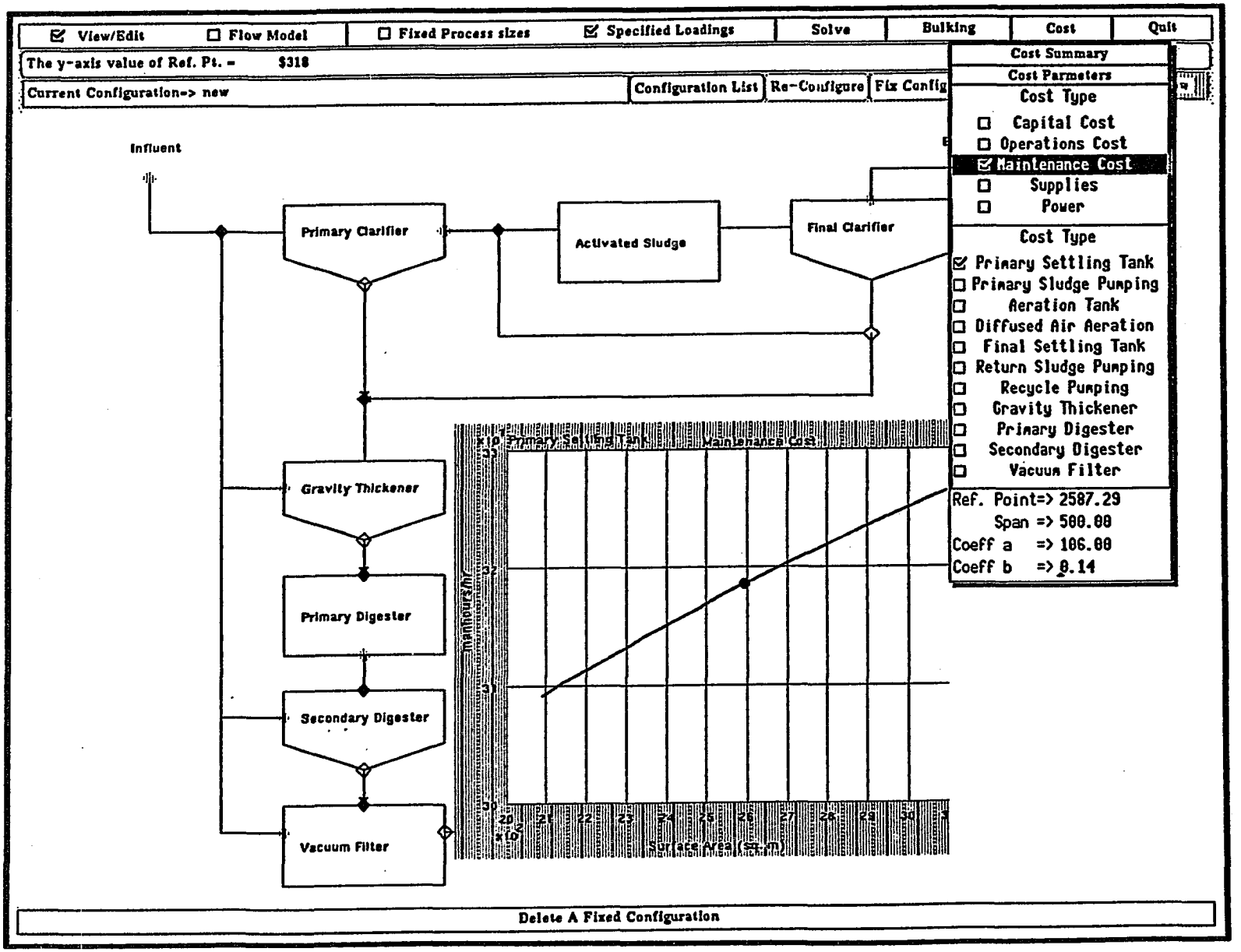


Figure 5.35

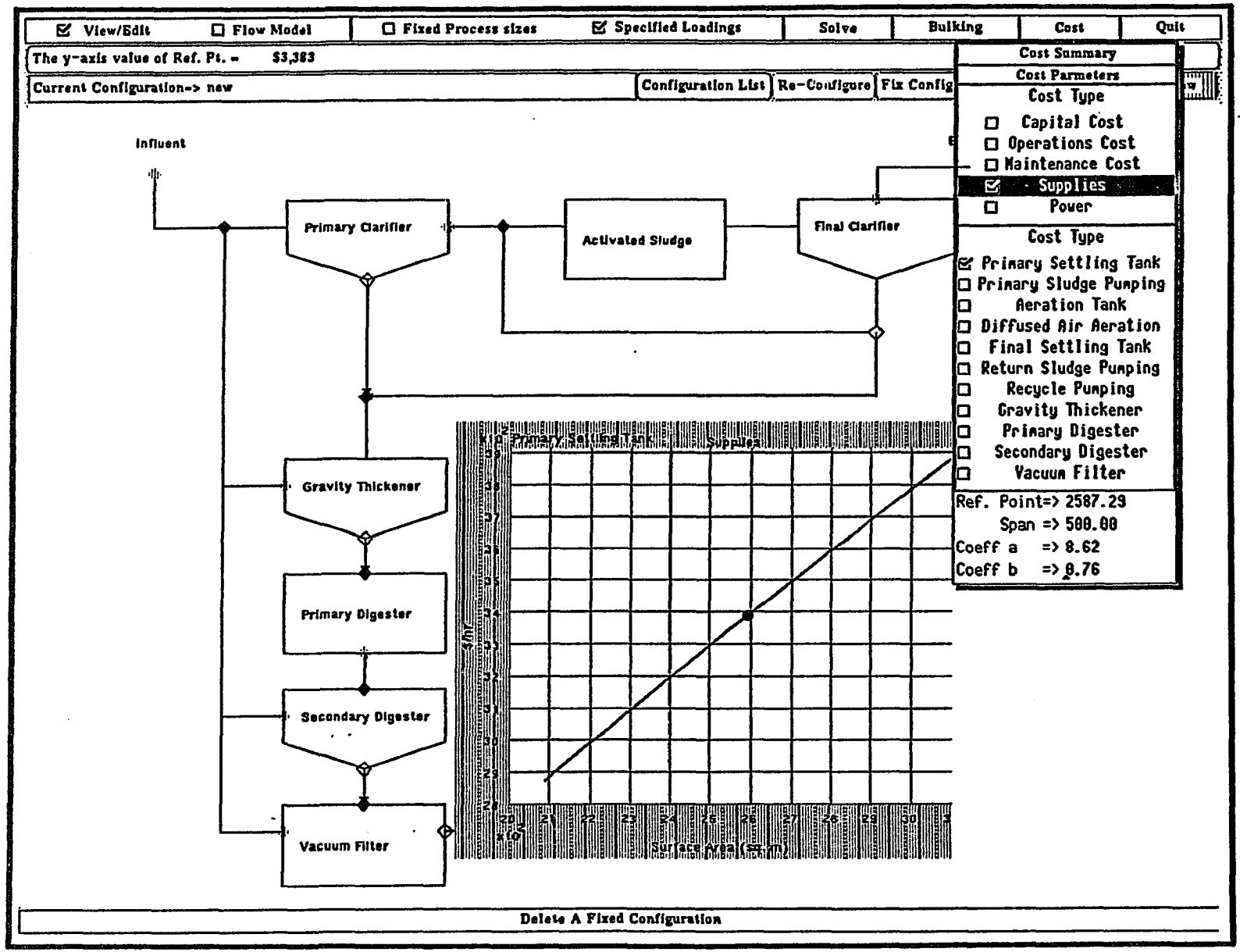


Figure 5.36

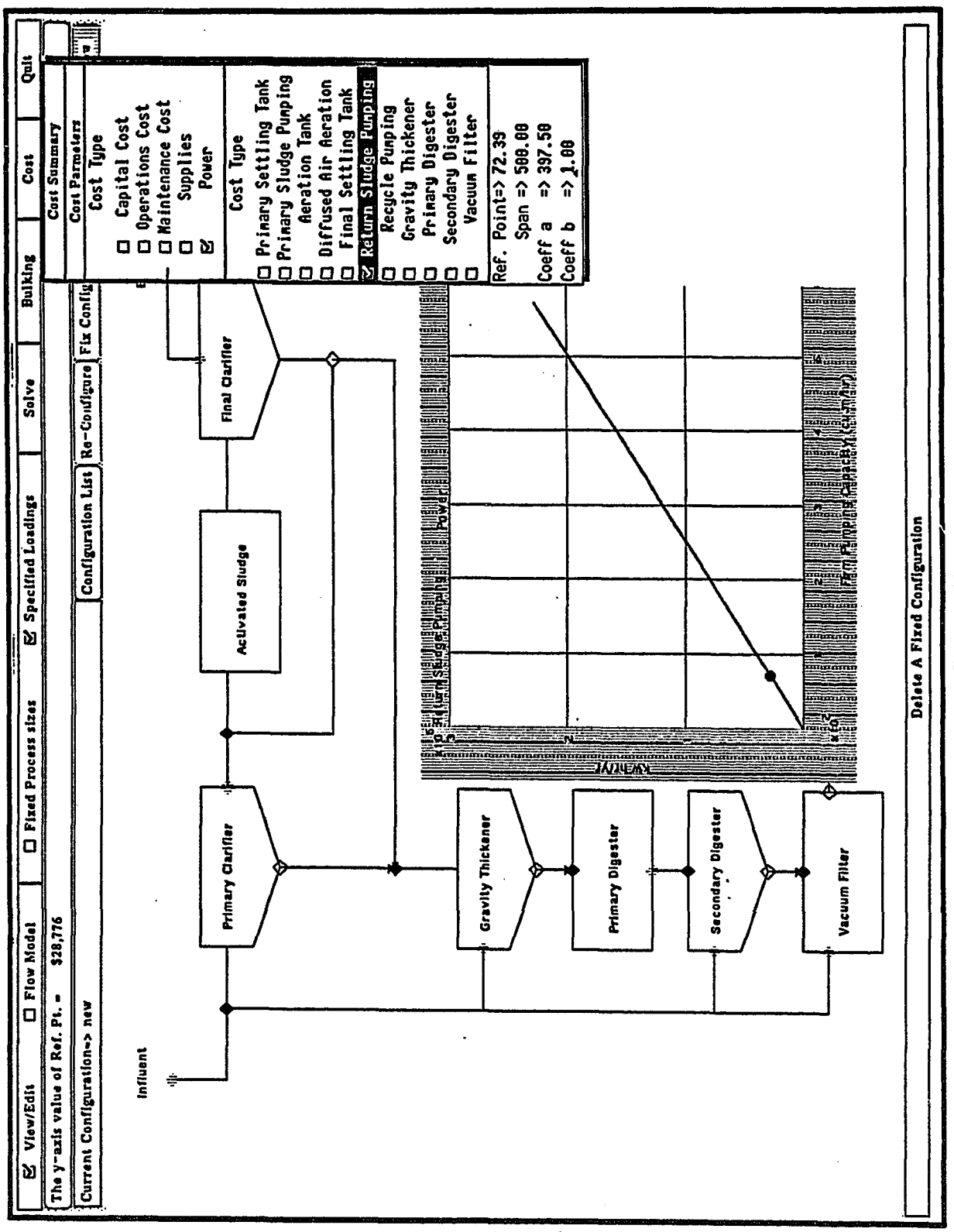


Figure 5.37

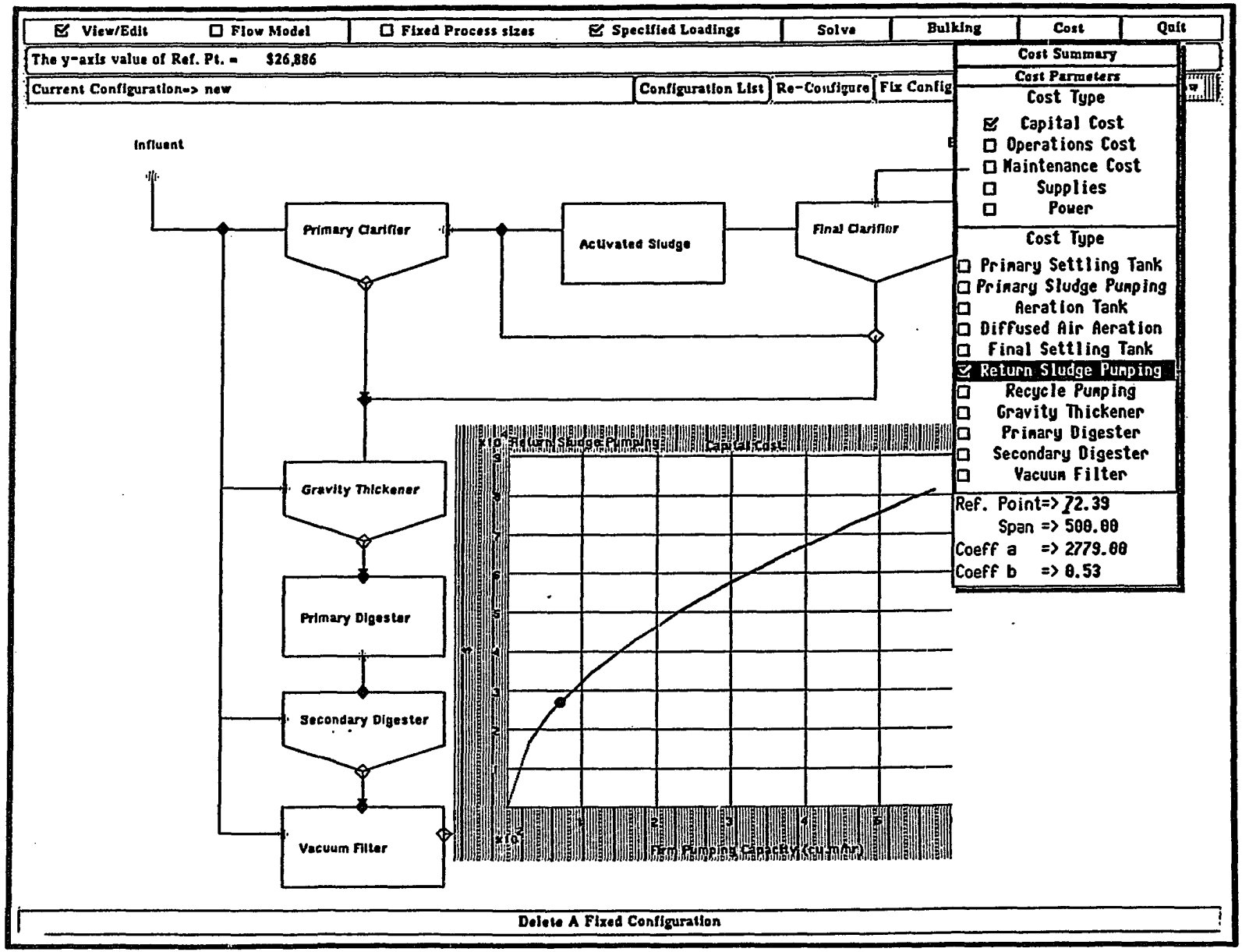


Figure 5.38

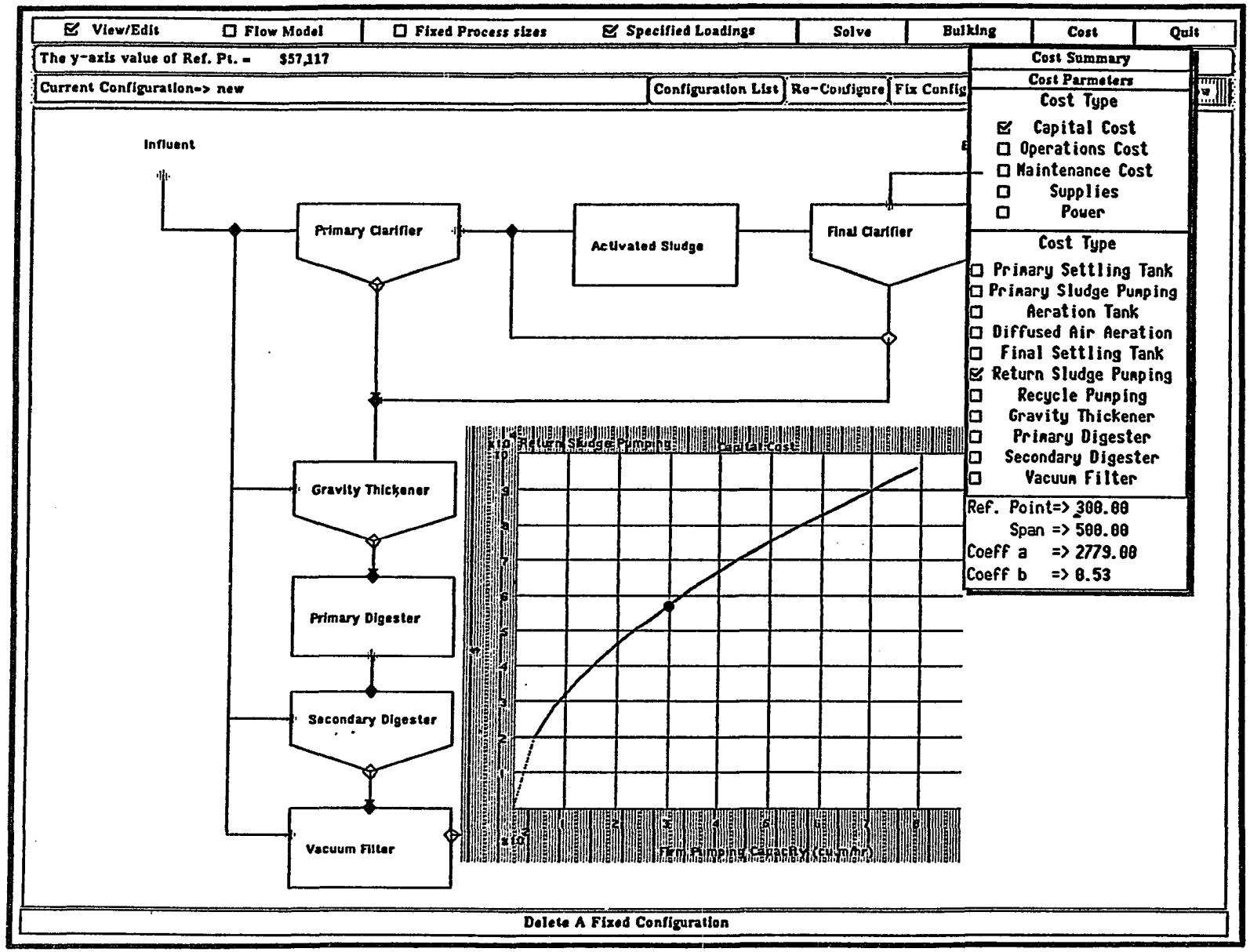


Figure 5.39

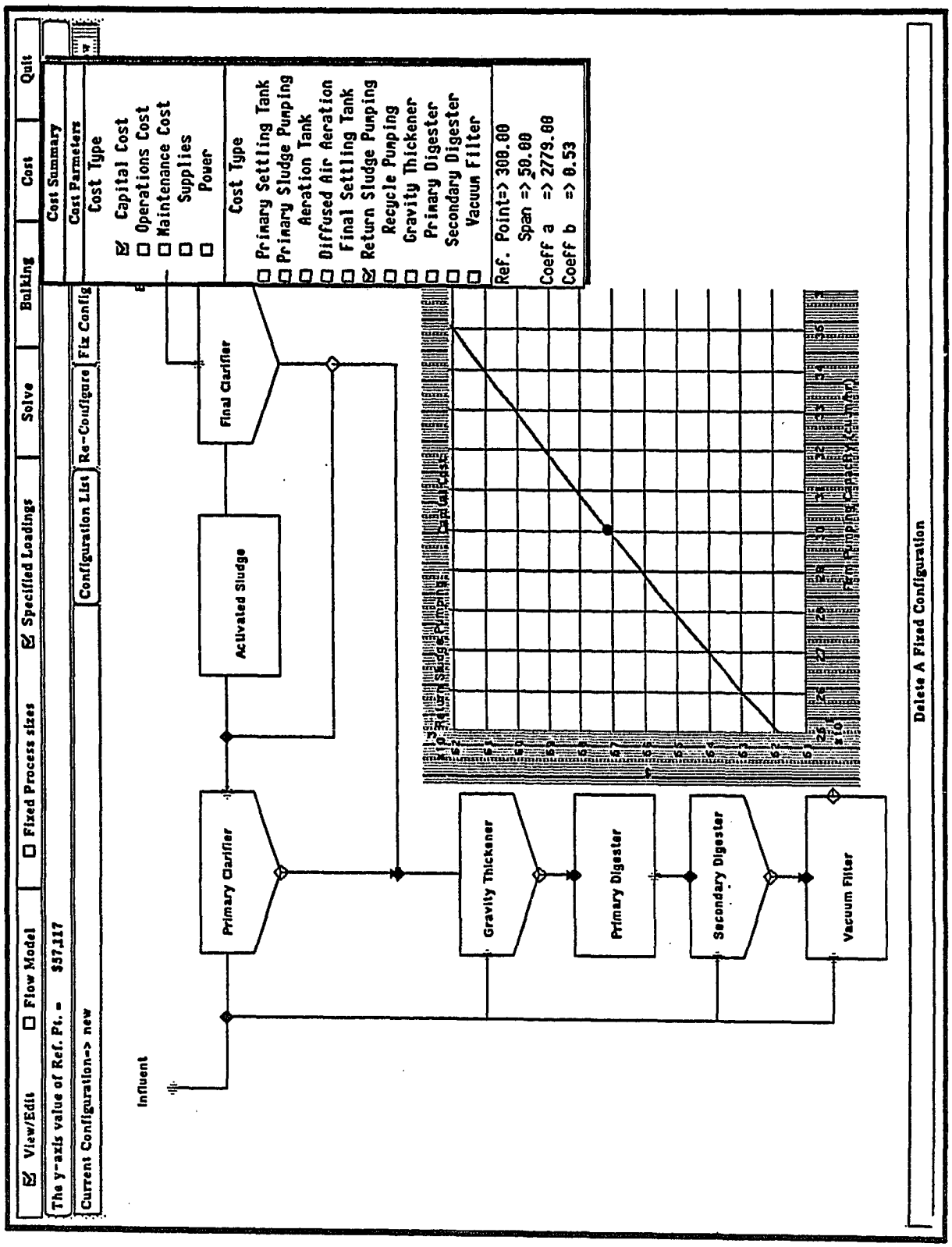


Figure 5.40

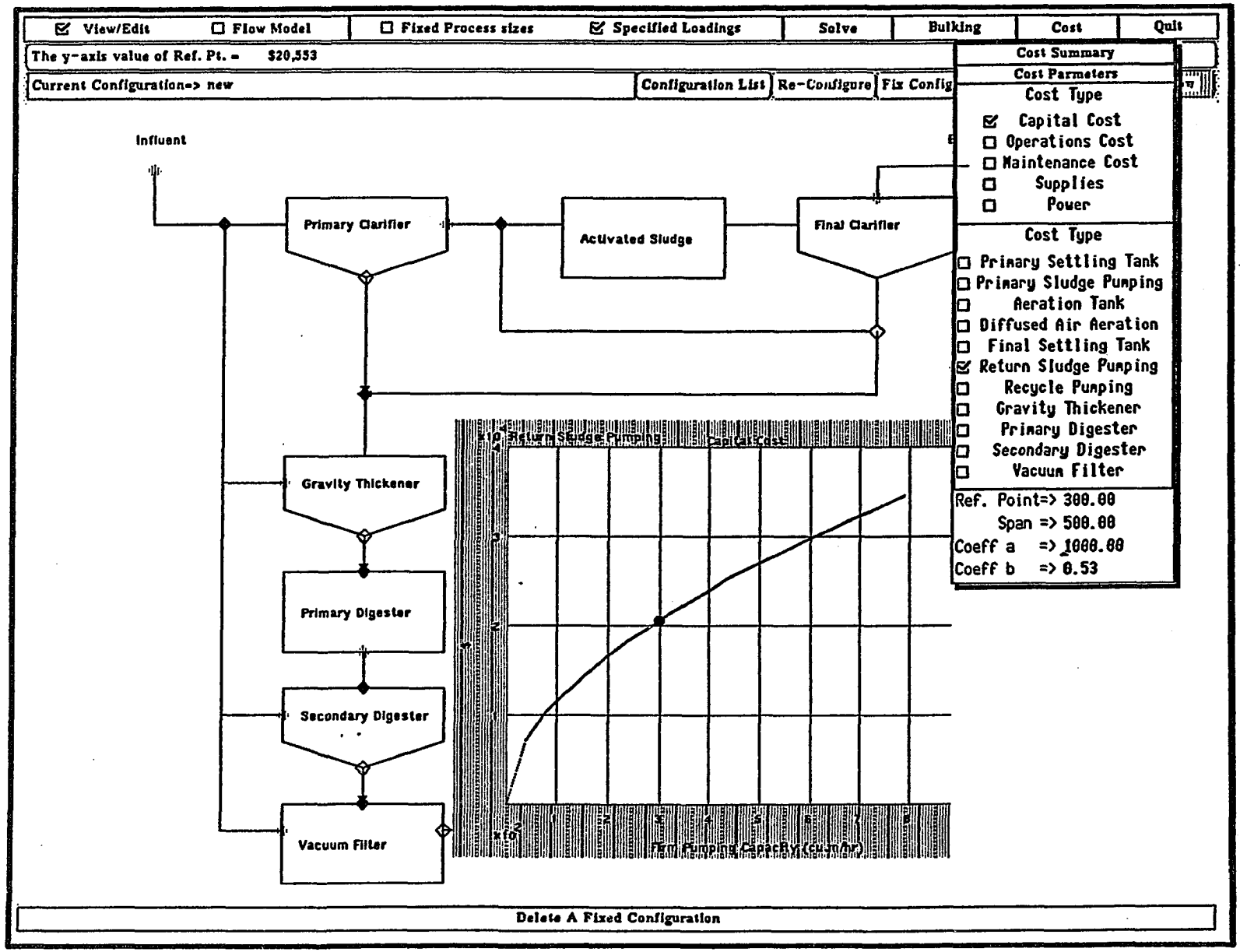


Figure 5.41

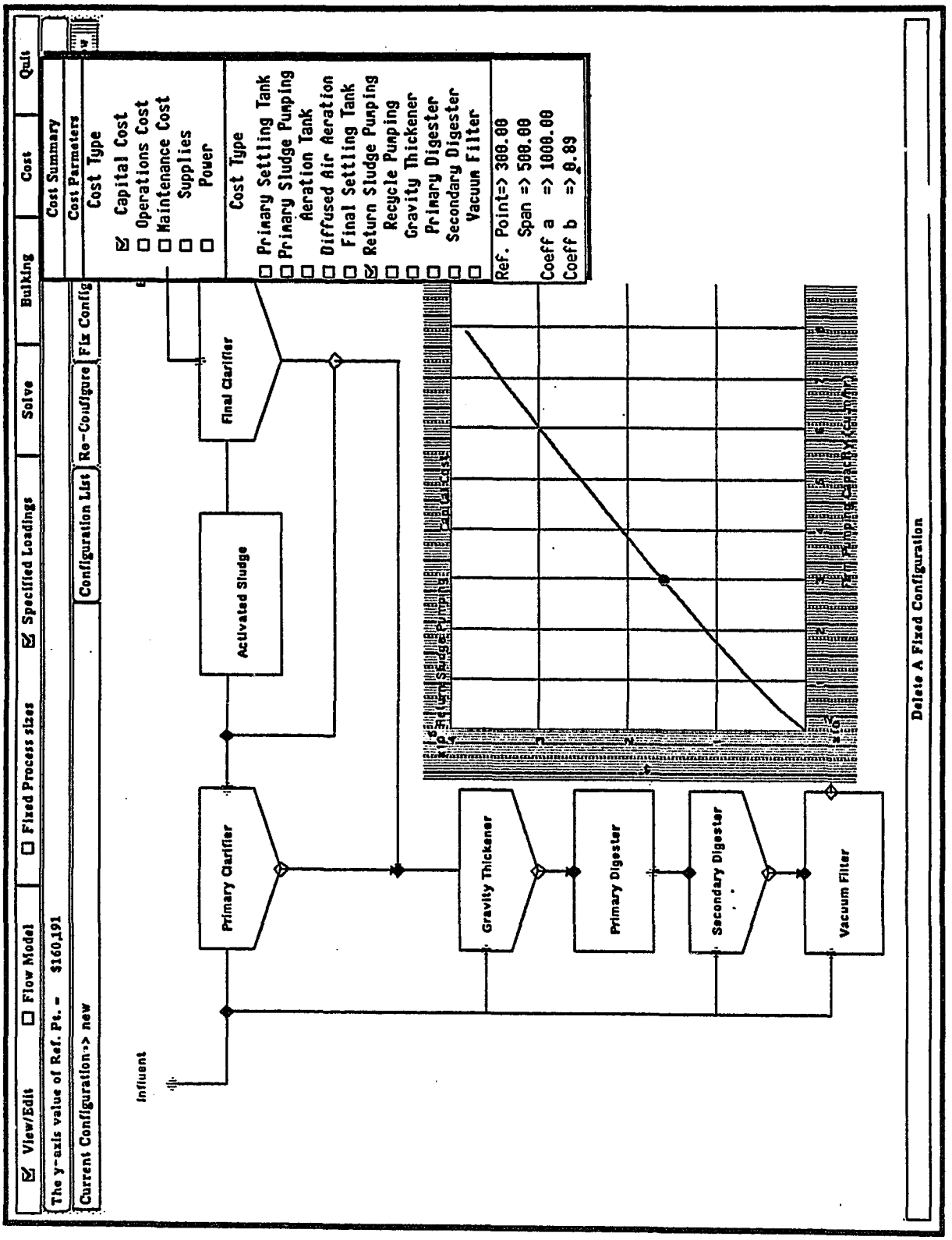


Figure 5.42

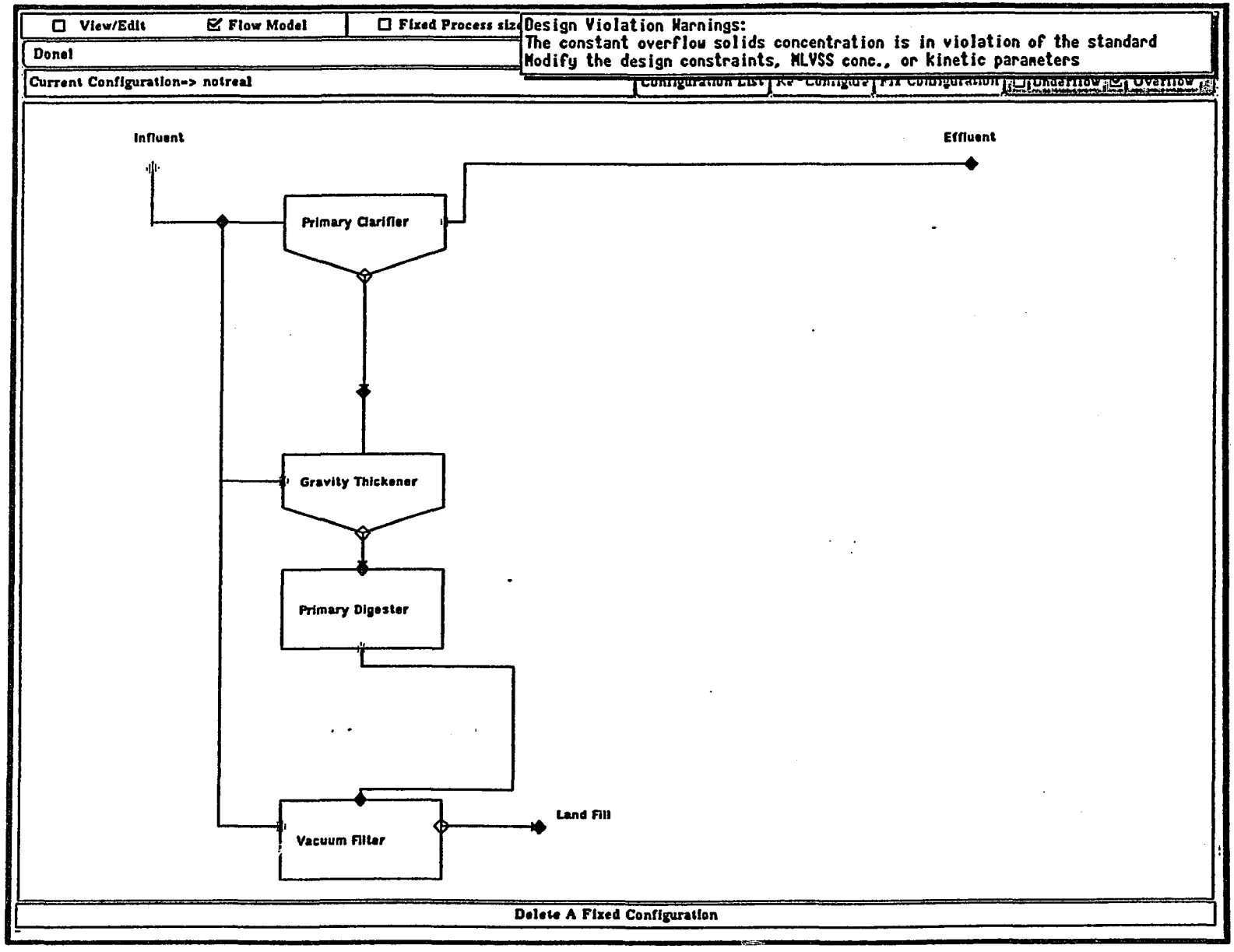
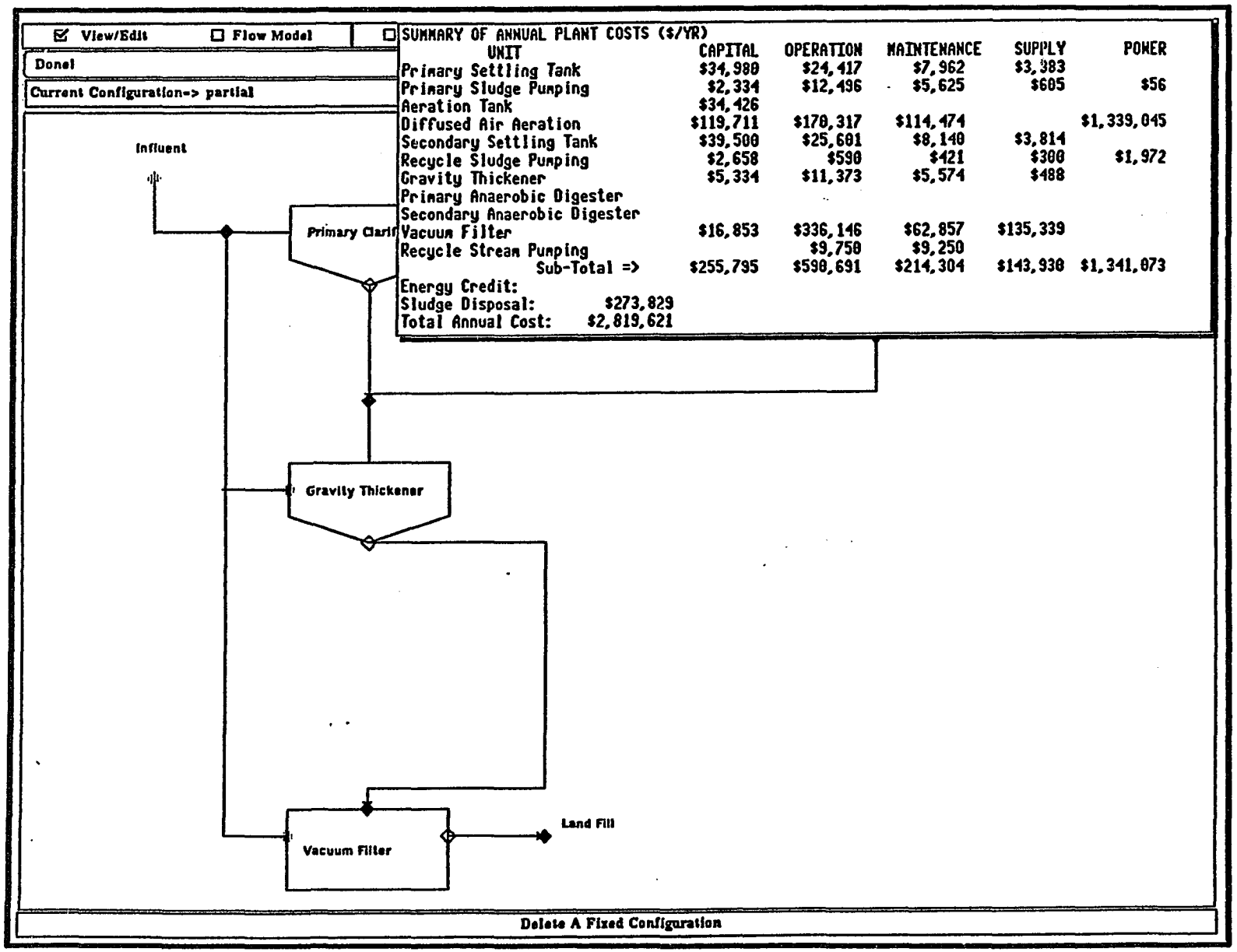


Figure 5.43



SUMMARY OF ANNUAL PLANT COSTS ($/YR)

UNIT	CAPITAL	OPERATION	MAINTENANCE	SUPPLY	POWER
Primary Settling Tank	$34,980	$24,417	$7,962	$3,383	
Primary Sludge Pumping	$2,334	$12,496	$5,625	$605	$56
Aeration Tank	$34,426				
Diffused Air Aeration	$119,711	$170,317	$114,474		$1,339,045
Secondary Settling Tank	$39,500	$25,601	$8,140	$3,814	
Recycle Sludge Pumping	$2,658	$590	$421	$300	$1,972
Gravity Thickener	$5,334	$11,373	$5,574	$488	
Primary Anaerobic Digester					
Secondary Anaerobic Digester					
Vacuum Filter	$16,853	$336,146	$62,857	$135,339	
Recycle Stream Pumping		$9,750	$9,250		
Sub-Total =>	$255,795	$590,691	$214,304	$143,930	$1,341,073

Energy Credit:
Sludge Disposal: $273,829
Total Annual Cost: $2,819,621

Figure 5.44

CHAPTER 6

COMPUTER AIDED SYSTEM FOR GROUNDWATER RESOURCES MANAGEMENT

6.1. Groundwater Resources Management (GRM)

To analyze a GRM model, a single objective approach is generally not sufficient to describe the interrelationships among several important issues such as yield, demand, cost, and equity. A multicriterion technique is thus generally used, e.g. Willis, et al. [1984] and Louie, et al. [1984]. Although a multicriterion model can provide insights for a problem, the model may not be complete if there exist several uncertainties or unquantifiable issues which are unmodeled (e.g. transmissivity and social values, respectively). To attempt to analyze a noninferior set obtained from such an incomplete model may not be very useful if unmodeled issues are significant. To compare a variety of alternatives which may not be noninferior but that have significant differences among them is one approach to dealing with an incomplete model.

In this research, the Vector Method is used for solving a multicriterion model, and the MGA approach is used for generating alternatives that are good when compared with a noninferior solution but that are significantly different in decision space.

The general formulation of a multicriterion GRM model is expressed in Table 6.1. Five typical objectives, total cost, water deficit, total hydraulic head, net groundwater extraction, and equity, are listed. Although more than three objectives can be considered for a GRM model, the presentation of noninferior sets is difficult. To analyze a model with more than three objectives, it is possible to use the software developed for implementing the Vector Method. The prototype developed, however, is so far limited for a model with at most three objectives. Thus, a model with more than three objectives can not be solved by the prototype, although it is possible to create the optimization model in the proposed modeling language.

A simulation model using the finite element method was used in this research to evaluate steady-state two-dimensional groundwater flow. This model is described in Appendix A.

$$Min \sum_{i=1}^{s} F_i\ (Q_i) \qquad \text{(total cost)}$$

$$Min \sum_{t=1}^{z} \sum_{i \in R_t} WD_i \qquad \text{(total water deficit)}$$

$$Max \sum_{i=1}^{m} \sum_{j=1}^{n} h_{i,j} \qquad \text{(total hydraulic head)}$$

$$Min \sum_{i=1}^{s} Q_i - \sum_{j=1}^{r} Q_j \qquad \text{(net groundwater extraction)}$$

$$Min \sum_{t=1}^{z} (u_t + v_t) \qquad \text{(equity for water deficit)}$$

S.T.

$$\mathbf{A\,h = B} \qquad \text{(see the simulation model in Appendix A)}$$

$$\sum_{i \in R_t} (Q_i + Sw_i + WD_i) = D_t - P_t \qquad \forall t$$

$$\frac{\left(\sum_{t=1}^{z} \sum_{i \in R_t} WD_i \right)}{\sum_{t=1}^{z} PA_t} = U_{d,ave}$$

$$u_t - v_t + \frac{\sum_{i \in R_t} WD_i}{PA_t} = U_{d,ave} \qquad \forall t$$

Table 6.1 General Formulation Of A GRM Model

where

s:	the number of groundwater resources, e.g. pumping well;
Qi:	the amount or discharge or recharge of groundwater at location i;
Fi(Qi):	cost function;
z:	the number of sub-basin regions;
Rt:	sub-basin region t;
WDi:	water deficit at region i;
m,n:	the number of subdivision elements in each principle axis of the two dimensional domain;
hi,j:	hydraulic head at location (i,j);
r:	the number of groundwater recharge sources;
A:	the coefficient matrix, a function of storage coefficients and transmissivities plus basis functions of the problem (finite element method is used here);
h:	a vector of the hydraulic heads;
B:	boundary conditions (Dirichlet and Neumann) and groundwater extraction or recharge.
Swi:	available surface water at location i;
Dt:	demand at region t;
Pt:	effective precipitation in region t;
PAt:	population or area of region t;
Ud,ave:	unit deficit per person or unit area;

Table 6.1 General Formulation Of A GRM Model (Continued)

6.2. Computer Aided System

6.2.1. General

In terms of decision making support, the prototype developed for a GRM model is more complete in some ways than the one developed for the WTPD problem. Current limitations of the WTPD prototype are the lack of a good comparison function and lack of optimization techniques. For the GRM prototype, since the computation time required for mathematical optimization is short, optimization can be incorporated in the interactive interface.

The groundwater simulation model used is a two dimensional flow model. A transport model for contaminants has not yet been incorporated into the prototype. The flow model can be used to analyze the impact of pumping and injection wells on hydraulic heads in an aquifer(s). The general purpose of wells may be to satisfy water demand, to control subsurface flow, or to monitor the contaminant plumes. The prototype is not restricted to any specific purpose. An analyst has flexibility to construct any flow model for any purpose(s). The prototype provides user friendly interfaces and reliable techniques to implement effectively each stage of a decision making process (see Chapter 3). It can be used to build and modify a model, solve the model, do simulation, determine impact, optimize the model, generate alternatives, and provide comparisons for alternatives.

In Section 6.2.2 the GRM prototype is described, and graphical presentations for improving the effectiveness of major analysis tasks are provided. Several general features adopted from the WTPD prototype were not developed again (e.g. cost information, selection of models or solution techniques, and tabular output of solutions or alternatives). However, these features could be added to the GRM prototype to make it more complete.

Software was developed for implementing some of techniques described in Chapter 3: the modeling language, finite element analysis, the Vector Method, and an MGA method. The GRM prototype contains many useful tools to aid decision making. In the next section, key figures are shown to demonstrate how the tools of the GRM prototype could be used to improve the effectiveness of major decision making analysis tasks. An entire working session is not simulated using figures as shown for the WTPD prototype. Again, the best way to understand the entire prototype is to view a videotape or a live demonstration.

6.2.2.Demonstration

The GRM prototype can be used to relieve an analyst(s) or decision maker(s) from tedious tasks involved with: the setup and modification of a simulation model(s), simulation analysis and optimization,

the generation of alternatives, presentations, and comparisons. The capabilities of the GRM prototype for these tasks are discussed with graphical presentations as follows. Menu items are explained with example(s) for tasks in the order of the decision making process described in Chapter 3. Although an overview of the GRM prototype is presented, there are numerous unexplained details, especially for the animated and interactive displays, which can be better seen from a live demonstration.

Setup and Modification of a Simulation Model(s)

To build a groundwater model, a figure(s) for the problem domain is often used. A program (MAKER) was developed to simplify the drawing tasks. In Figure 6.1, the user interface of MAKER is shown. By using a pointing device (mouse) the analyst can click the mouse button and move the mouse cursor to construct a groundwater domain easily. The graphical objects created by MAKER are intended to provide better comprehension of the model; they are not part of the mathematical models described in the following discussions. MAKER provides most basic drawing functions (line, box, circle, text, and free draw), options (fill, outline, arrow, duplication, grids, fill patterns, drawing patterns, and fonts), figures management (open, copy, name list, delete, and print), and actions (mouse clicking and moving, undo, erase, clear, reverse, pick, refresh, and exit). MAKER should be capable of handling most general drawing tasks. After a GRM problem domain is set up by MAKER, another program, which is the main body of the prototype, is used to implement the rest of analysis tasks.

Once the problem domain is determined, attributes or parameters should be set to construct the simulation model. Attributes such as discharges or recharges from or to wells can be interactively set using the GRM prototype. For example, to add a well the analyst can click the mouse on the menu item 'Add' in a popup window (see Figure 6.2), a unfilled circle would be shown on the screen (see Figure 6.3). Then the circle can be moved using the mouse to a desired position, and a filled circle which indicates an added well would be shown (see Figure 6.4). By using a similar procedure (click and move), boundary conditions (filled squares), pumping or injection wells (filled circles), and check points (diamonds) can be added or deleted, and related information can be shown or edited.

Figure 6.5 shows a popup window with information related to the well at position (50,60). If withdrawal is positive, then the well is a pumping well. Otherwise, it is an injection well. Two options on the bottom are Demand and Reference. If Demand is selected, then the value entered for Withdrawal would be the lower bound of discharge which would be used in the optimization model. The option of Reference is for simulation analysis to examine the solution under the value specified. Figure 6.6 shows a popup window with information related to the boundary condition at the position (100,90). Two types of boundary

conditions can be used: specified head and specified flow. A specified head boundary is expressed by a filled square, and a specified flow boundary is expressed by a wide line (see Figure 6.7).

Figure 6.7 shows a popup window with information related to the check point at the position (50, 70). Check points are used for optimization models, and the Minimum Head of a check point sets the lower bound on hydraulic head at the check point.

Figure 6.8 shows a popup window for transmissivity and hydraulic head at the position (70, 90). Before a simulation model is solved, all heads are undetermined except at boundary conditions and all undetermined heads are indicated by '-1.0'. (Note the menu item on the middle of the second row indicates the kind of object, e.g. Well (see Figure 6.9) or Check point (see Figure 6.7), being examined.)

Since the groundwater simulation model used for this research uses a rectangular finite element method, it would be useful if elements can be shown on the screen. The grid option is thus designed to show and change the elements (see Figure 6.9). However, this grid option is different from the one used in MAKER. The latter is only a drawing aid for the analyst to use in constructing a problem domain, and the former provides an indication of the size of elements used for the simulation model as well as a drawing aid. If grid size is changed, then the sizes of elements would be changed too. After wells, boundary conditions, and transmissivities are determined, a simulation model is then set up. The interactive procedure described above is easy to learn, and the graphic object-oriented interface helps the analyst set up a simulation model quickly and efficiently.

Simulation Analysis and Optimization

After building a simulation model, the problem domain can be reduced in size so that the screen can be partitioned into four small subscreens and four models can be displayed at the same time. The one model being examined at a time is highlighted by a wide border (see Figure 6.10). The four screen display is very useful for making comparisons.

One sequence of steps for using optimization is: simulation analysis, objective(s) selection(s), model formulation, and then optimization. Simulation analysis would be implemented first to determine impact coefficients. Figure 6.10 shows a popup window of menu options in the sequence described. In the figure, the simulation model is being solved by selecting the top menu option. After a simulation model is solved, the impact coefficients from each well to all grid points are also computed (see Appendix A). The impact coefficients can be used not only for optimization models but also for re-computing the hydraulic heads on the domain without using the simulation program if only well discharges or recharges

are changed (see Figure 6.11). If a change is made in boundary conditions or transmissivities, then the impact coefficients would be different and the simulation model must be solved again.

After impact coefficients for all grid points are determined, it is necessary to specify the objective(s) of an optimization model. Three objectives available on the GRM prototype are cost, total drawdown, and total withdrawal (see Figure 6.12). The number of available objectives can be easily extended, but the display of a noninferior set for more than three objectives is difficult. After an objective is selected, an optimization model can be created by selecting the menu option 'Model Formulation.' The optimization model is shown in the new modeling language. Figure 6.13 shows an optimization model for a design named 'testcpt' which has six check points and two pumping wells. The objective is to maximize the total withdrawal from wells. At check point 2, a minimum hydraulic head is specified. The optimization model shown in Figure 6.13 is automatically created by the GRM prototype as the option 'Model Formulation' is selected. The analyst may want to add other constraints or objectives to the model. The additional constraints or objectives can be added by providing a file called 'constraints' (see Figure 6.14). The constraints or objectives should be described in the developed modeling language. Figure 6.15 shows the model with an additional constraint.

After the optimization model is created, it can be solved. The optimization, by XMP [Marsten, 1984], can be accomplished by selecting the menu option 'Optimize'. The objective value and solution vector are then shown in the message area which is the top row in the GRM prototype user interface (see Figure 6.16). Other objectives can also be included. Figure 6.17 shows a two objective GRM model. The optimization for a two objective model is also accomplished by selecting the menu option 'Optimize.' However, the result of optimizing a two objective model is different from that of a single objective model. Instead of showing only one optimum, a tradeoff curve is shown in one of the four screens (see Figure 6.18). Similiarly, an additional objective can be brought into the model. Figures 6.19 and 6.20 show a three objective GRM model and the noninferior set of the model. Although the 3-D noninferior set is difficult to understand, it can be improved if shading software and a high resolution color monitor are used.

The display of a noninferior set provides a presentation of results and an interface to examine noninferior solutions. For example, the tradeoff curve shown in Figure 6.18 can be used as an interface. The analyst can move the mouse cursor along the tradeoff curve, and the message area will show the object vector associated with the points on the noninferior solution set. The analyst selects a noninferior point to be examined by clicking the mouse button; the objective vector of the desired noninferior point is shown in the message area (see Figure 6.21 and 6.22).

Different alternatives can also be generated if there exist important unmodeled issues. This step is demonstrated below.

Generation of Alternatives

By comparing a variety of alternatives, the analyst can examine tradeoffs with unmodeled issues. The MGA method is developed for generating maximally different alternatives. In this research, only the HSJ method is used because of its simplicity, although there are many other options that could be used. As shown in the message area in Figure 6.23, each time the menu option 'Generate Alternative' is selected three alternatives would be generated. The alternatives can be then evaluated and compared.

Presentations

Many of the presentations have been demonstrated in the previous figures (e.g. noninferior sets, grids, models in the new model language, and graphic objects for attributes). These presentations are intended to help the analyst perceive a result or easily implement a task. In this subsection, the presentation for transmissivities and hydraulic heads is described. In Figure 6.24, the hydraulic heads for the model 'testcpt' are represented on the top right screen. The darker areas indicate higher head values; and vice versa. Although this relative display does not give the exact value of each head, the direction of subsurface flow can be easily observed. Similarly, the transmissivities on the domain can be easily compared by a similar display (see Figure 6.25 where constant transmissivities are used and thus no difference exists.). The relative display is designed to be useful for comparisons. The drawdowns and significant impact can be easily detected (see next subsection).

Comparisons

Making comparisons is very important in a decision making process. From the experience of the earlier work by Brill et al. [1989], a decision maker can only compare a small number of alternatives at a time. In this research, four subscreens are used to show up to four alternatives at a time for comparisons. In the GRM prototype, the menu option 'Compare Alternatives' can be selected, and a popup up window is shown for a list of alternatives. Figure 6.26 shows a list of 4 alternatives for a single objective GRM model, and hydraulic heads of three (1, 2, and 3) of them are represented with shading for comparisons. A different set of alternatives can be selected such as in Figure 6.27 where alternatives 4, 3, and 1 are shown. Figure 6.28 shows a list of 6 alternatives for a two objective GRM model, and hydraulic heads of

four(3, 4, 5, and 6) of them are shown. The four subscreens can be used not only for comparing alternatives but also for comparing different models, e.g. Figure 6.29 shows four models with different configurations.

Other Prototype Features

Several other important features of the GRM prototype are briefly described below.

Figure 6.30: the initial display of the GRM prototype to get the analyst's name. All files created by a given analyst would be saved by his name (see Appendix B for program structure).

Figure 6.31: Models saved under Group 2 are shown. The model management capability provides a simple interface to retrieve models. Different models can be grouped in any way the analyst desires.

Figure 6.32 and Figure 6.33: the model 'testbc' on the top right subscreen is selected as the working model. The working model can be easily changed by selecting a subscreen from the popup window.

Other options are:

Grid size (x and y): the corresponding scale of a real problem for each rectangle element;

OPEN: to open a new mode to be examined;

SAVE: to save the working model;

RESTART: to ignore all changes made on a working model and reset it to its initial state;

TERMINATE SESSION: to quit the work session.

6.3. Summary

This chapter provides an overview of the GRM prototype based on tasks expected in a decision making process. The prototype incorporates mathematical techniques, a modeling language, graphical displays and friendly user interfaces. The prototype is intended to reduce the complexity associated with the modification, generation, and presentation of a model, solution, or alternative. If the user can work more efficiently and effectively, then the decision making process should be improved.

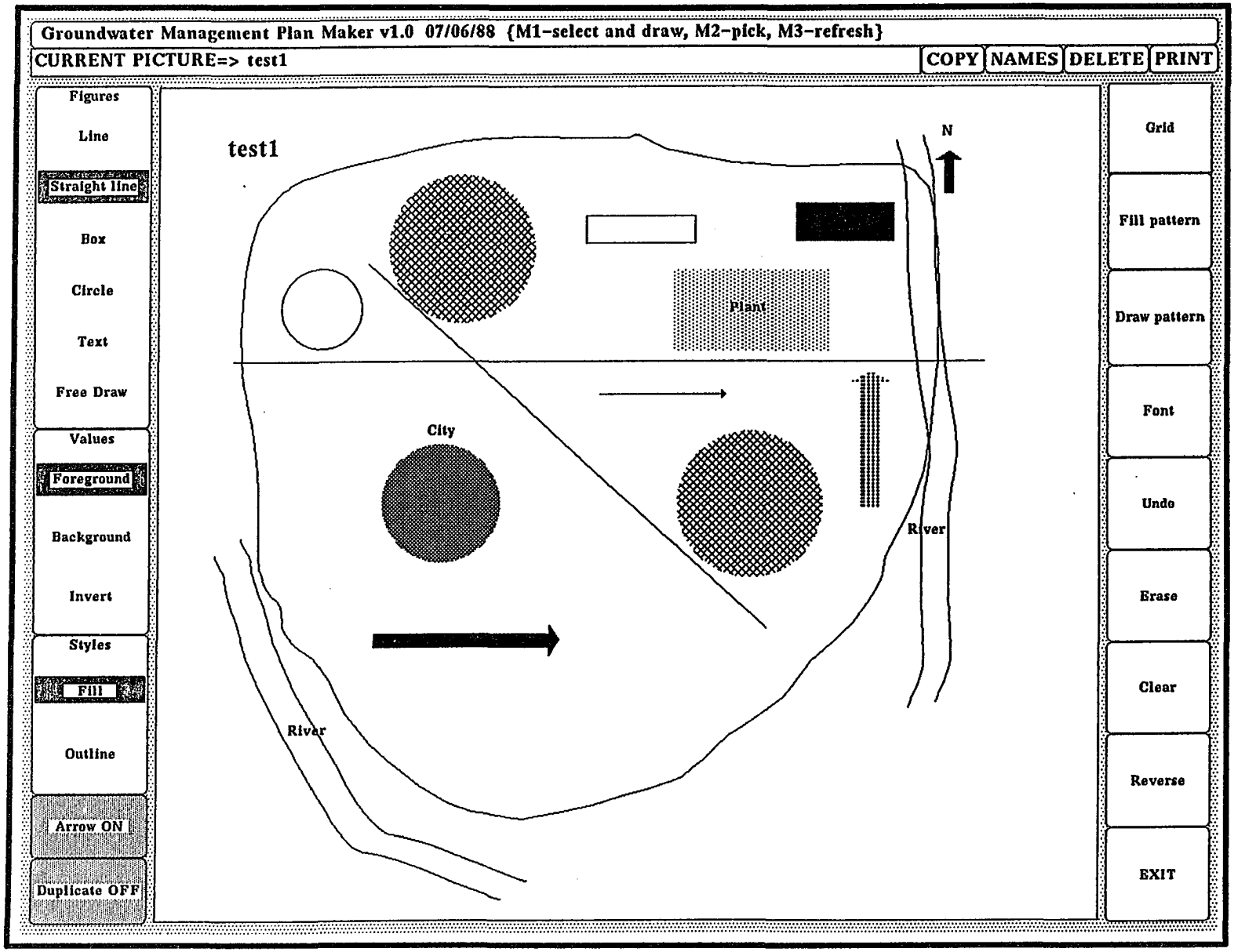


Figure 6.1

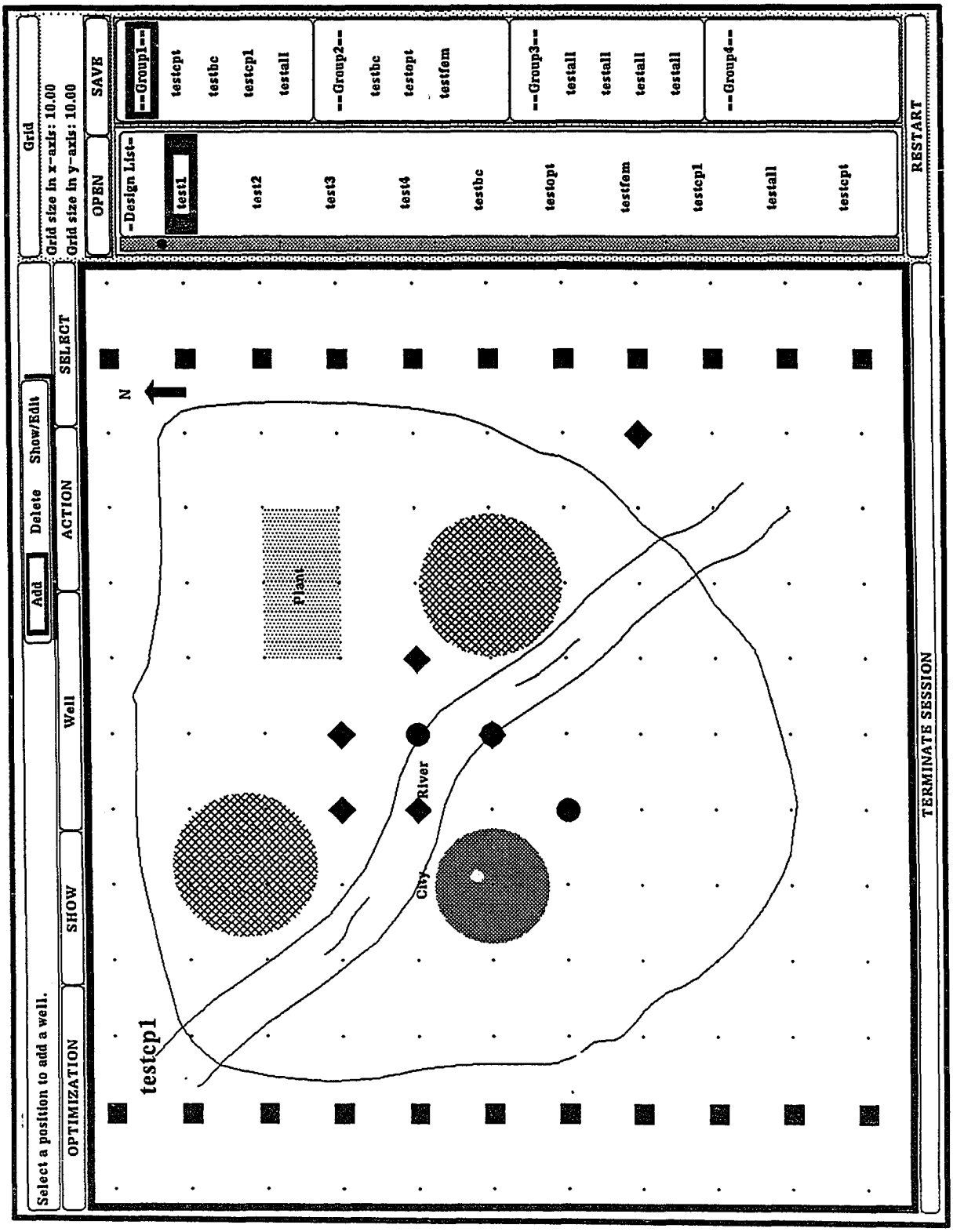


Figure 6.2

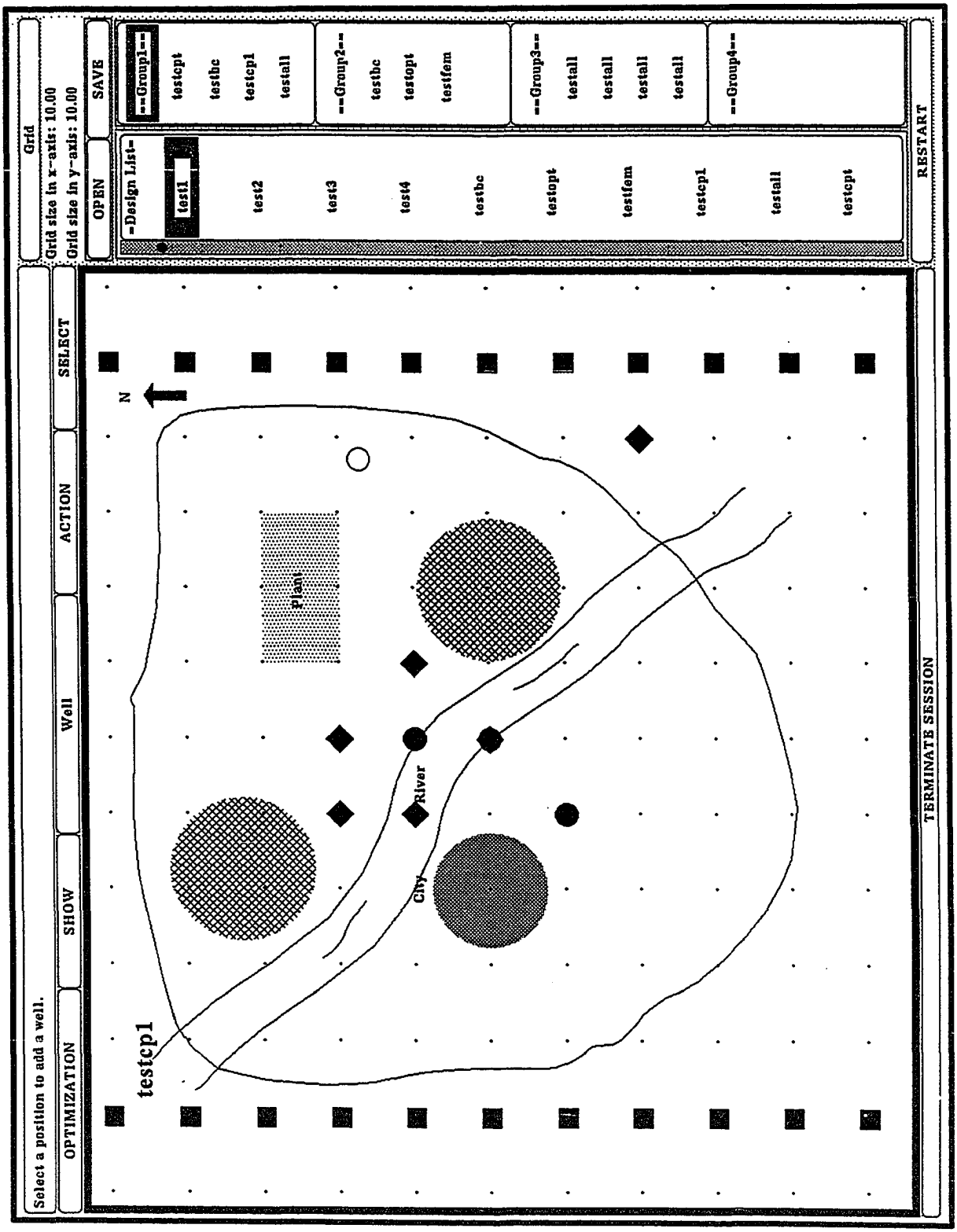


Figure 6.3

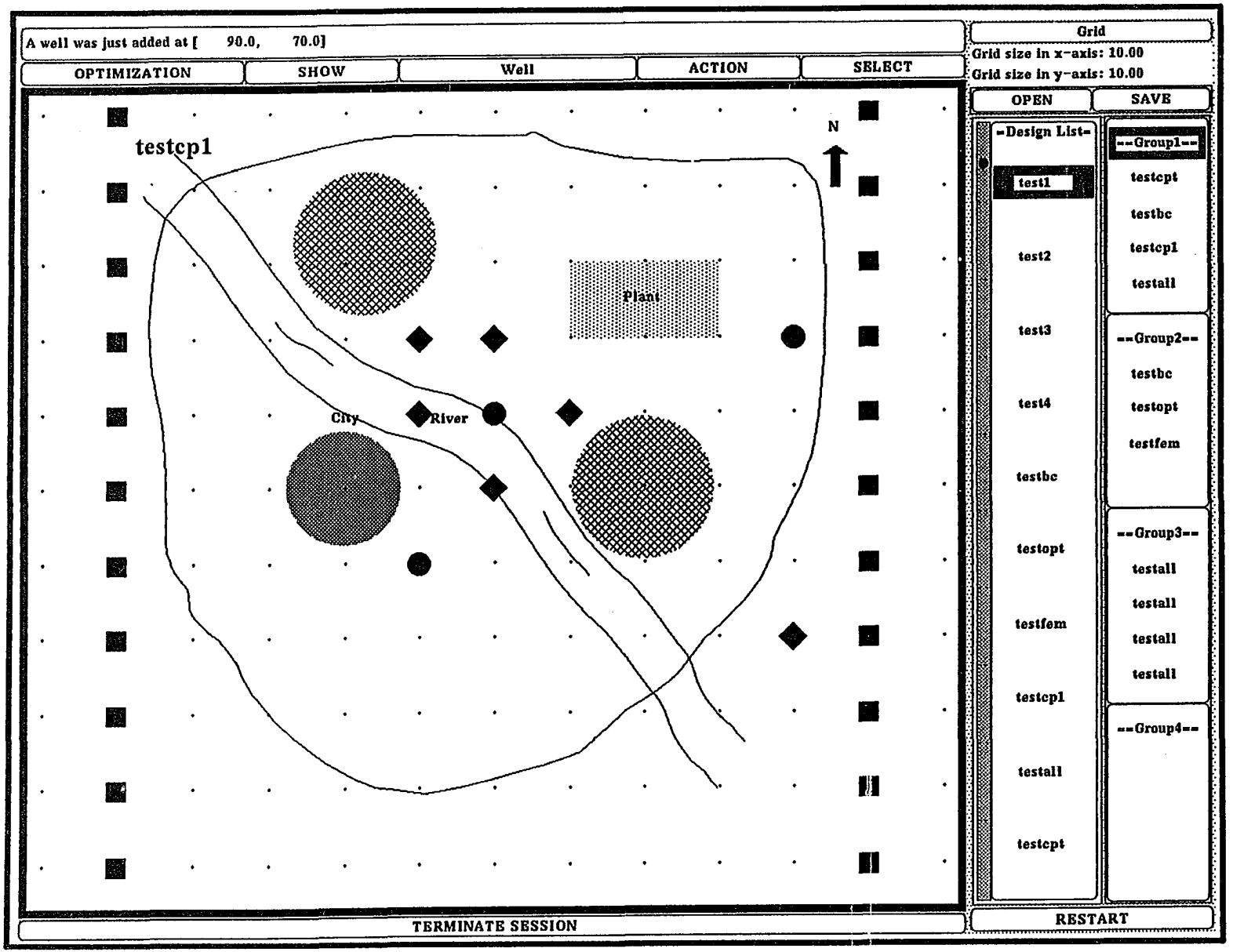


Figure 6.4

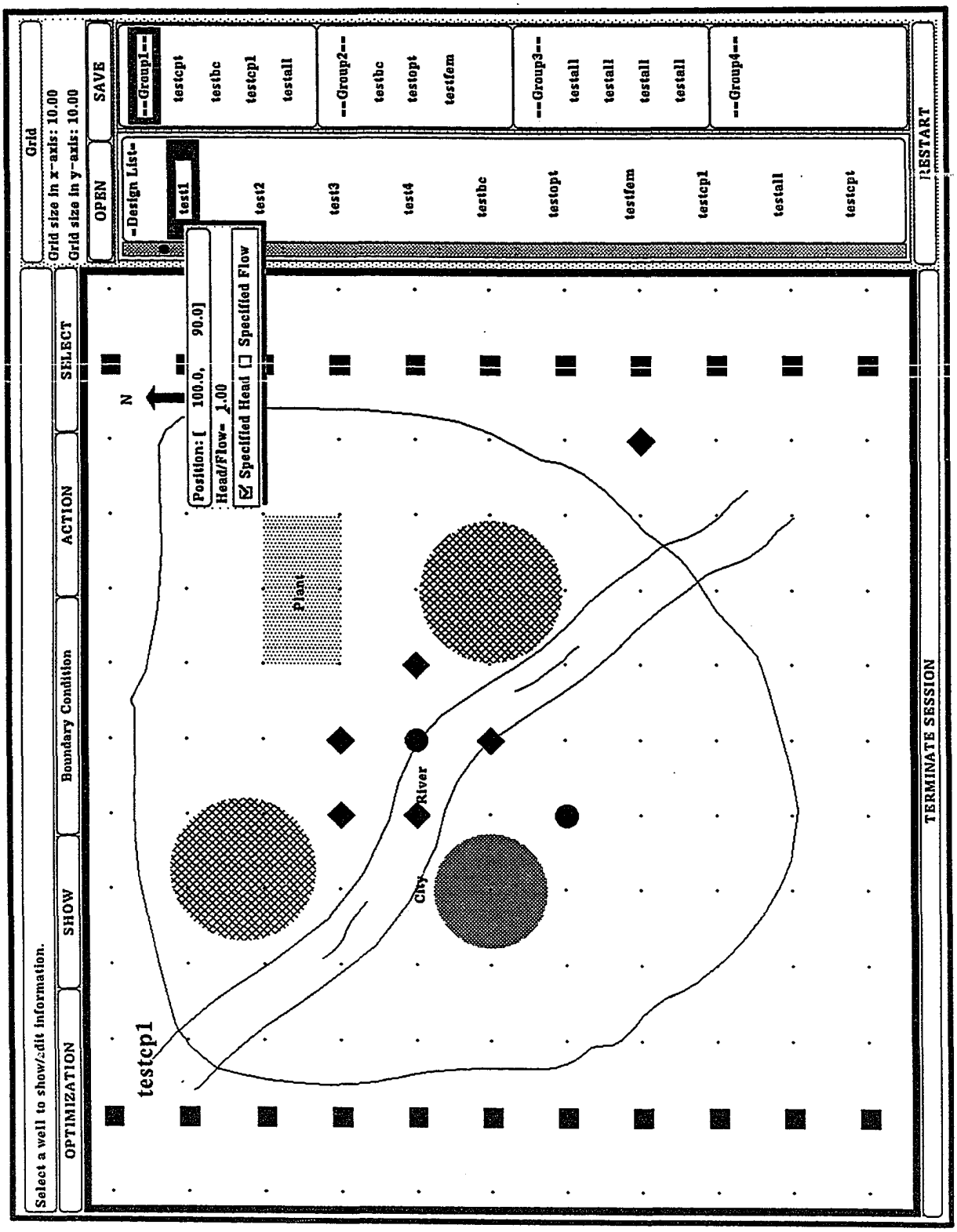


Figure 6.5

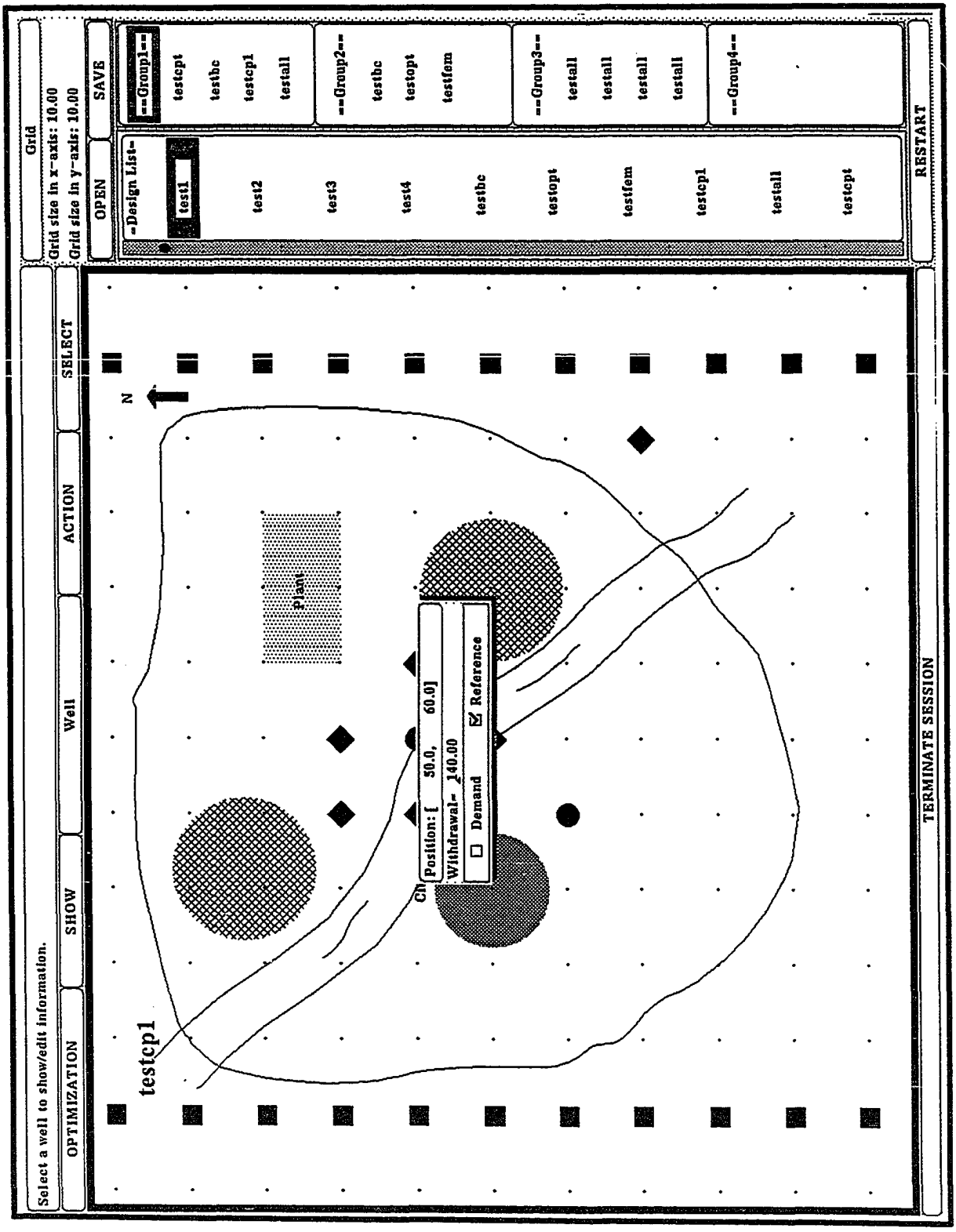


Figure 6.6

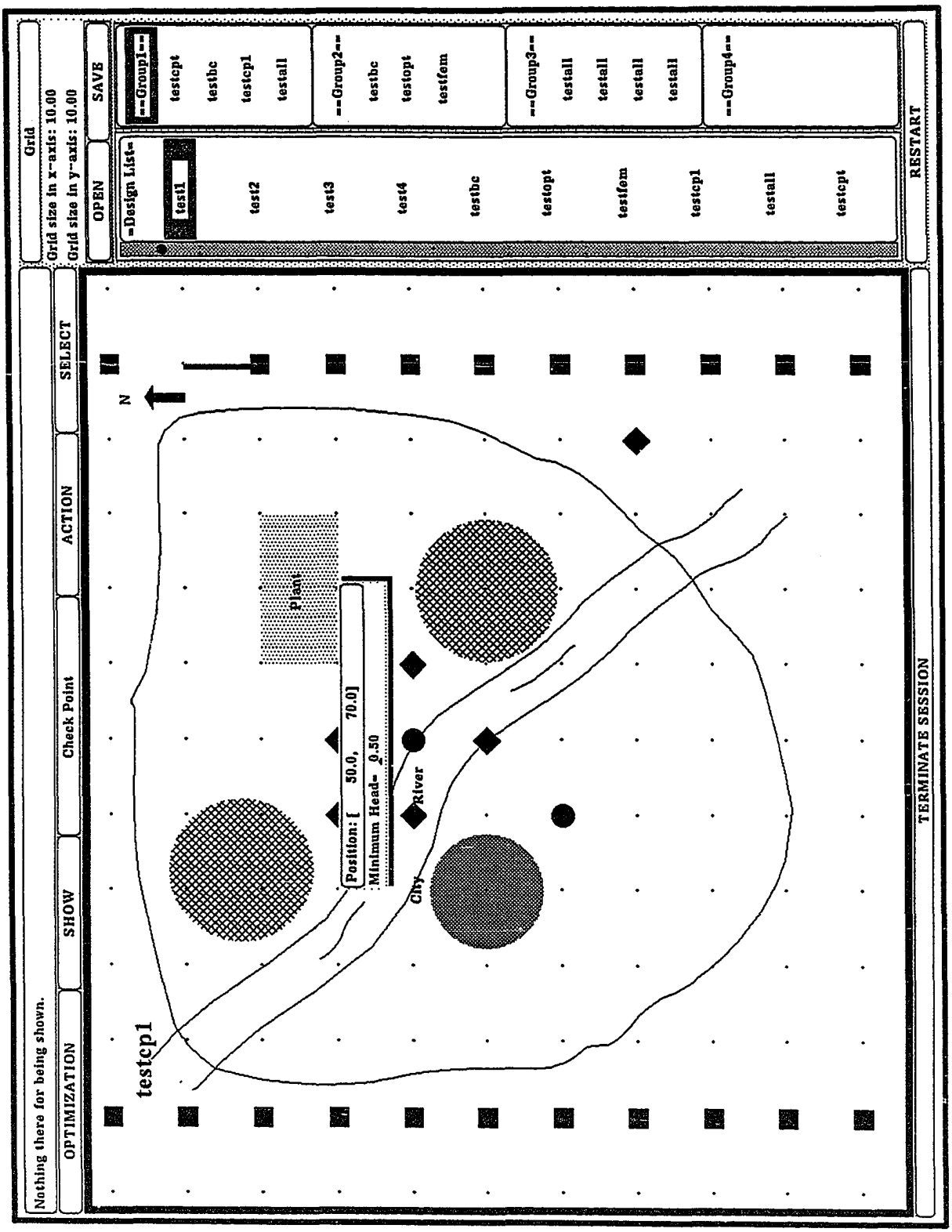


Figure 6.7

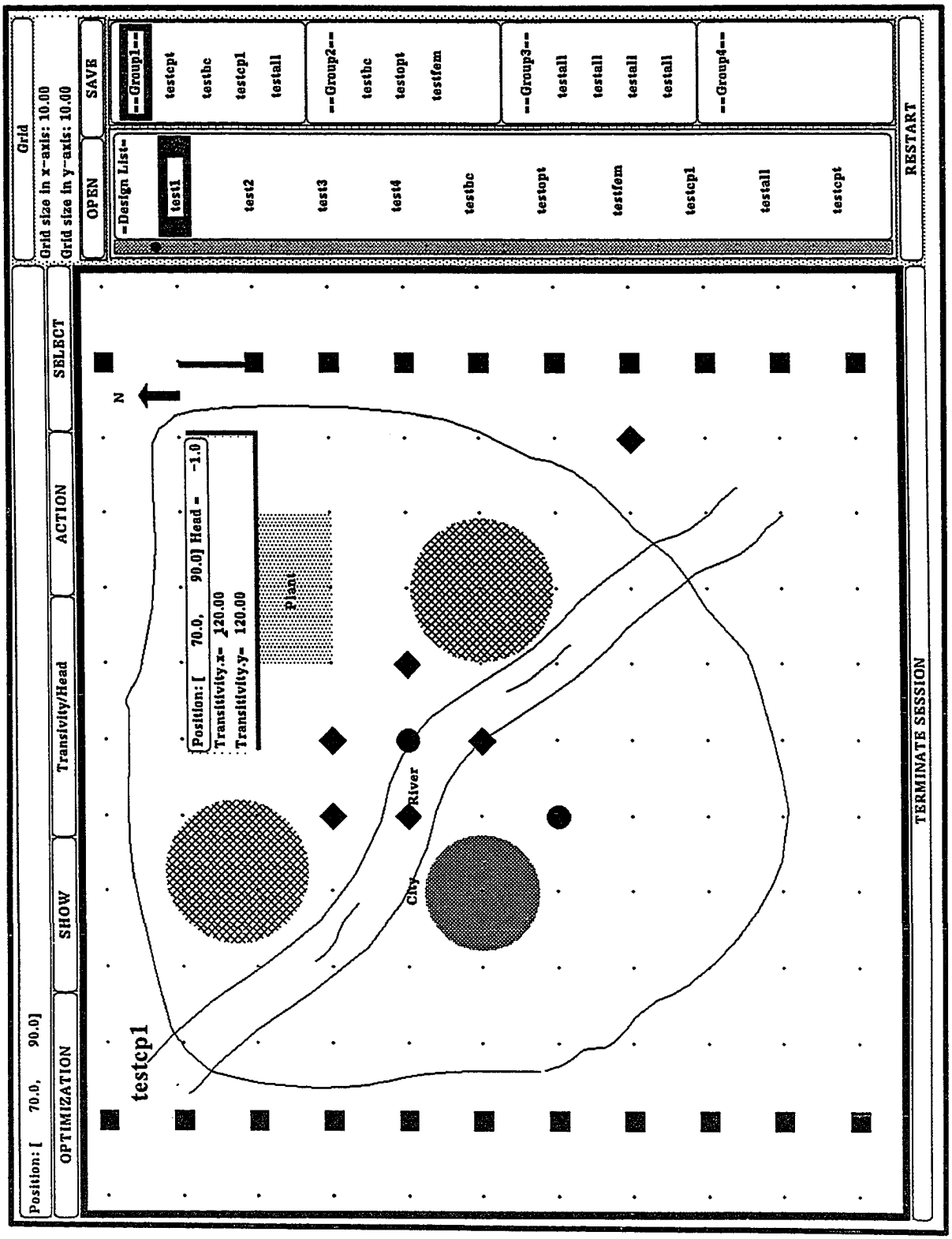


Figure 6.8

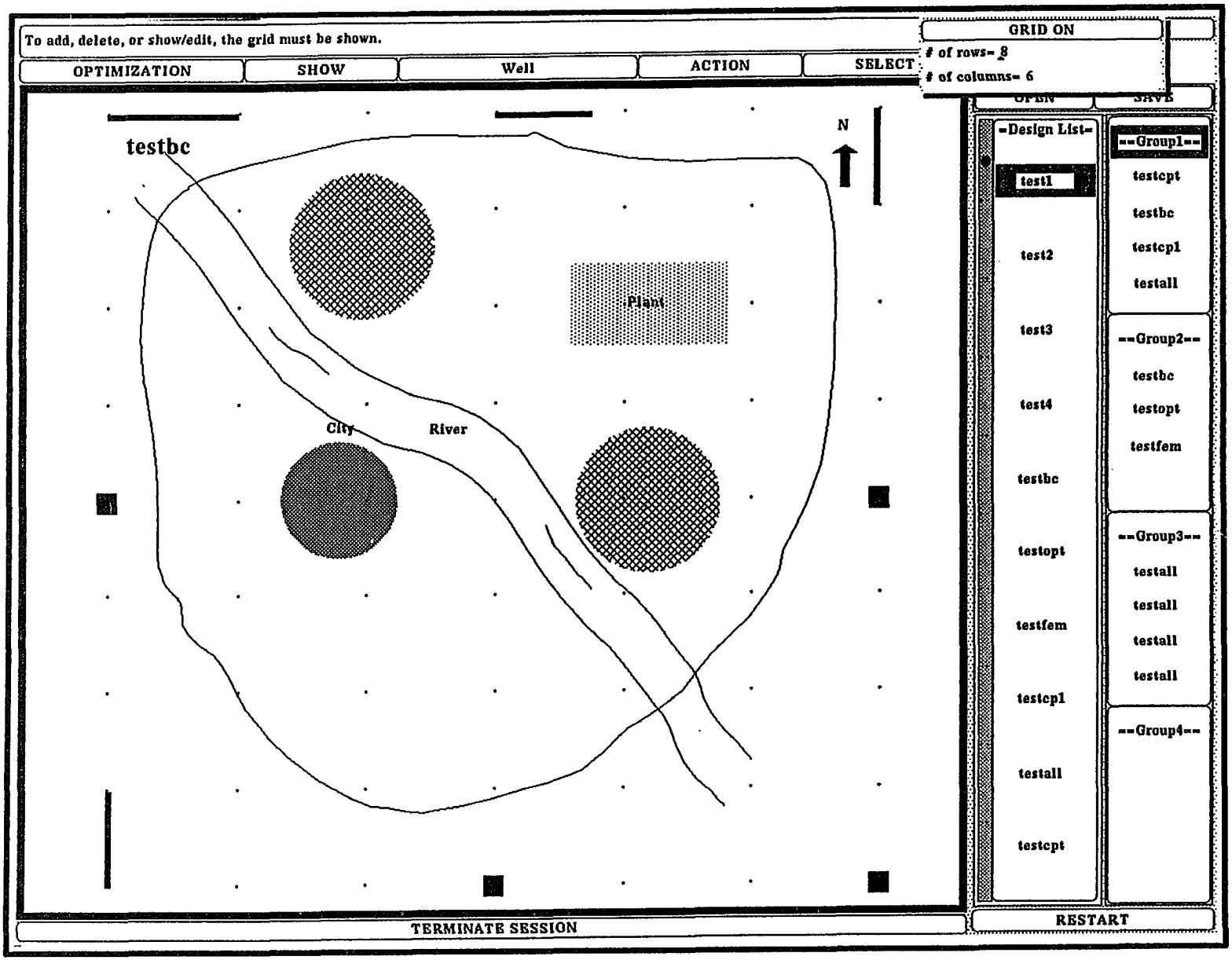


Figure 6.9

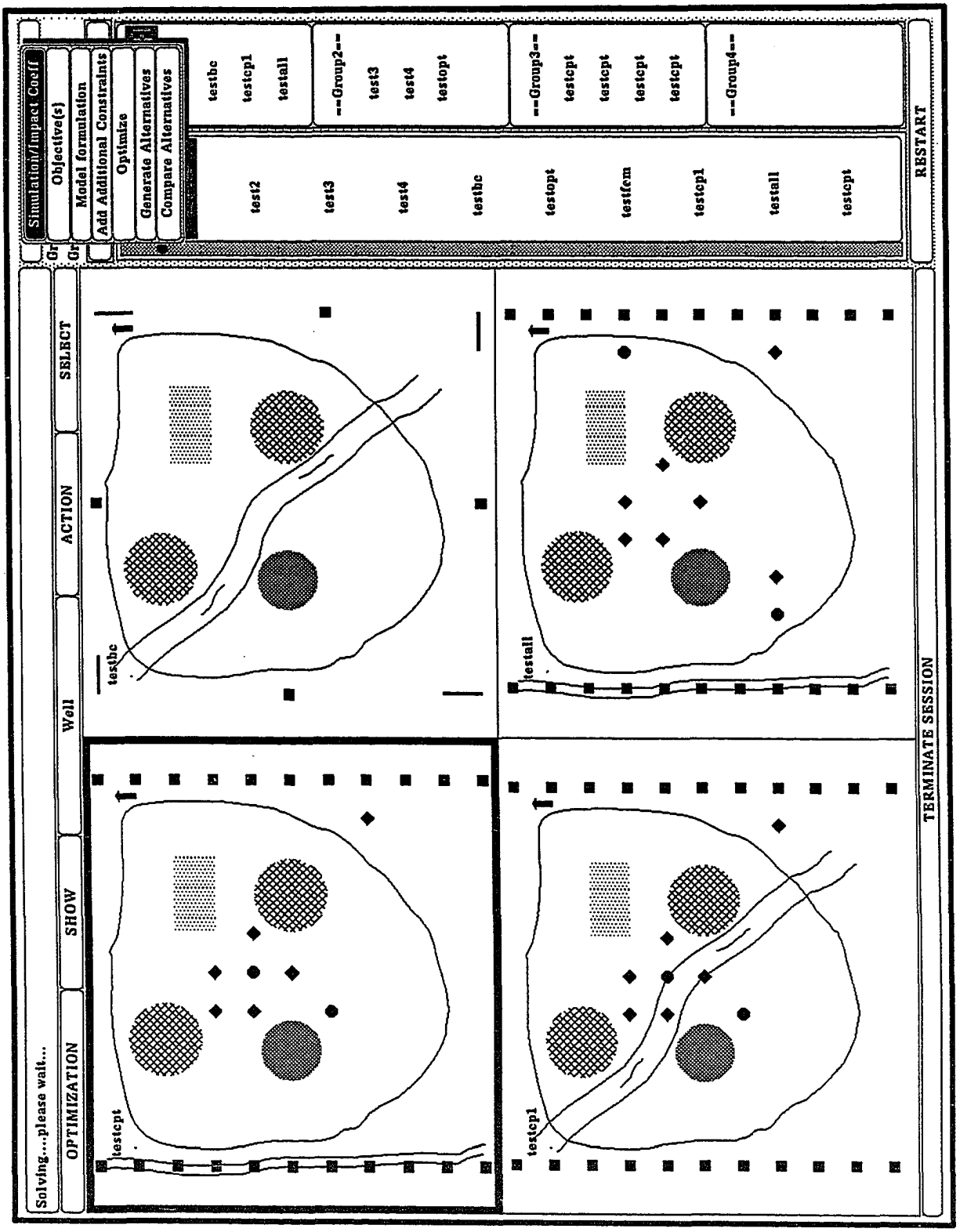


Figure 6.10

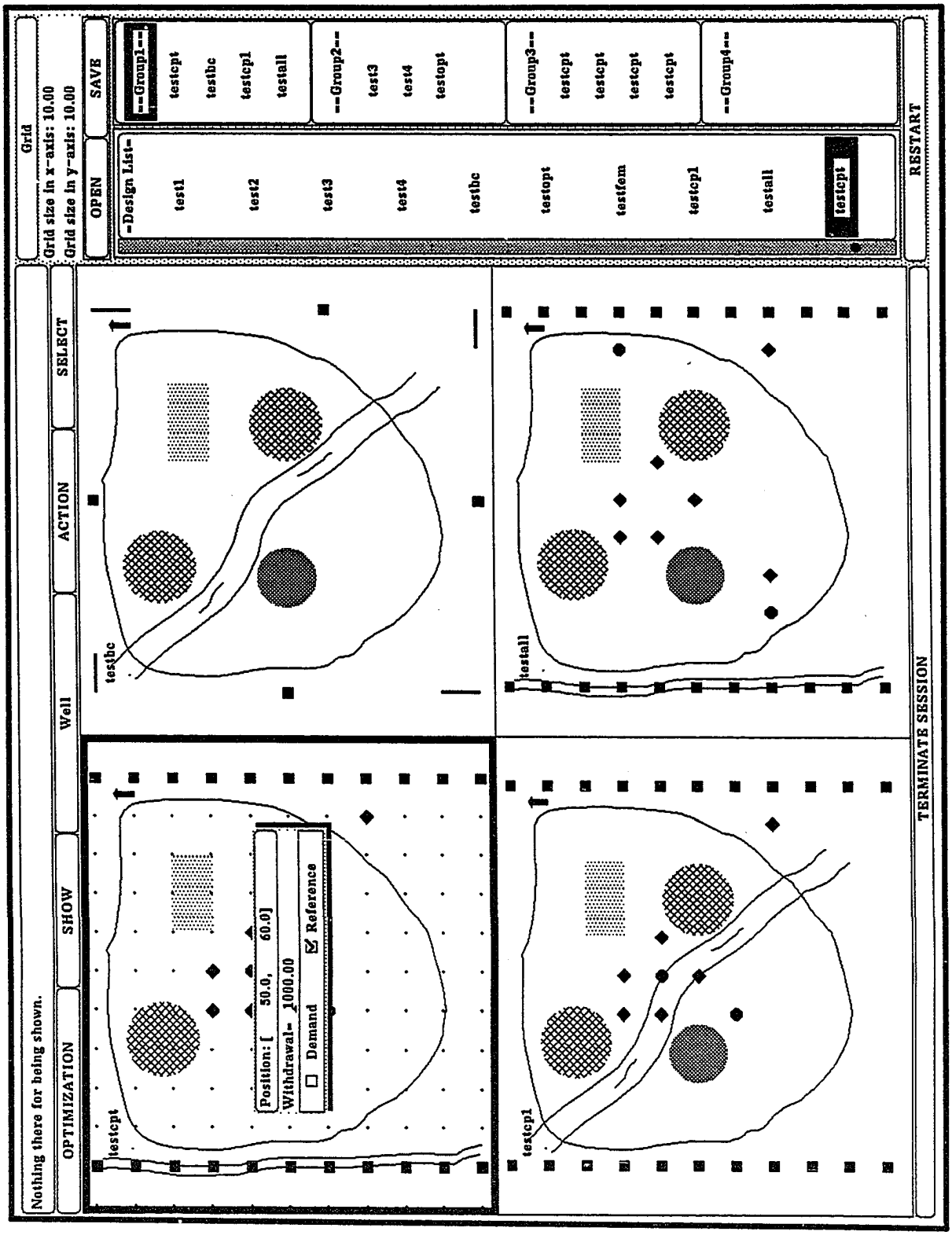


Figure 6.11

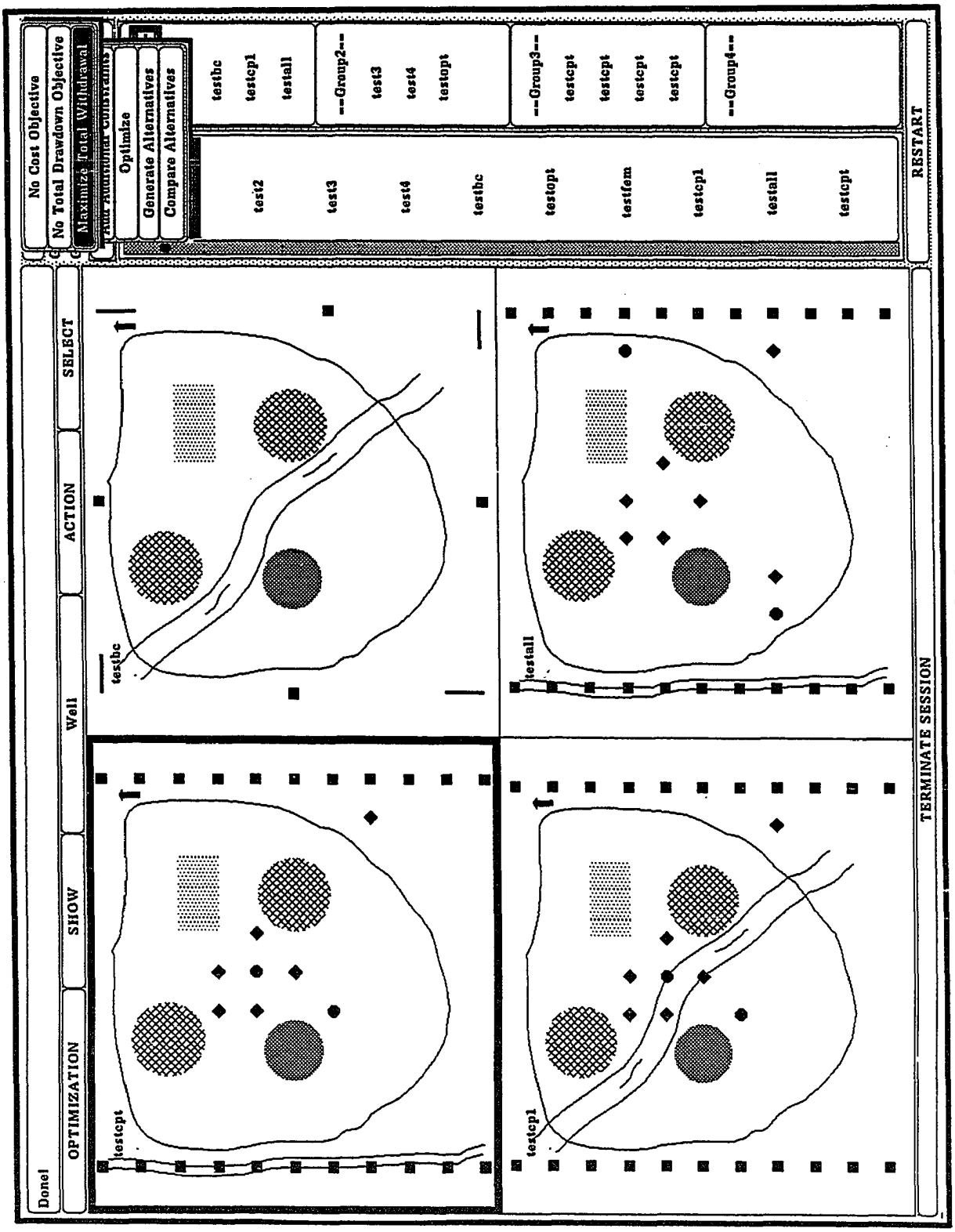


Figure 6.12

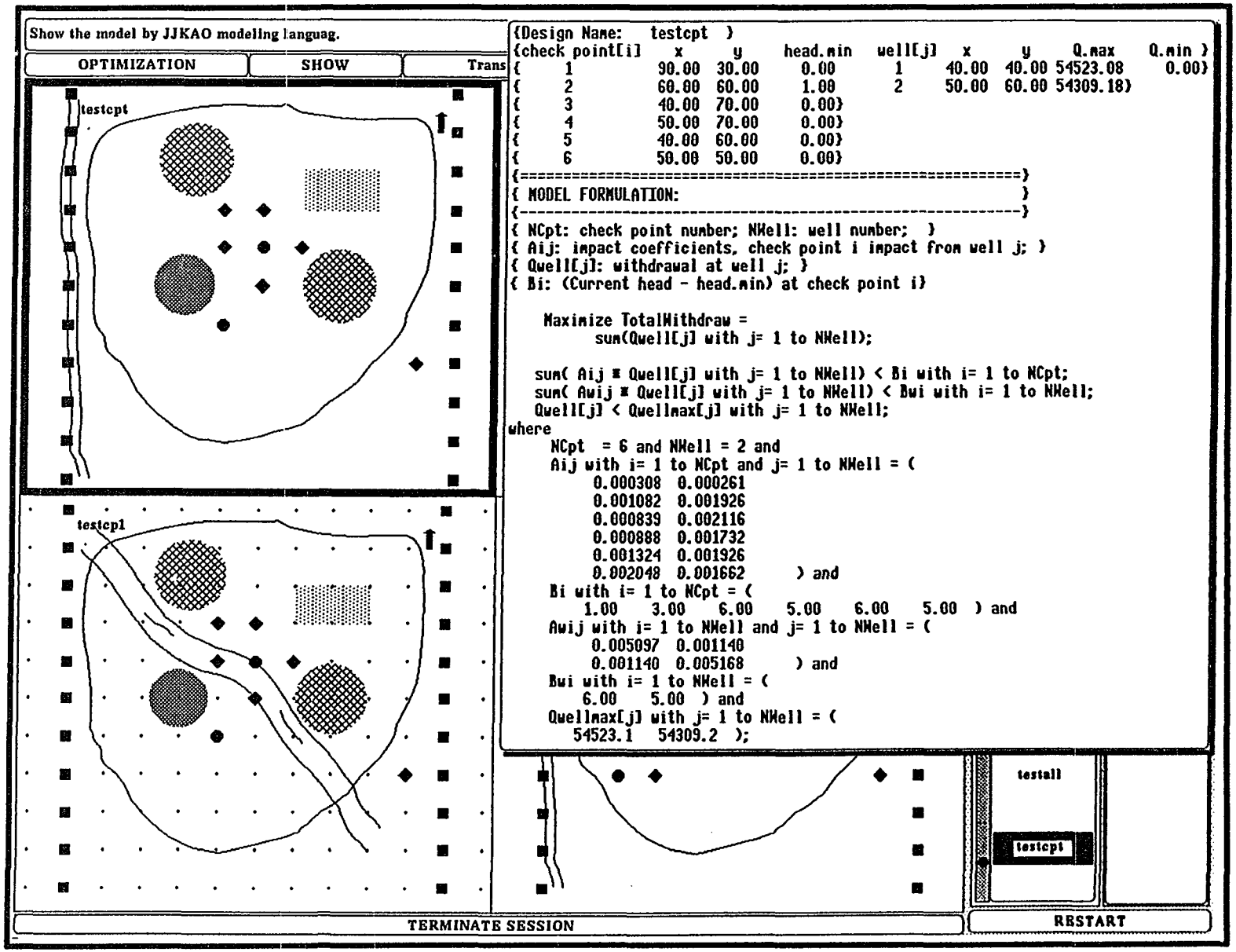


Figure 6.13

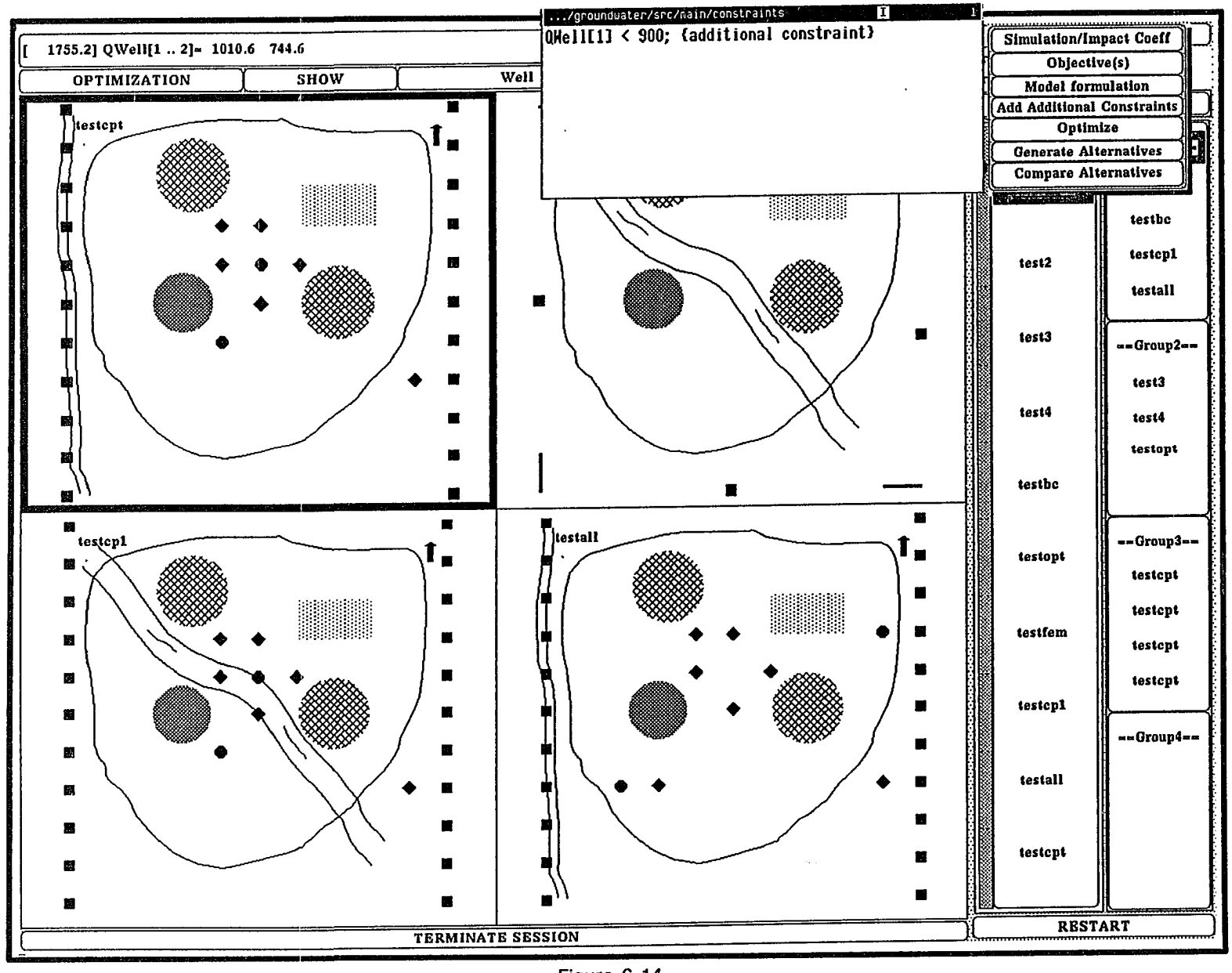


Figure 6.14

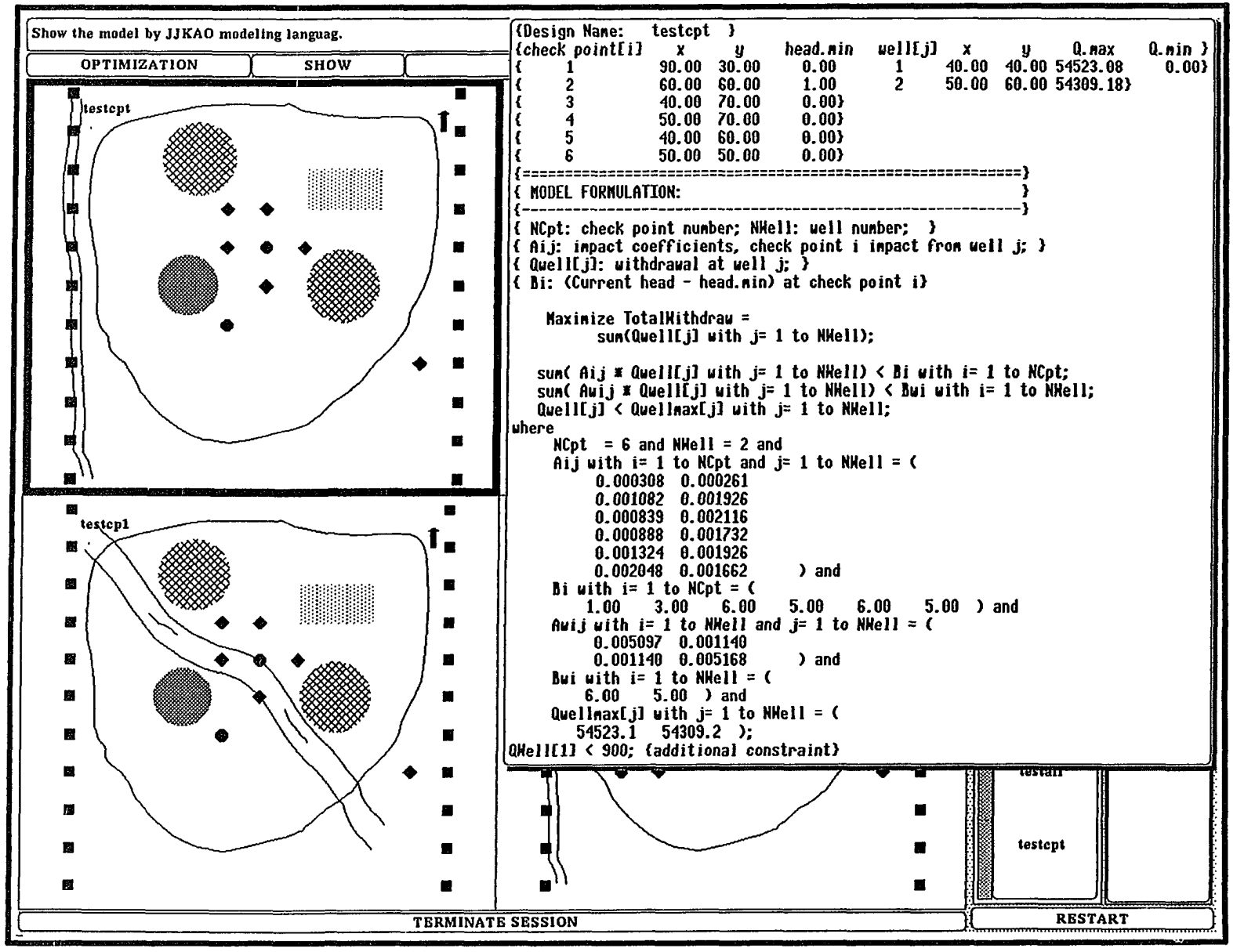


Figure 6.15

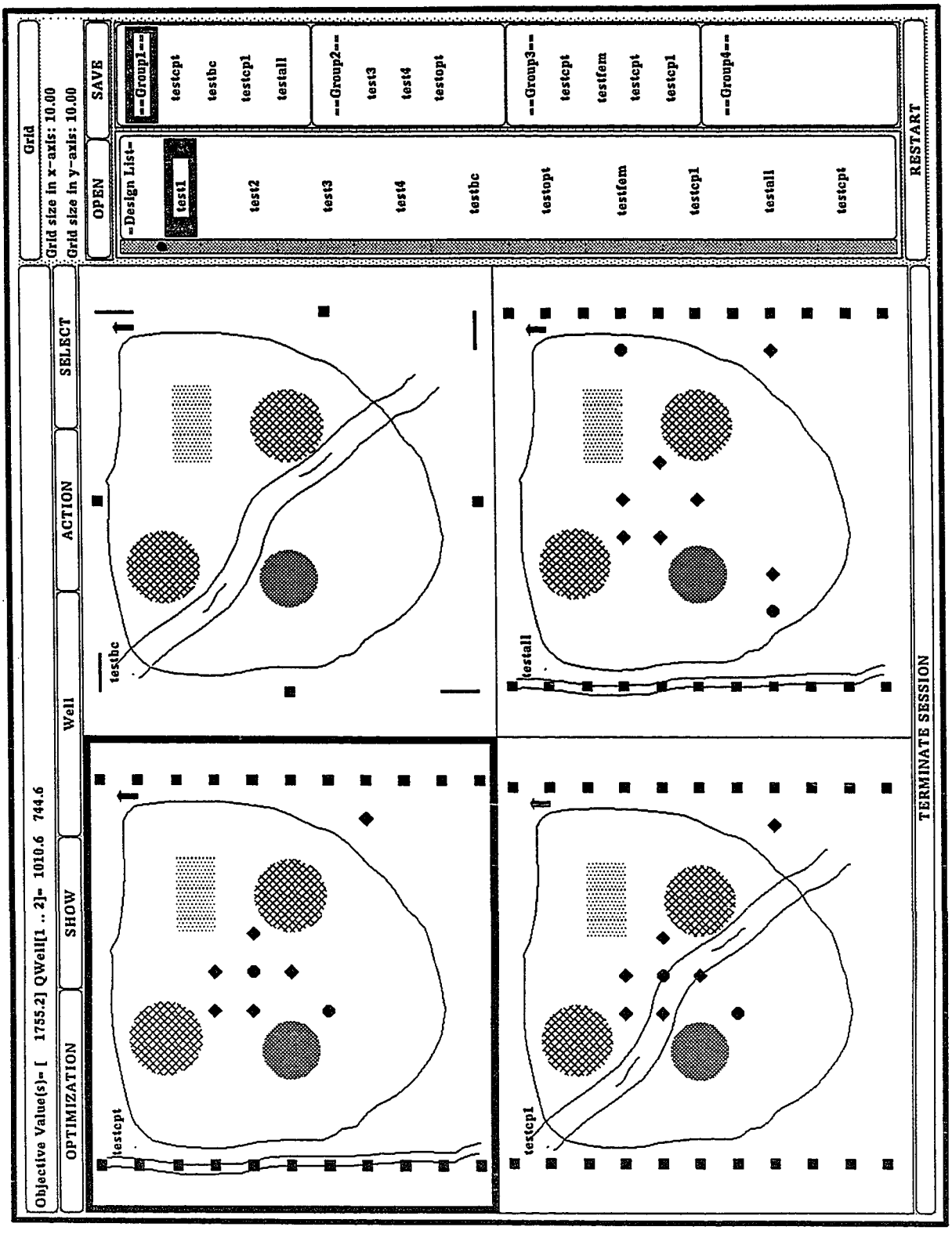


Figure 6.16

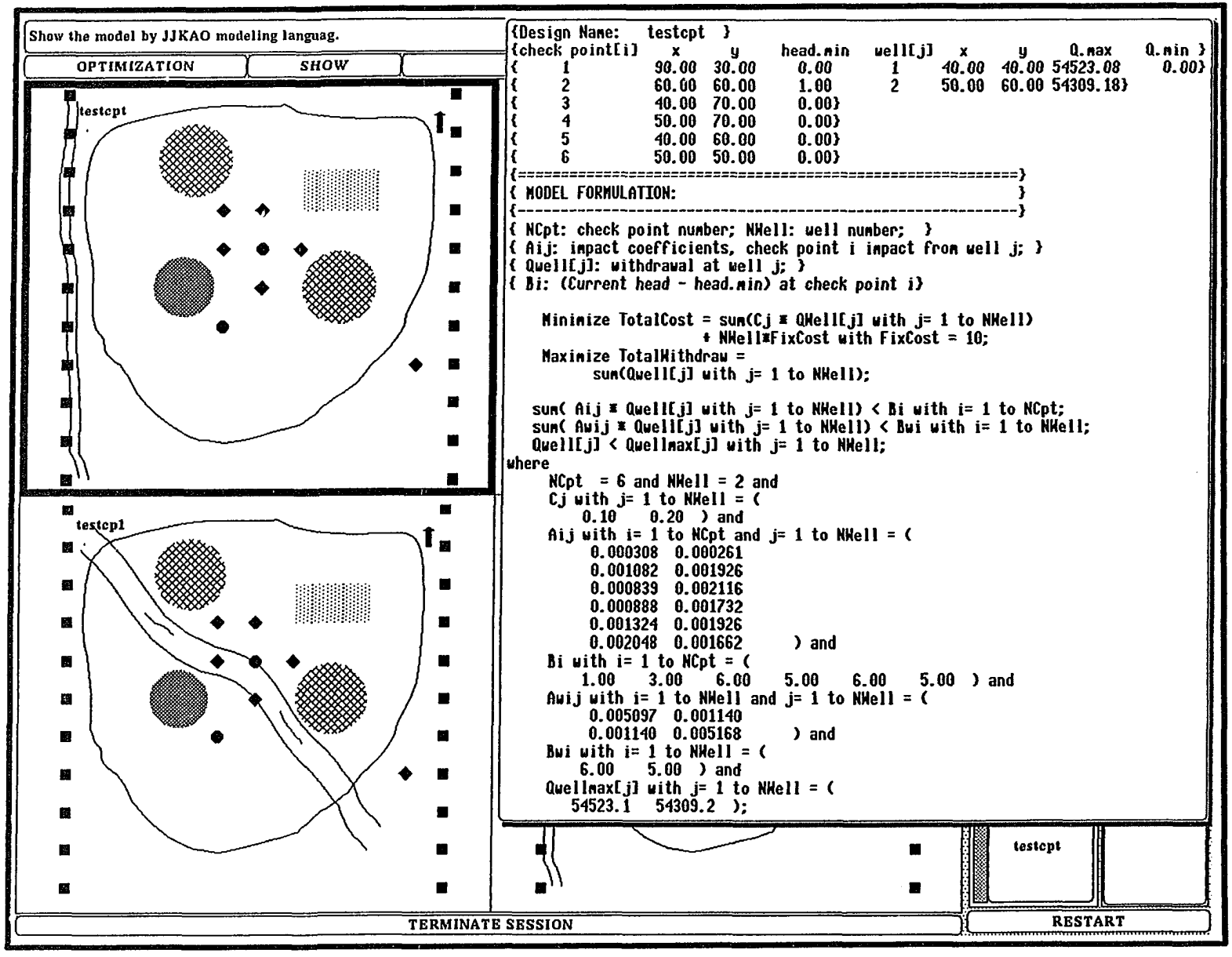


Figure 6.17

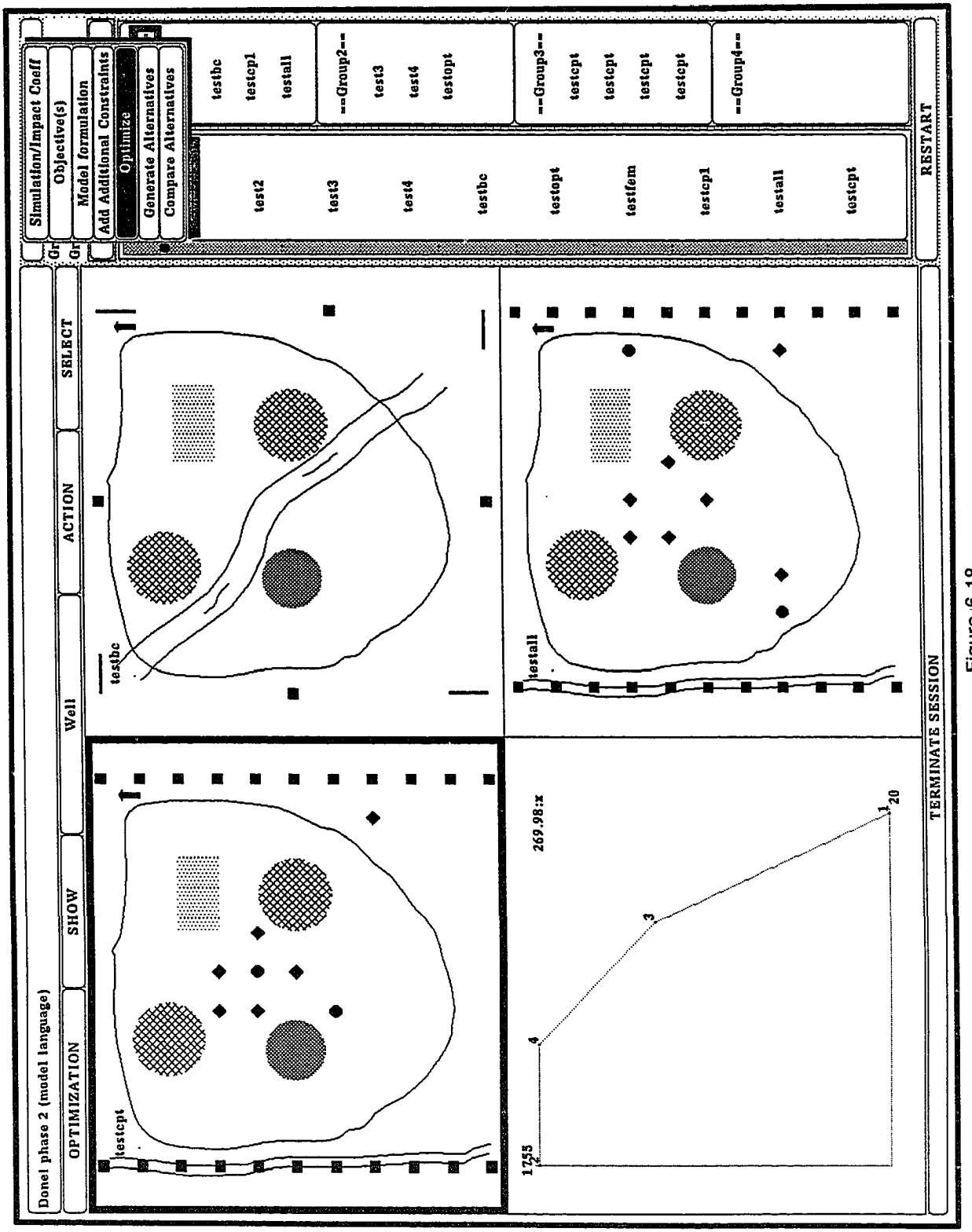


Figure 6.18

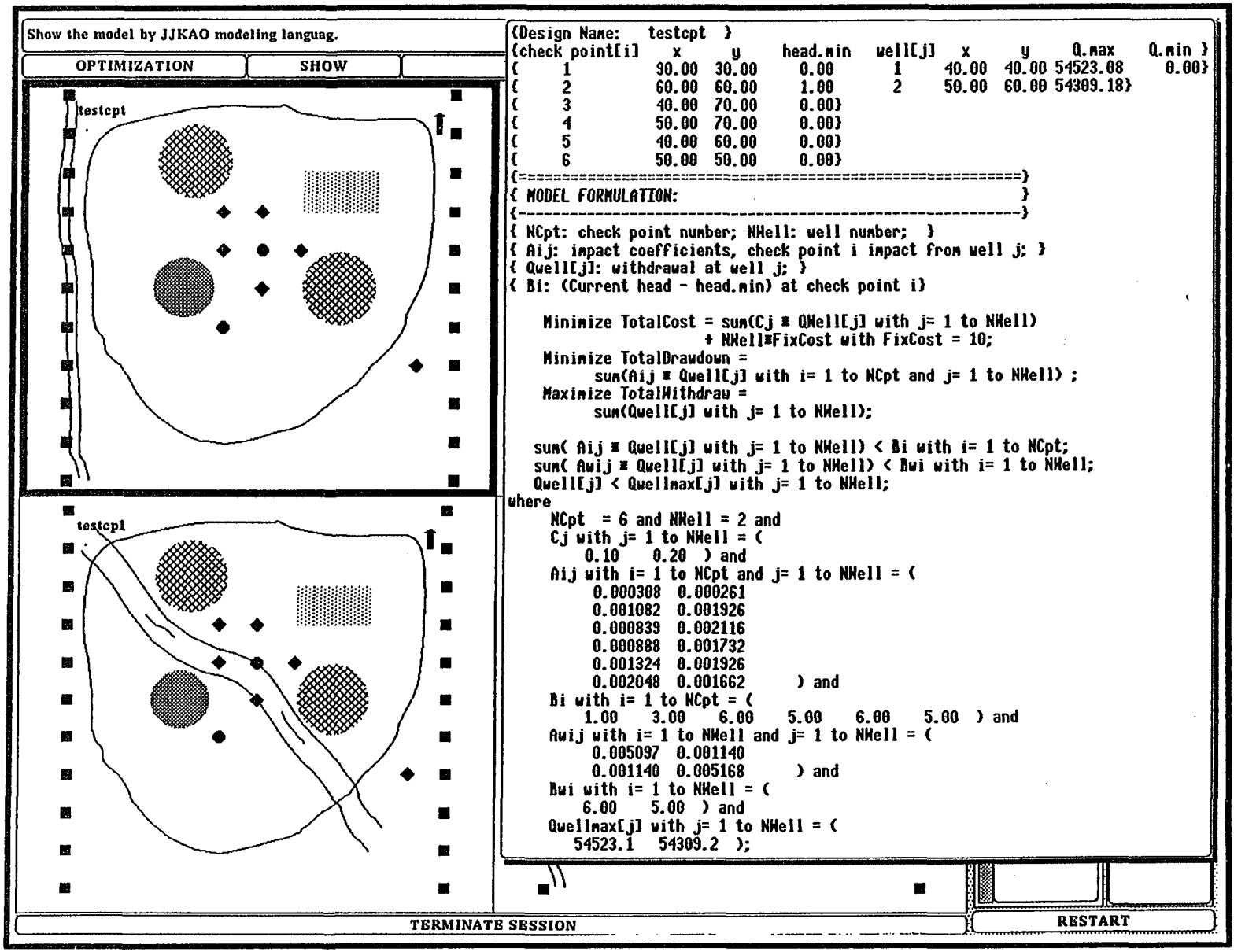


Figure 6.19

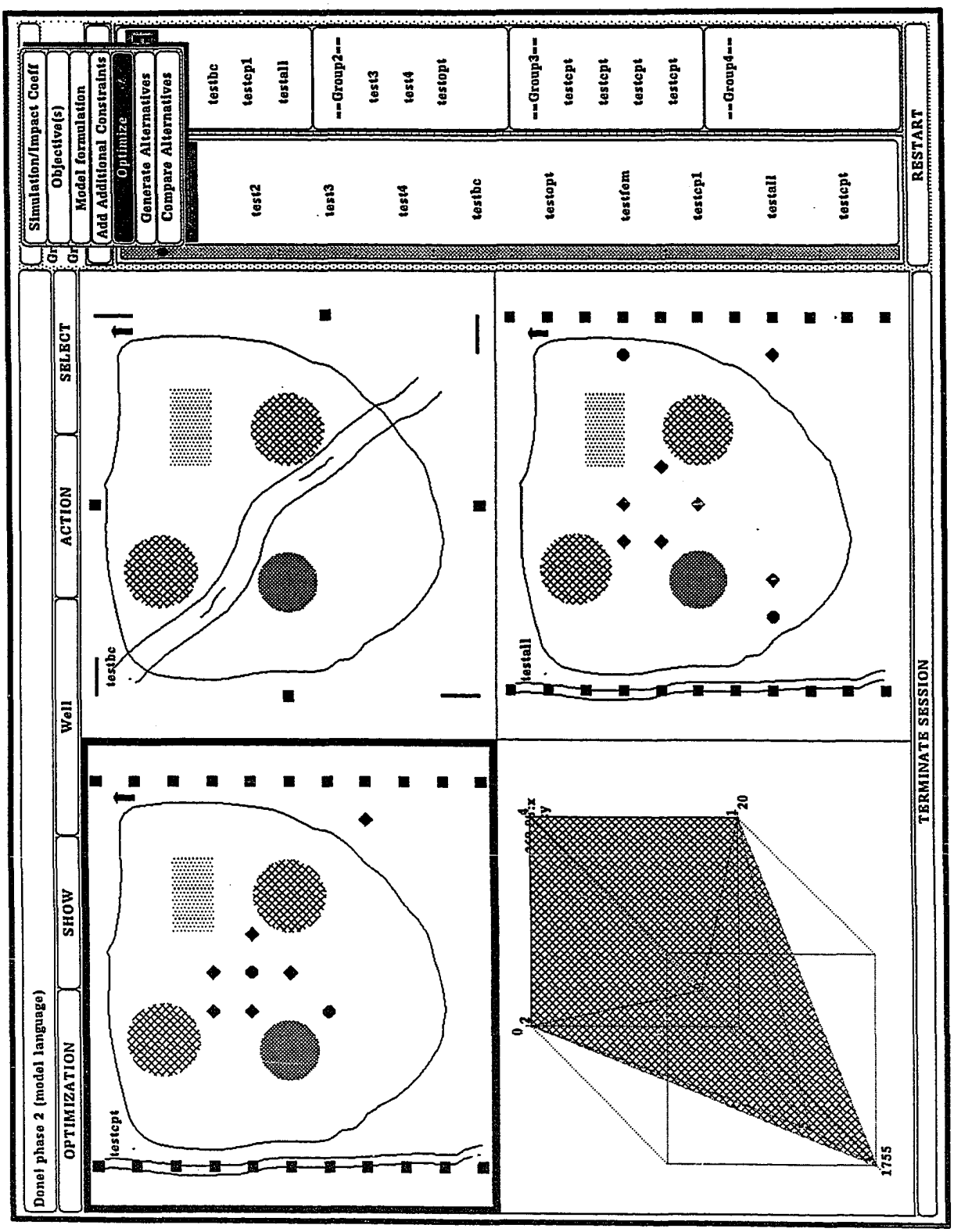


Figure 6.20

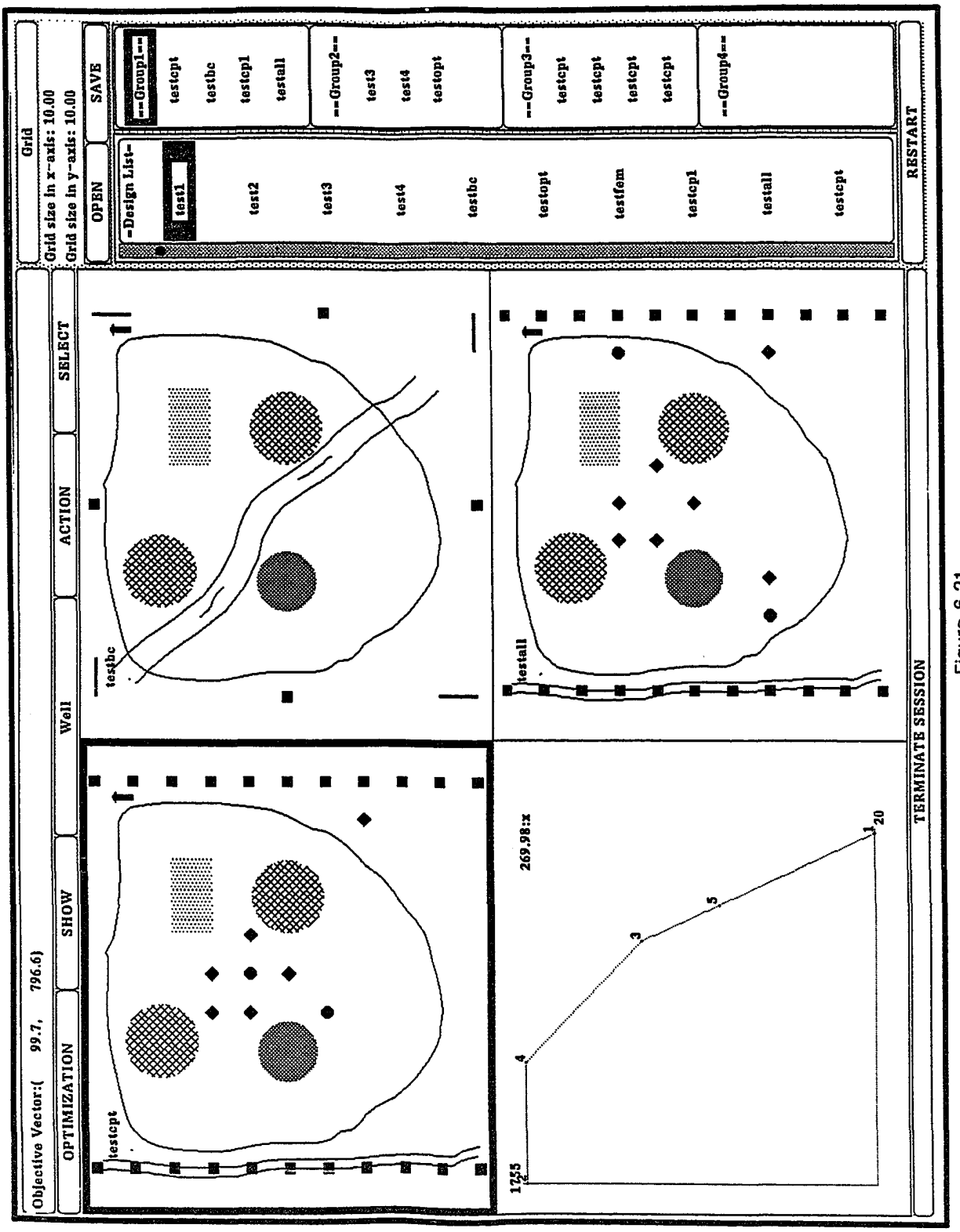


Figure 6.21

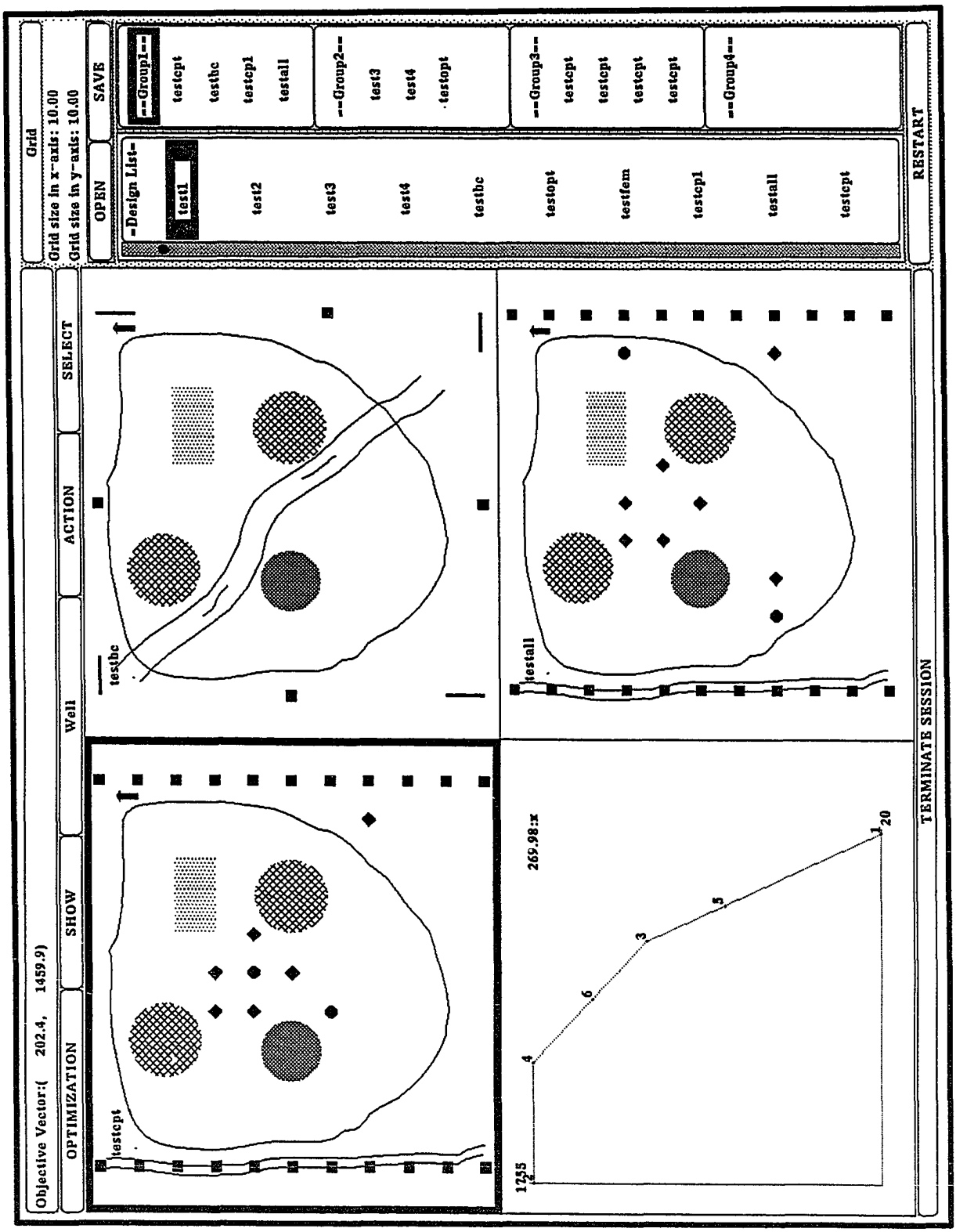


Figure 6.22

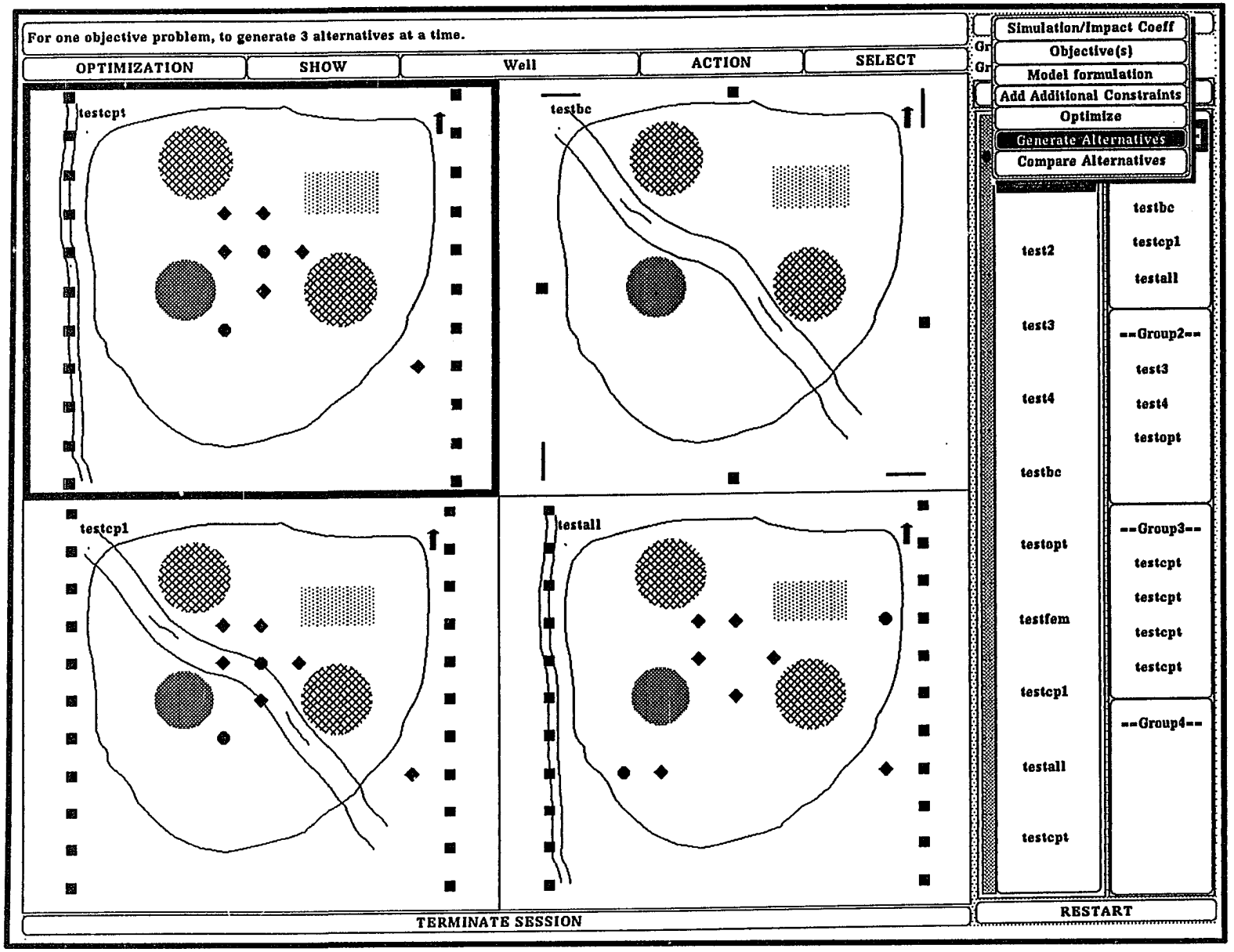


Figure 6.23

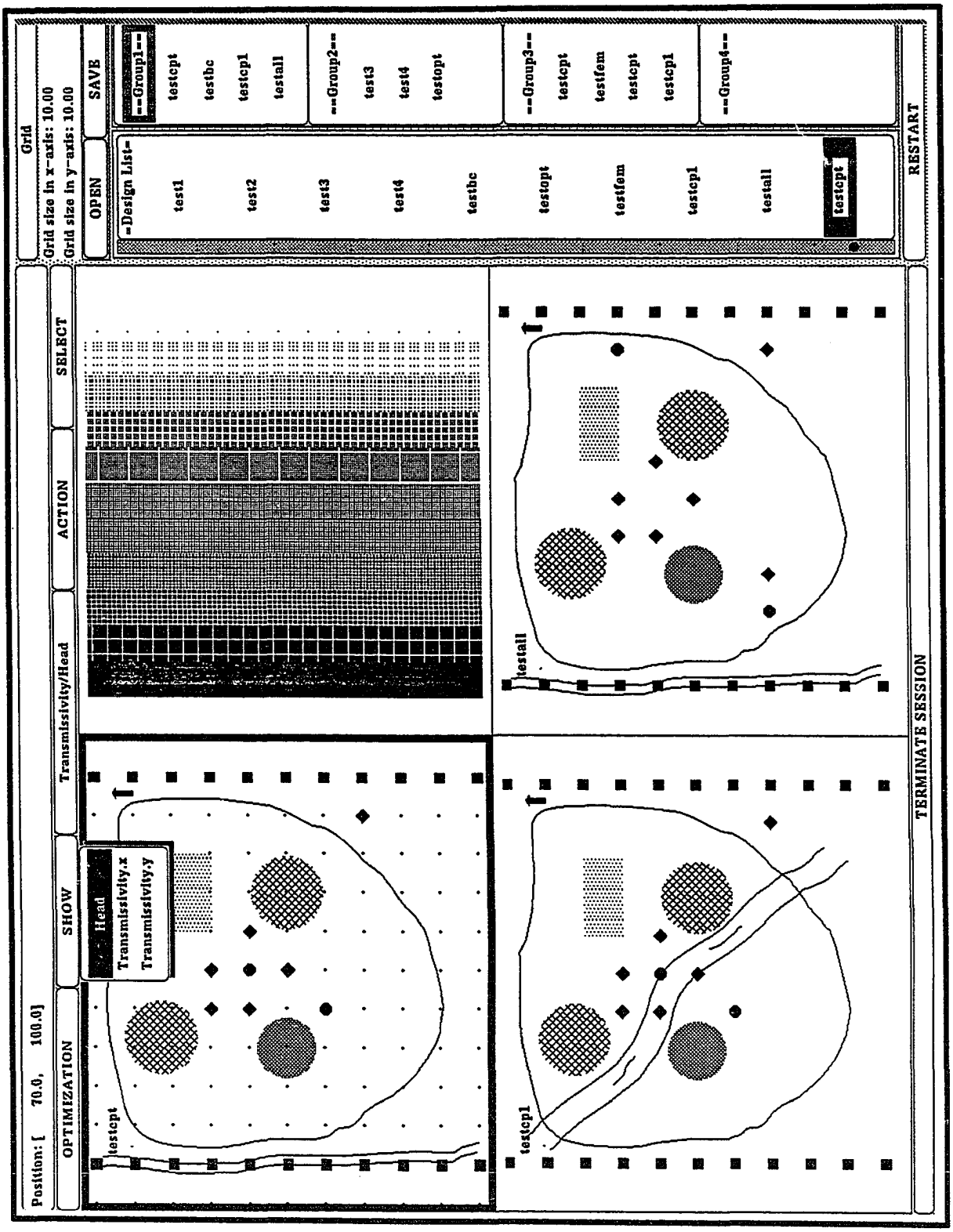


Figure 6.24

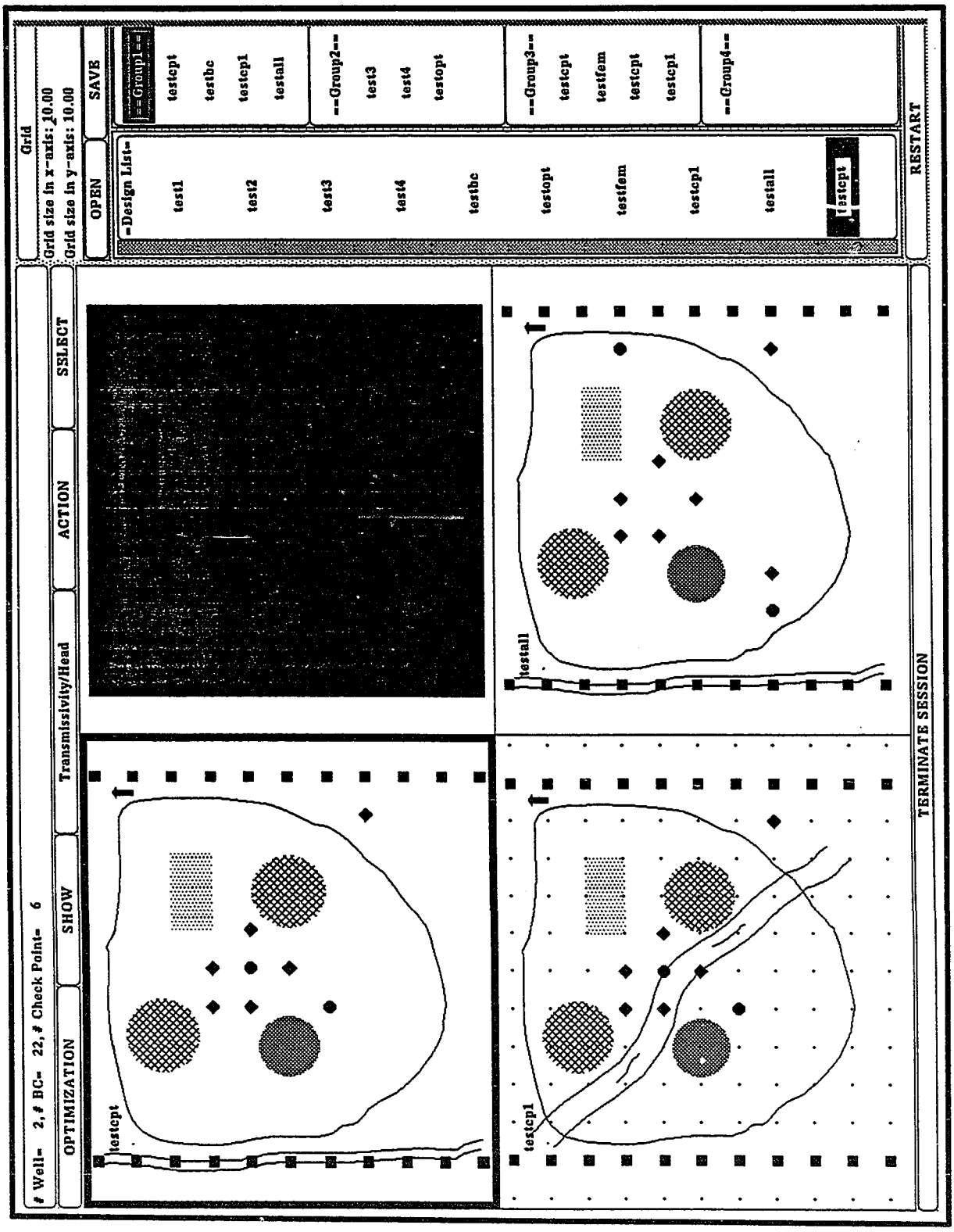


Figure 6.25

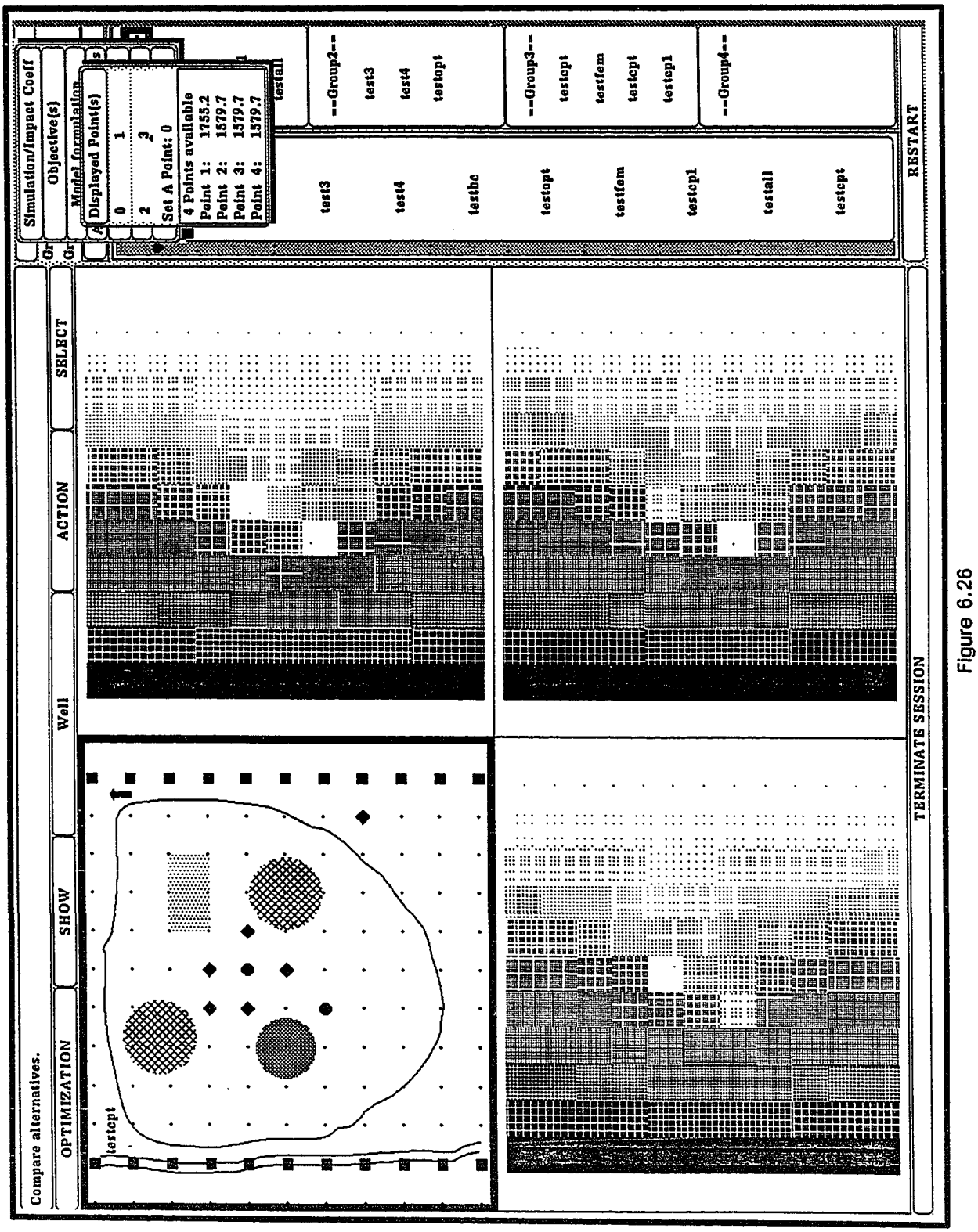


Figure 6.26

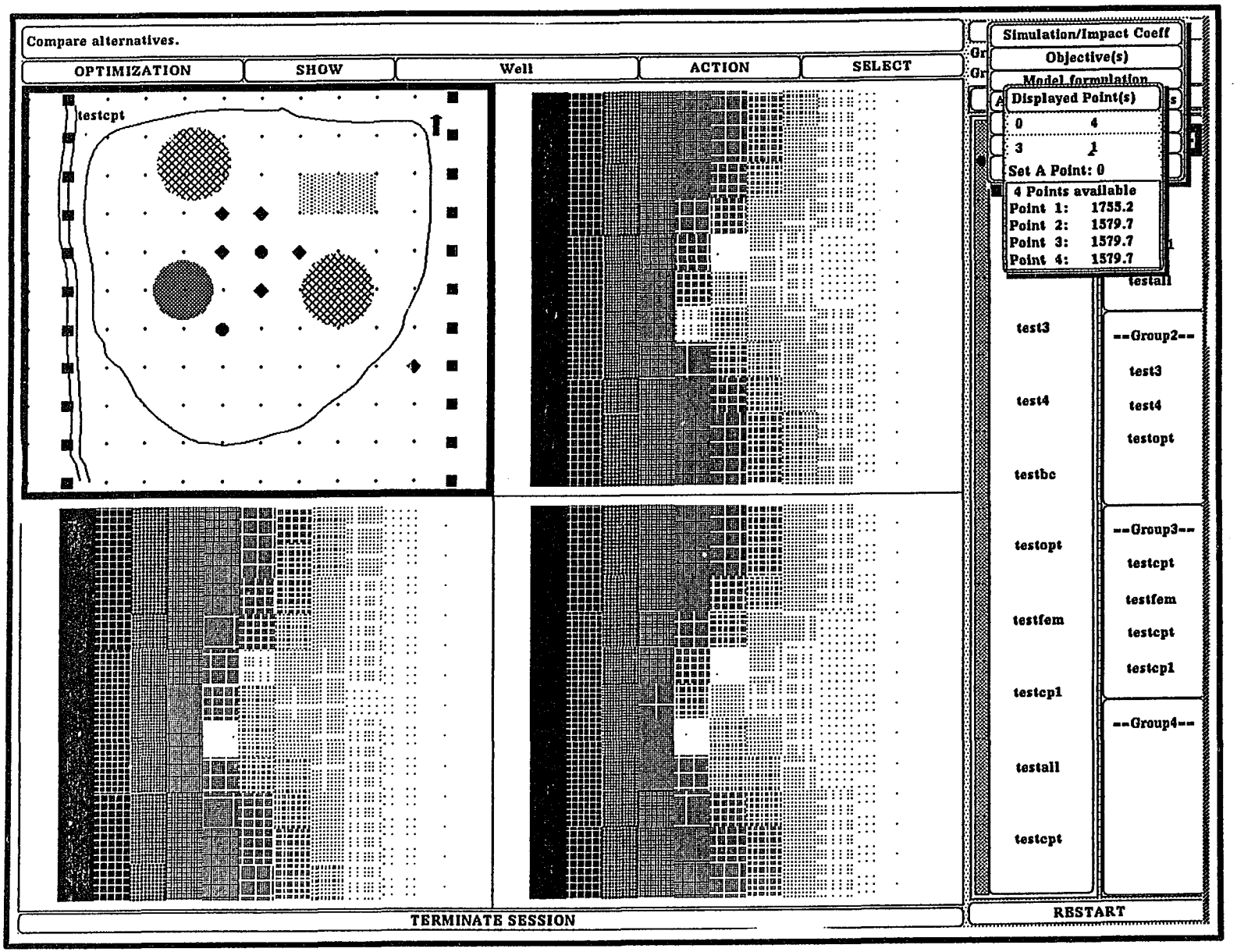


Figure 6.27

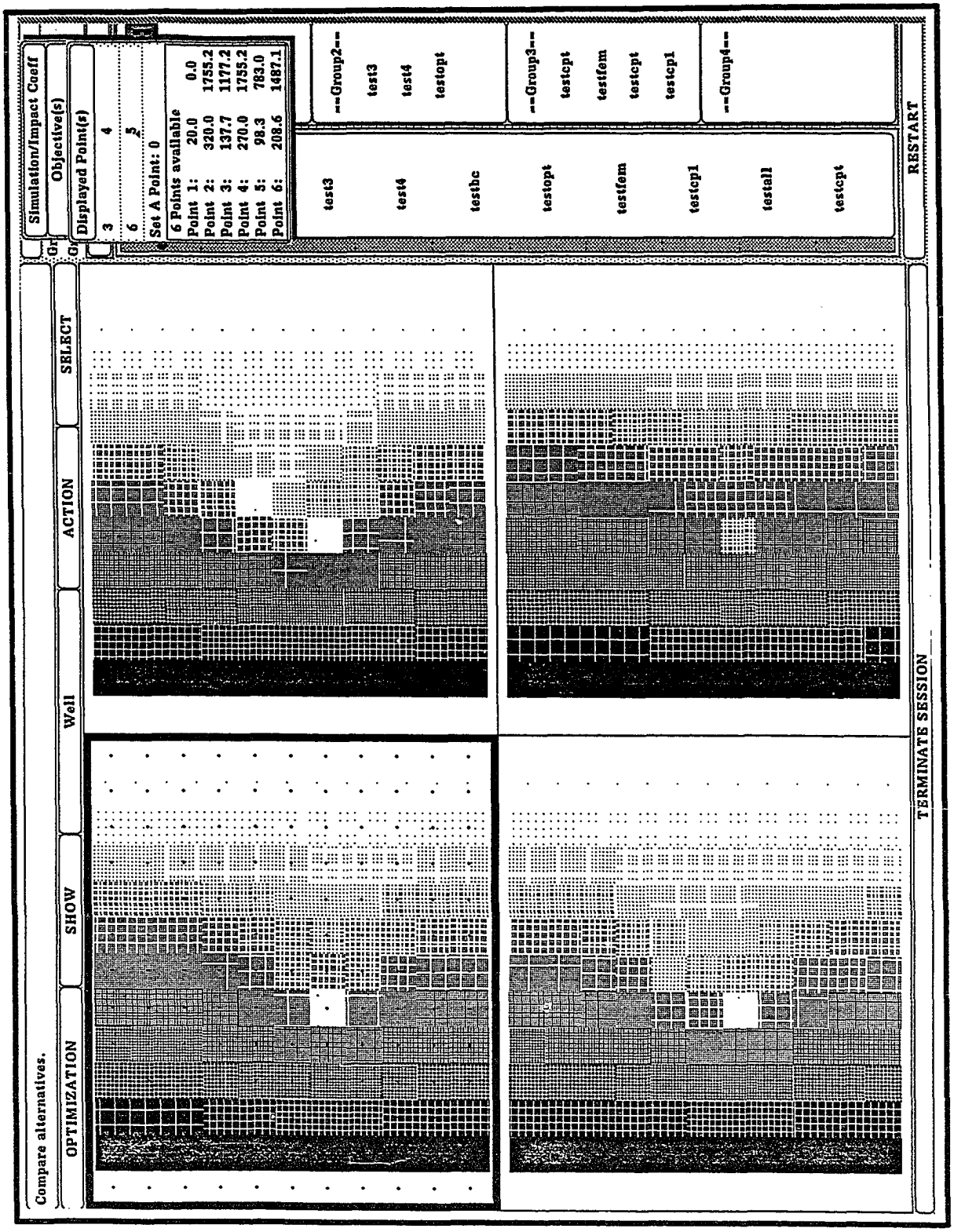


Figure 6.28

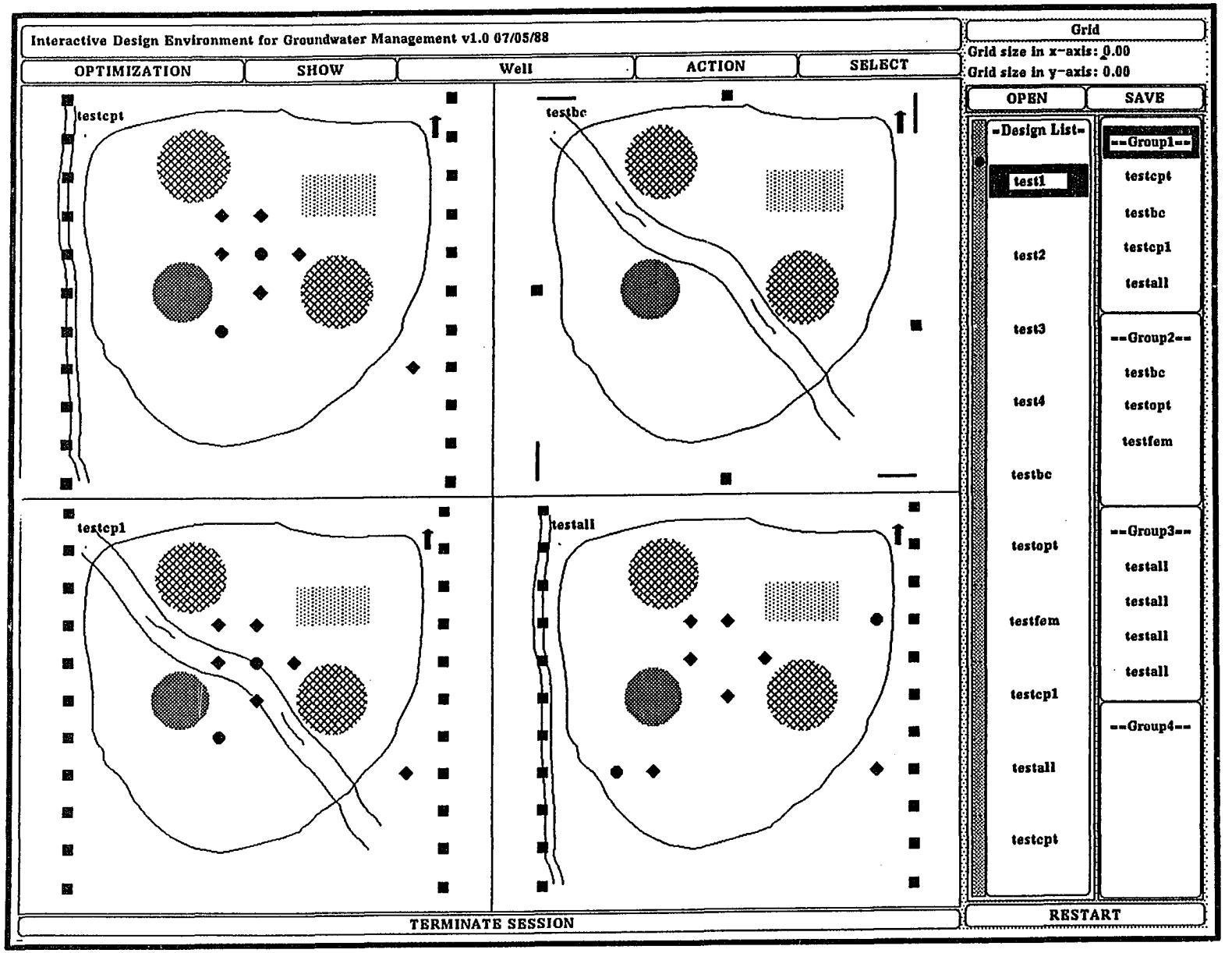


Figure 6.29

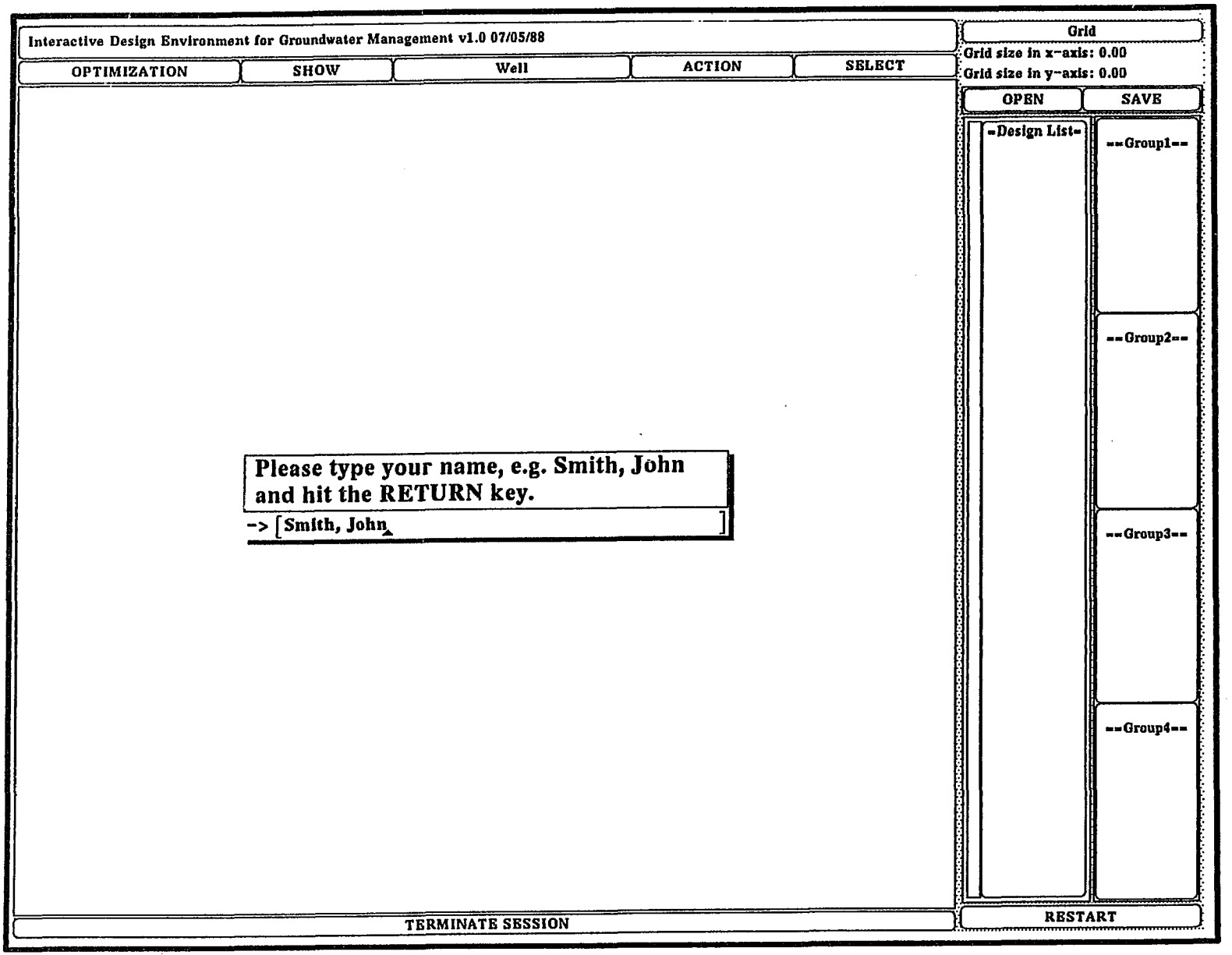


Figure 6.30

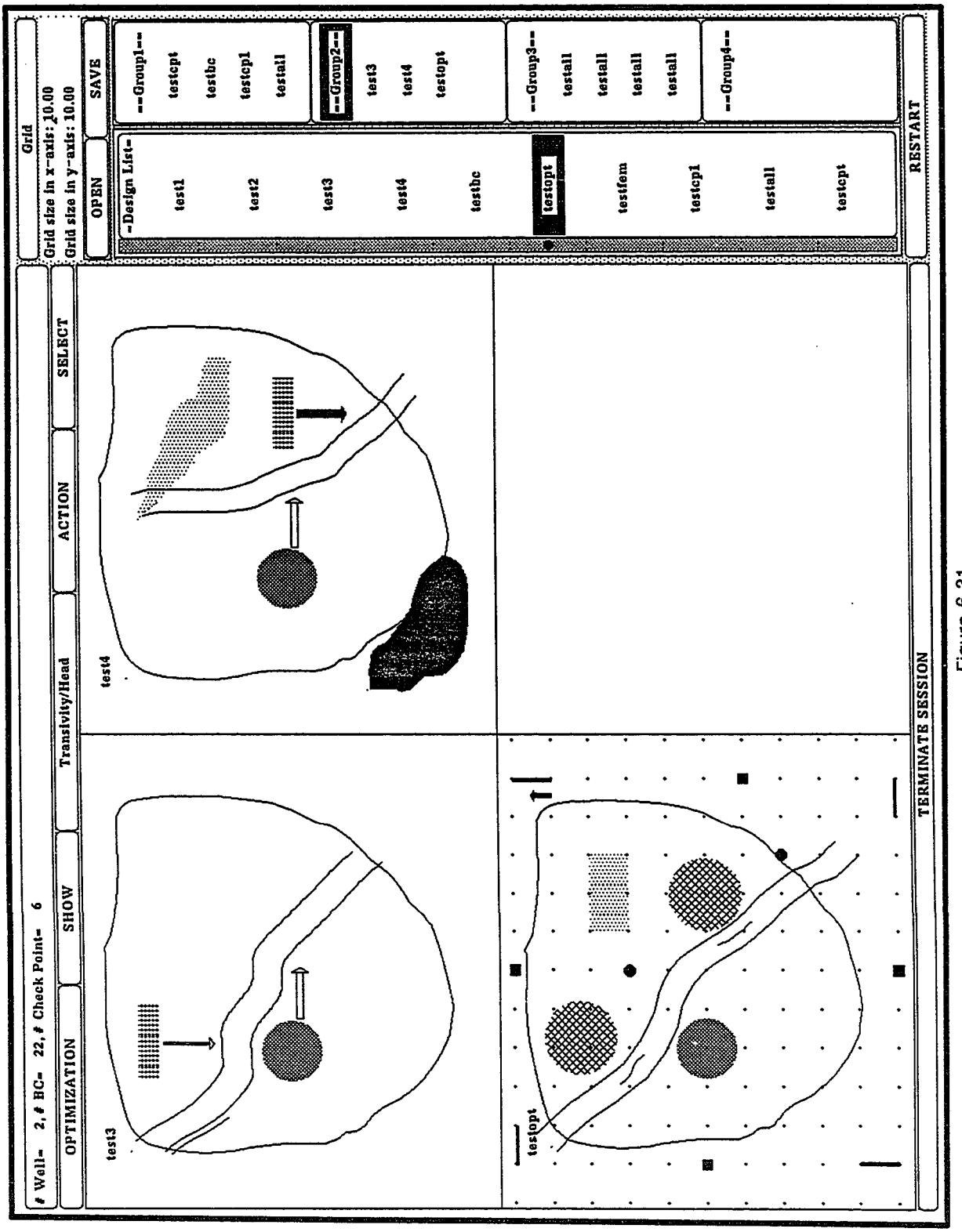


Figure 6.31

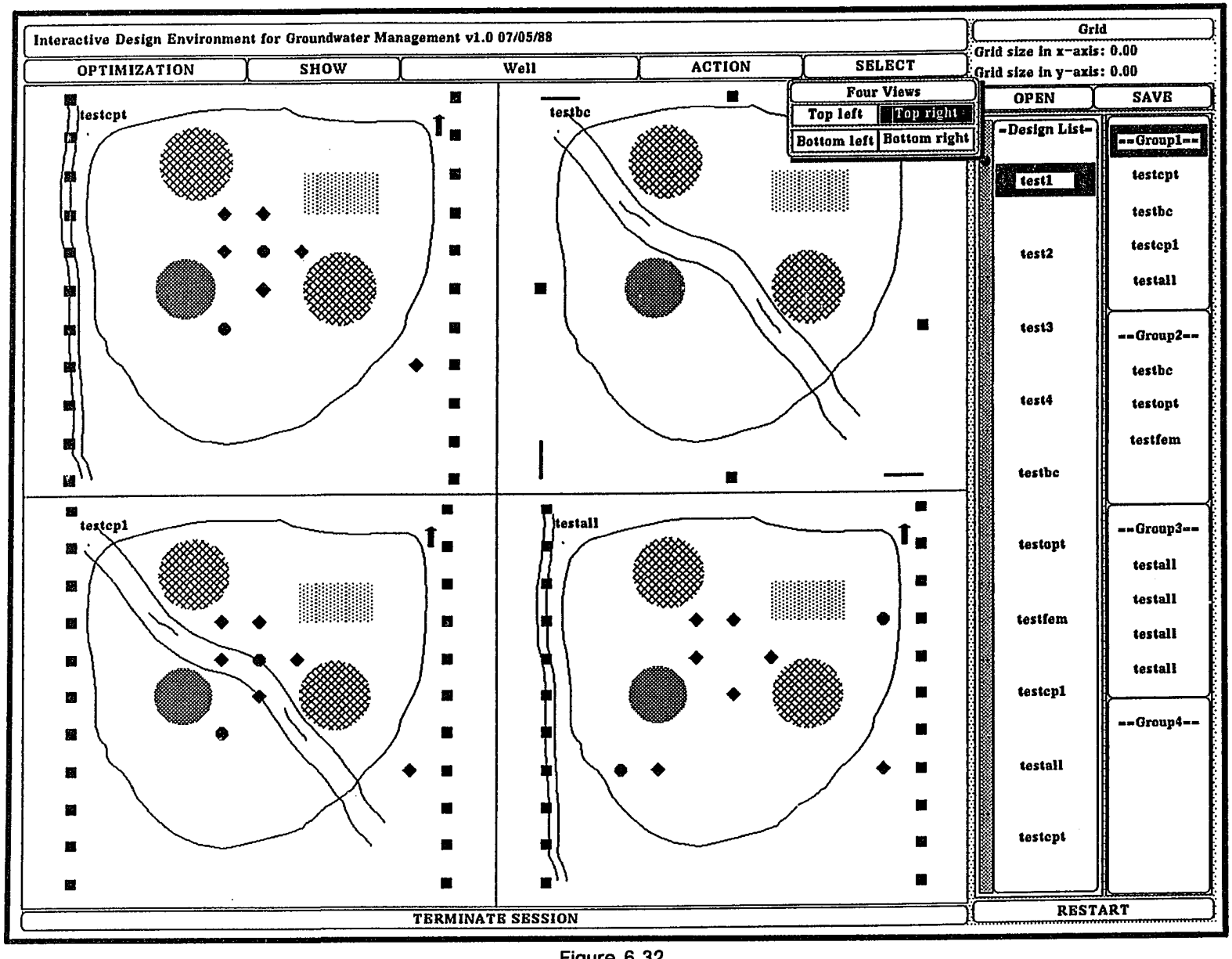


Figure 6.32

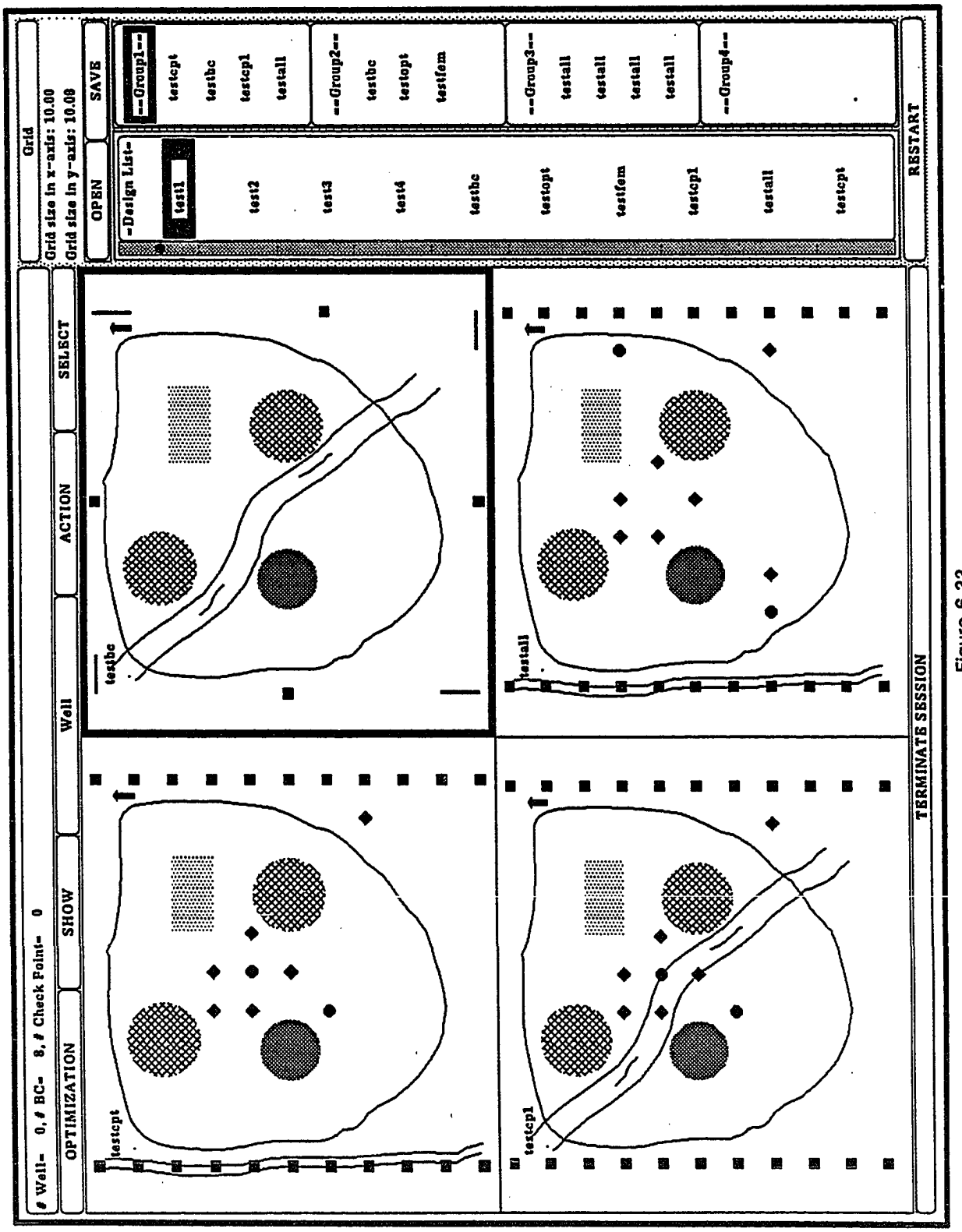


Figure 6.33

CHAPTER 7

DISCUSSION AND CONCLUSION

Two prototype computer-aided systems have been developed for a WTPD and a GRM model. The prototypes are intended to improve the effectiveness of implementing decision making analysis tasks (see Chapter 3). An analyst or decision maker does not have to be a computer expert to use the prototypes, and the user-friendly interfaces require a minimal time for learning. The interactive response time is quick and editing is easy. The prototypes also take care of many time-consuming tasks: bookkeeping, tabular data preparation, data management, etc. These tasks usually occupy a significant amount of an analyst's time. With the prototypes, these tasks can be done as simply as pushing a button. Thus, the analyst's time can be spent more efficiently in examining issues for generating good alternatives. The productivity of the analyst is then enhanced.

Several new effective techniques have also been developed. The Vector Method overcomes drawbacks in using the NISE method and generates a complete noninferior set efficiently for a multicriterion problem. The new modeling language that was developed can be used to construct a mathematical programming model in a human-understandable form. A new interpretation of the HSJ method and two variant HSJ methods is also introduced. The two variant methods were not examined in the prototypes, but they can be included in future research.

Although the two systems are for two specific problems, they illustrate concepts of developing computer aided systems for engineering design. The discussions in the next section focus on general issues in the design of a computer aided system for engineering decision making problems. The discussions are based on the general components of such a system discussed in Chapter 3 and use examples from the two prototypes.

7.1. Issues In Developing Computer Aided Systems

Mathematical Techniques and Tools

For an engineering decision making problem, it is usually required to use (or develop) several mathematical techniques or tools. In addition to general characteristics such as accuracy and efficiency, the linkages among the users, techniques, tools, and other components in the computer aided system should be effectively built for incorporation of mathematical techniques and tools into a computer aided system. Further, the sequence of implementation of techniques should be carefully determined.

Setting up an input format, e.g. Table 4.4, for a software package is generally time-consuming for somebody who is not familiar with the package, and it may be difficult to transfer an output from a package to become an input to another package. An interface is therefore suggested as a bridge among mathematical packages.

The interface can be presented either alphabetically and numerically as in the modeling language approach or graphically as in the graphic object oriented approach. A new modeling language was demonstrated for the GRM prototype, and the graphical objects used let a designer easily manipulate attributes. The graphical objects, of course, cannot fully replace a mathematical model because of completeness and accuracy, but they provide a good overview of a model. Furthermore, the perception of graphical objects as a model can be improved if a more precise presentation can be provided. For example, the size of any circle used in Chapter 6 to express a pumping well can be used to represent the amount of withdrawal, and the size can be understood if a scaling aid is provided. Even though such an improvement may be possible, there is always a tradeoff between capability and simplicity. Providing a scaling aid may complicate the working screen, and it may take more time to see the exact value from the size than from the mathematical model.

Making decisions about tradeoffs between capability and simplicity is a very important activity in this area of research. Different decisions may change the final product significantly. The main factors considered in this research for the decisions are: user's preference, available software and hardware facilities and their limitations, implementation time, frequency of usage, and contribution. In other words, a function should not be designed if it is hard for the user to understand or learn, is difficult to implement on current facilities, forms a bottleneck and slows down the system, or is only useful for a specific situation or purpose.

The sequence of application of the techniques may also greatly affect the performance of a computer aided system. In general, short response time is desired for frequently used functions. The response time, however, may slow down if the sequence of implementing the techniques is not carefully considered. For example, in the GRM case the simulation program is implemented once for each existing well, and the impact coefficients of each well on each grid point in the problem domain are stored. Since all impact coefficients are available, the response for adding, deleting, changing information about a well or checking point will be quick because the new solution (hydraulic heads) can be determined by using the impact coefficients. Although the problem can be solved only once for all wells at the same time to determine hydraulic heads on the problem domain, impact coefficients will not be available, and the response would be delayed because the incorporation of an analysis program is required to determine the

new solution. The general rule used in this research for implementing a technique or other non-mathematical functions is to reduce the response time for frequently used functions as much as possible and to shift the time burden to less frequently used functions.

Graphical Interface

As mentioned and demonstrated, the graphical interface is mainly used for presentation. Presentation is important in making comparisons and thus good decisions. However, presentation of an attribute (e.g. cost), model (e.g. wastewater treatment plant model), or solution (e.g. noninferior set) sometimes is difficult.

Clarity and simplicity are characteristics of a good presentation, but they are usually in conflict with each other. For example, in the WTPD case a lot of information can be shown for a design. In the PC version, a menu system with up to four levels of popup windows was used. Although each screen provides clear information for an individual piece of the design, the user may not easily recall the entire system. One way to overcome this complexity is not to use popup windows. The screen size of an Apollo workstation monitor is suitable to hold most of desirable information, but there are several complexities for this way of presentation: 1) too much information shown at the same time may be difficult to handle; 2) display of detailed information about a component, such as a unit process, would reduce room for other information, e.g. about the plant scheme; 3) the designer usually does not need most of the information simultaneously; 4) the design screen may become complicated and the response time will be increased. Thus, a decision was made as a compromise between the clarity and simplicity: a menu system with up to two levels of popup windows was used for the final WTPD prototype. The second level of popup windows shows infrequently examined or less important attributes of a design, and these attributes are expected to be examined only about once for a design. Even when a second level popup window is needed, it is displayed beside the first level popup window instead of erasing the first one (Figures 5.12, 5.14 and 5.16). The designer, therefore, still has a chance to examine all related information at the same time, while the simplicity of usage, display, and response is maintained.

Judgments on the tradeoff between clarity and simplicity in presentations occur in many situations, e.g. use of hidden screen(s), places to show popup windows, layout of presented information, etc. There are no explicit rules for making the judgments because they are problem dependent, but the general concept for making the judgments is to make modifications to increase both clarity and simplicity while maintaining an acceptable level of each. This itself is a multiobjective decision making problem with

unquantifiable issues. Using the MGA conceptual approach, having typical users examine several different alternative presentations may be the best way to make the decisions.

Although the graphical interface is used mainly for presentation, it should not be restricted to that purpose. For example, the plant scheme and graphic objects demonstrated in Chapters 5 and 6 can also be used as an interface to retrieve information related to a physical component expressed as a graphic object. A graphic interface may reduce the complexity of a user interface significantly.

User Interface

A computer aided system requires a good user interface. The characteristics of a good user interface are, as described in Chapter 3: ease of learning, minimization of mistakes, flexibility in modification, efficiency of data and solution organization, and clarity of instructive feedback.

A pointing device, mouse, is used to make most option selections in the two prototypes. By moving a mouse and clicking a button on a desired menu item or graphic object, a function can be easily selected with the response displayed immediately after the selection. In case the user makes a logic mistake, feedback (e.g. beeper or error message) will explain the mistake and give some suggestions. For example, in the WTPD case if an attempt is made to solve an infeasible design, then a popup window will be shown with a message for violations and possible modifications to the design. Mistakes such as the infeasibility example are hard to prevent in advance of a design session, but many general mistakes are preventable. For example, in developing the two prototypes, the layout of all menu items was carefully arranged to avoid inadvertently selecting wrong menu items. In the WTPD case, there are free and fixed groups of menu items. The two groups are put in different areas on the screen, and the menu texts of the inactive items are turned gray and cannot be selected. The chance of inadvertently selecting wrong items is thus significantly reduced.

Another important aspect of a user interface is data and/or solution management. Tedious tasks, such as bookkeeping, data storage, classification, etc., usually take a significant amount of an analyst's time. The computer aided systems should be responsible for implementation of these tasks so that an analyst can spend time more efficiently. Although the details of the design of data and solution management are not described, the general concept used was to keep most data during a working session in hardware memory which has fast accessibility but is of limited capacity. Data are saved permanently in an alternative memory when the user is not actively engaging the system, e.g. he is thinking or reading. This concept is to try to make the best use of capabilities of a computer. The storage and retrieval of data in the two prototypes are generally not necessarily explicit to a user except when the

user interface is used, e.g. the grouping function in the GRM case. The file in storage, however, should be opened to a designer if needed. For example, the mathematical models in the new modeling language for the GRM prototype were stored by the designer's name and can be used separately without the prototype. The files for models can be used for other purposes, e.g. presentation or input to a software package.

This section discusses general issues with examples from the demonstrated prototypes. The two prototypes are for two specific problems. The general issues such as those described above and in Chapters 1, 3, 5, and 6, are applicable to many other problems. Currently, few systems have been developed for environmental decision making problems. This research is intended to demonstrate the capability of a computer aided system to improve an environmental decision making analysis. To summarize this research, the use of a computer aided system in a decision making process is discussed further in the next section.

7.2. Decision Maker(s), Analyst(s), Computer Aided System(s) and Decision Making Process(es)

Without computer aided systems, the decision making processes presented in Chapter 3 (Figure 3.1 and 3.2) would be implemented step by step and iteratively until a good decision is reached. However, with computer aided systems on a multi-tasking workstation, the analysis and decision making process can become more dynamic. The analyst can jump from one task to another more easily. Both analyst and decision maker can use the same systems to examine the issues of concern, and thus it is possible to have greater interactions between them. Also a new task(s) which is significant for a particular decision making problem may be needed in a given case. For example, the decision maker may want to identify a utility function to generate a compromise solution. Computer aided systems allow the extension of tasks, e.g. a utility function can be easily added to a model by the modeling language. The decision making process shown in Figure 3.1 is therefore not enough to explain these dynamic manipulations. A new dynamic decision making process is thus proposed as in Figure 7.1. The process is less sequential. Instead, interaction, interruption, and detour may occur more readily at any stage of the process.

As shown in Figure 7.1, the analyst, decision maker, and friendly interface provided by computer aided systems form an efficient dynamic decision making process. By using computer aided systems like the prototypes, it is expected that decisions can be made in a more efficient and effective manner.

The prototypes can also serve as an interface between an analyst and a decision maker. Several useful functions were designed to make it easy to compare alternatives to gain insights. The prototypes can also be used to present the alternatives to the decision maker, and the decision maker can examine the alternatives directly on the graphical interface provided. Interactions between the analyst and

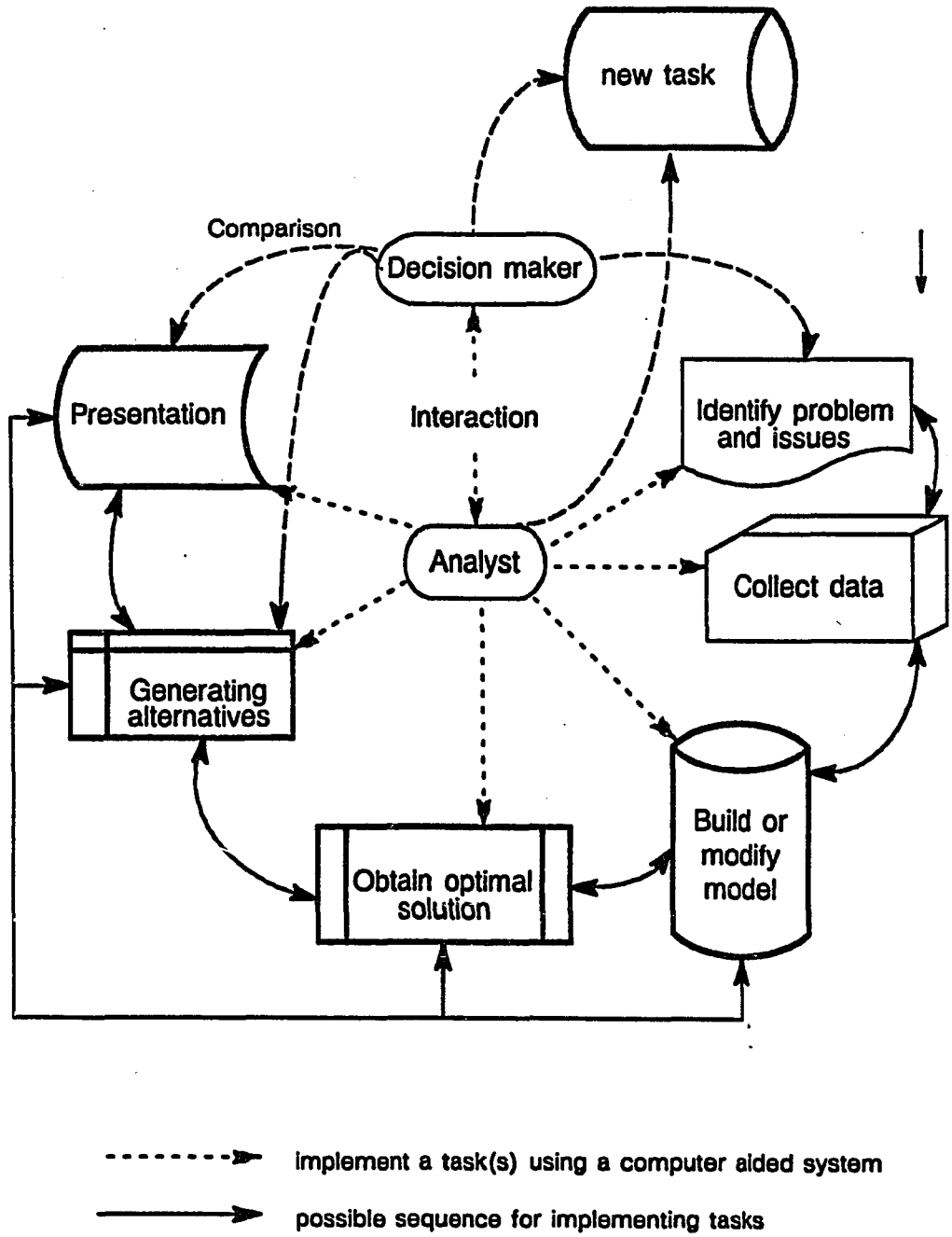


Figure 7.1 A Working Process For A Decision Making Problem

decision maker should be made easier. Better solutions could result and the time required for a decision making process could then be significantly reduced.

7.3. Future Research

Since the computer aided systems are research prototypes, a number of changes or extensions are needed to make them more complete, more robust, and more efficient. For example, other user-friendly features such as 'cut and paste' may be used. Of course, there is always a tradeoff between capability and simplicity. How to provide the maximum capability while maintaining simplicity is a key research issue in developing a computer aided system. Several suggestions for improvements or potential extensions of the prototypes or new techniques are listed below:

- develop new ways to present a noninferior set for a model with three or more objectives;
- explore ways to present attributes of the WTPD model for comparison;
- overcome the numerical difficulties that occur in solving a highly nonlinear WTPD model and improve the computation efficiency in optimizing the model;
- extend the new modeling language to handle nonlinear models;
- incorporate the new HSJ methods or develop other MGA methods to guarantee the generation of maximally different alternatives for most problems;
- demonstrate and prove the applicability of the Vector Method to an N-dimensional problem;
- extend the WTPD model to include any process system;
- add contaminant transport to the GRM model;
- extend the MGA capability of the GRM prototype to handle problems with two or more objectives;
- provide flexibility in: selecting a graphic object to express an attribute or real object, adding attributes to the prototypes, modifying the way to create an optimization model, and arranging the display and layout of menu items or models; and
- provide easy-to-learn tutorials (as the one developed for IDEAS [Brill, et al., 1989]).

Certainly, there are many other useful options to extend the prototypes. To provide an additional capability might, however, increase the complexity of using the prototypes. Before an extension is made, it should be evaluated carefully to ensure that the benefit justifies the complexity. An improved computer aided system should also be easy to learn and use.

APPENDIX A

GROUNDWATER SIMULATION MODEL AND IMPACT COEFFICIENTS

This appendix presents the simulation model and impact coefficients used for the computer aided system for the groundwater management computer aided system described in chapter 5.

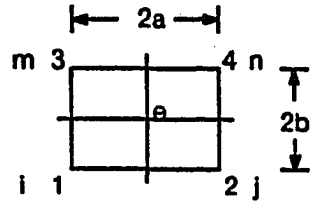
A.1. Simulation Model

The finite element method has been widely applied to solve groundwater problems in recent years. This method leads to a set of algebraic equations in which the unknowns are at a finite number of nodal points (or grid point in this context). The simulation model used is a 2-dimensional rectangular finite element model. The rectangular element is defined by four nodes, one at each corner. These nodes serve the purpose of locating unknown heads. The head within each element is defined in terms of the nodal values using basis or interpolation functions. The head throughout the domain can be defined by the weighted residual principle (Galerkin's method, see Wang et al. [1982]). The details of the algorithm are shown in Algorithm 2D-FEM. Gaussian Quadrature was applied in the formulation. A computer program was developed using FORTRAN 77 to solve a groundwater model numerically. The Crout method was used to decompose a matrix into two triangular matrices for solving the simultaneous equations of the model. The boundary condition can be either a fixed-head or fixed-flow type.

Algorithm 2D-FEM

$$\frac{\partial}{\partial x}T_x\frac{\partial h}{\partial x}+\frac{\partial}{\partial y}T_y\frac{\partial h}{\partial y}=Q(x,y)$$

$$h(x,y)=\hat{h}(x,y)=\sum_{j=1}^{4}\hat{h}_j\,\phi_{e,j}(x,y)$$



global: i, j, m, n
local: 1, 2, 3, 4

$$\sum_{j=1}^{4}\hat{h}_j\int_{\Omega^e}\left(\partial\frac{\phi_i^e}{\partial x}T_x\frac{\partial\phi_j^e}{\partial x}+\frac{\partial\phi_i^e}{\partial y}T_y\frac{\partial\phi_j^e}{\partial y}\right)d\Omega-\int_{\Gamma^e}\phi_i^eT_n\frac{\partial\hat{h}}{\partial n}\,d\Gamma+\int_{\Omega^e}Q(x,y)\phi_i^e\,d\Omega=0$$

$$\begin{bmatrix}\phi_1^e=\frac{1}{2}\left(1-\frac{x}{a}\right)\frac{1}{2}\left(1-\frac{y}{b}\right)\\ \phi_2^e=\frac{1}{2}\left(1+\frac{x}{a}\right)\frac{1}{2}\left(1-\frac{y}{b}\right)\\ \phi_3^e=\frac{1}{2}\left(1+\frac{x}{a}\right)\frac{1}{2}\left(1+\frac{y}{b}\right)\\ \phi_4^e=\frac{1}{2}\left(1-\frac{x}{a}\right)\frac{1}{2}\left(1+\frac{y}{b}\right)\end{bmatrix}\quad\begin{array}{l}set\ \ \xi=\frac{x}{a}\\ \qquad\eta=\frac{y}{b}\\ \Rightarrow dx=ad\xi\\ \qquad dy=bd\eta\end{array}\quad\Rightarrow\quad\begin{bmatrix}\phi_i^e=\frac{1}{4}(1-\xi)(1-\eta)\\ \phi_j^e=\frac{1}{4}(1+\xi)(1-\eta)\\ \phi_m^e=\frac{1}{4}(1+\xi)(1+\eta)\\ \phi_n^e=\frac{1}{4}(1-\xi)(1+\eta)\end{bmatrix}$$

From Gaussian Quadrature

$$\int_{-1}^{1}g(\xi)\,d\xi=g(\xi_1)+g(\xi_2)$$

$$\int_{-1}^{1}\int_{-1}^{1}g(\xi,\eta)\,d\xi\,d\eta=g(\xi_1,\eta_1)+g(\xi_1,\eta_2)+g(\xi_2,\eta_1)+g(\xi_2,\eta_2)$$

$$where\ \xi_1=\eta_1=\frac{-1}{\sqrt{3}}\ \text{and}\ \xi_2=\eta_2=\frac{1}{\sqrt{3}}$$

$$\int_{-a}^{a}\int_{-b}^{b}\phi(x,y)\,dx\,dy=ab\int_{-1}^{1}\int_{-1}^{1}\phi(\xi,\eta)\,d\xi\,d\eta$$

For node L

$$\left[\begin{array}{l} A^e_{L,i} = \int_{-a}^{a}\int_{-b}^{b}\left(\frac{\partial\phi^e_i}{\partial x}T_x\frac{\partial\phi^e_L}{\partial x}+\frac{\partial\phi^e_i}{\partial y}T_y\frac{\partial^e_L}{\partial y}\right)dx\,dy \\ A^e_{L,j} = \int_{-a}^{a}\int_{-b}^{b}\left(\frac{\partial\phi^e_j}{\partial x}T_x\frac{\partial\phi^e_L}{\partial x}+\frac{\partial\phi^e_j}{\partial y}T_y\frac{\partial^e_L}{\partial y}\right)dx\,dy \\ A^e_{L,m} = \int_{-a}^{a}\int_{-b}^{b}\left(\frac{\partial\phi^e_m}{\partial x}T_x\frac{\partial\phi^e_L}{\partial x}+\frac{\partial\phi^e_m}{\partial y}T_y\frac{\partial^e_L}{\partial y}\right)dx\,dy \\ A^e_{L,n} = \int_{-a}^{a}\int_{-b}^{b}\left(\frac{\partial\phi^e_n}{\partial x}T_x\frac{\partial\phi^e_L}{\partial x}+\frac{\partial\phi^e_n}{\partial y}T_y\frac{\partial^e_L}{\partial y}\right)dx\,dy \end{array}\right] \Rightarrow$$

$$\left[\begin{array}{l} A^e_{L,i} = ab\int_{-1}^{1}\int_{-1}^{1}\left(\frac{1}{a^2}\frac{\partial\phi^e_i}{\partial\xi}T_x\frac{\partial\phi^e_L}{\partial\xi}+\frac{1}{b^2}\frac{\partial\phi^e_i}{\partial\eta}T_y\frac{\partial^e_L}{\partial\eta}\right)d\xi\,d\eta \\ A^e_{L,j} = ab\int_{-1}^{1}\int_{-1}^{1}\left(\frac{1}{a^2}\frac{\partial\phi^e_j}{\partial\xi}T_x\frac{\partial\phi^e_L}{\partial\xi}+\frac{1}{b^2}\frac{\partial\phi^e_j}{\partial\eta}T_y\frac{\partial^e_L}{\partial\eta}\right)d\xi\,d\eta \\ A^e_{L,m} = ab\int_{-1}^{1}\int_{-1}^{1}\left(\frac{1}{a^2}\frac{\partial\phi^e_m}{\partial\xi}T_x\frac{\partial\phi^e_L}{\partial\xi}+\frac{1}{b^2}\frac{\partial\phi^e_m}{\partial\eta}T_y\frac{\partial^e_L}{\partial\eta}\right)d\xi\,d\eta \\ A^e_{L,n} = ab\int_{-1}^{1}\int_{-1}^{1}\left(\frac{1}{a^2}\frac{\partial\phi^e_n}{\partial\xi}T_x\frac{\partial\phi^e_L}{\partial\xi}+\frac{1}{b^2}\frac{\partial\phi^e_n}{\partial\eta}T_y\frac{\partial^e_L}{\partial\eta}\right)d\xi\,d\eta \end{array}\right] \Rightarrow$$

where

$$\frac{\partial\phi^e_i}{\partial\xi}=\frac{-1}{4}(1-\eta) \qquad \frac{\partial\phi^e_i}{\partial\eta}=\frac{-1}{4}(1-\xi)$$

$$\frac{\partial\phi^e_j}{\partial\xi}=\frac{1}{4}(1-\eta) \qquad \frac{\partial\phi^e_j}{\partial\eta}=\frac{-1}{4}(1+\xi)$$

$$\frac{\partial\phi^e_m}{\partial\xi}=\frac{1}{4}(1+\eta) \qquad \frac{\partial\phi^e_m}{\partial\eta}=\frac{1}{4}(1+\xi)$$

$$\frac{\partial\phi^e_n}{\partial\xi}=\frac{-1}{4}(1+\eta) \qquad \frac{\partial\phi^e_n}{\partial\eta}=\frac{1}{4}(1-\xi)$$

$$A_{L,i} = A_{g_k}(\xi_1, \eta_1) + A_{g_k}(\xi_1, \eta_2) + A_{g_k}(\xi_2, \eta_1) + A_{g_k}(\xi_2, \eta_2)$$

where $\xi_1 = \eta_1 = \frac{-1}{\sqrt{3}}$ and $\xi_2 = \eta_2 = \frac{1}{\sqrt{3}}$

$$A_{g_k}(\xi, \eta) = ab\left(T_x \frac{1}{a^2} \frac{\partial \phi_i^e}{\partial \xi} \frac{\partial \phi_L^e}{\partial \xi} + T_y \frac{1}{b^2} \frac{\partial \phi_i^e}{\partial \eta} \frac{\partial \phi_L^e}{\partial \eta} \right)$$

Then, a matrix system can be formulated as

$$[A]\,\underline{h} = \underline{f_b} + \underline{f_q}$$

where

$\underline{h}$ is vector of head values;

$\underline{f_b}$ is vector of boundary conditions; and

$\underline{f_q}$ is vector of recharges or withdrawals.

A.2. Impact Coefficients

To compute the impact coefficients for drawdown at each check point resulting from a withdrawal at each pumping (or recharge) well, the first step is to solve the simulation program for the groundwater model without any pumping or injection. Then, the model is solved once again for each well with the introduction of an unit withdrawal (or recharge) at that well location. The impact coefficients at all grid points for each well is then determined by computing the difference between the second solution and the first solution.

The drawdown at each point can be determined for other levels of withdrawal or recharge by multiplying the value of discharge or injection by the associated impact coefficient. The hydraulic head at each grid point can be determined by subtracting the summation of the drawdowns caused by all wells from the head determined in the first solution. Each time a new well is added, the impact coefficients related to the well can be determined using the simulation program. Since the computer aided system stores the solutions rather than the impact coefficients, to add or delete a check point does not require implementing the entire simulation program again. This approach reduces the interactive response time for adding or deleting check points. The impact coefficients are actually determined and stored while an optimization model is created.

APPENDIX B

PROGRAM STRUCTURE

The overall design and characteristics of the programs of the two prototype computer aided systems are described in this appendix. The designs of programs for mathematical techniques and the newmodeling language are not described. The codes are not listed, but an overview and the structure of the programs are provided. The length of the programs is about 9000 lines in total. Several languages and software packages were used to develop the programs on an Apollo workstation, a Unix based computer equipped with a 1024x1280 monochrome screen monitor, a three-button mouse, and a QWERTY keyboard.

The two major software packages that were used in the development of the programs are briefly discussed, with several examples, in the following section. The programming structure of the programs is then described for several major tasks.

B.1. Software Packages

Two major packages, DOMAIN/DIALOGUE [1987] and DOMAIN/2D Metafile [1985], were used to develop the programs. Each package is briefly discussed below with some examples from the programs.

DIALOGUE (Purpose: Interface Design for Menu Items)

DIALOGUE, an Apollo DOMAIN interface language, was mainly used to design the user-interface of the programs. A detailed description of DIALOGUE can be found in the DOMAIN/DIALOGUE User's Guide. Only a brief discussion based on the programs is presented here.

The interface between the users and application programs (developed by the software designer for special tasks) was established by means of a DIALOGUE descriptive file. The descriptive file contains two major sections: Application Interface (see the left column in Table B.1) and User Interfaces (see the right column in Table B.1).

Table B.1: Sample DIALOGUE descriptive file:
Application and User Interfaces

```
APPLICATION_INTERFACE wwtpface
{---------MainMenus----------}
ViewEditModel:= ENUM:
      COMP => <call pSwitchWork>;
      CHOICES = (ViewEdit ModelOpt);
      VALUE = ViewEdit;
END

MethodList:= ENUM:
      COMP => <call pMethod>;
      CHOICES = (FixSize FixLoading);
      VALUE = FixLoading;
END

Solve := NULL:
      COMP => <call Solving>;
END

Bulking := NULL:
      COMP => <call pBulk>;
END

CostShow := NULL:
      COMP => <call pCostUnits>;
END

Quit:= Null:
      COMP => <RETURN>;
END
```

```
USER_INTERFACE wwtpface
{-------Top  Row --------}
MainMenuS := ROW:
      ORIENTATION = horizontal;
      CONTENTS = (SwitchWorkI MethodSwitch
          SolveI BulkingI CostI QuitI);
END

SwitchWorkI := MENU:
      ORIENTATION = horizontal;
      MARKSTYLE = checkbox;
      Color_set = off;
      FONT = "/sys/dm/fonts/times-bold14";
      Task = ViewEditModel;
      ENTRIES = (  "View/Edit"  => ViewEdit
                   "Flow Model" => ModelOpt );
      HELP_TEXT = "Switch the working status";
 End

MethodSwitch := MENU:
      ORIENTATION = horizontal;
      MARKSTYLE = checkbox;
      FONT = "/sys/dm/fonts/times-bold14";
      Task = MethodList;
      Color_set = off;
      ENTRIES = ("Fixed Process sizes"
                                => FixSize
                 "Specified Loadings"
                                => FixLoading);
      HELP_TEXT= "Selection solution method.";
End

SolveI := ICON:
      TASK = Solve;
      STRING = "Solve";
      FONT = "/sys/dm/fonts/times-bold14";
END

BulkingI := ICON:
      TASK = Bulking;
      STRING = "Bulking";
      FONT = "/sys/dm/fonts/times-bold14";
END

CostI:= ICON:
      TASK = CostShow;
      select =>
         <CostTypesPop show; + take_locator>;
      FONT = "/sys/dm/fonts/times-bold14";
      String = "Cost";
END

QuitI := ICON:
      FONT = "/sys/dm/fonts/times-bold14";
      TASK =Quit;
      STRING = "Quit";
END
```

The Application Interface consists of programmer defined selections which are linked to application programs. Each selection could be associated with a predefined DIALOGUE task (e.g., task MSG will display a message, and task STRING will wait for input of string data--see DOMAIN/DIALOGUE User's Guide for a complete list). For example (see the left column in Table B.1), the selection "ViewEditModel" is linked to the application program "pSwitchWork" and associated with the DIALOGUE task MENU. This means when one of the options, VeiwEdit or ModelOpt, in "ViewEditModel" is selected by the user, DIALOGUE will pass control to the application program "pSwithWork" to switch working status.

The User Interface is used to construct the menu items to be displayed on the screen of the interface. Several characteristics of the menu items can be defined separately: characteristics such as the orientation, the shape and color of the menu item, the character string that appears on the menu item, the help message associated with the menu item, and the font and size of characters. For example (see the right column Table B.1), the selection "SwitchWorki" will display a checkbox type menu with two selections of "View/Edit" and "Flow Model" and the help message will be displayed on the screen when requested by the user. The overall layout of the menu items on the screen is then defined. For example (see the right colum in Table B.1), the menu items defined by "SwitchWorki", MethodSwitch", "Solvei", "Bulkingi", "Costi", and "Quiti" are grouped next to each other by "MainMenuS" as the top row options described in Chapter 4.

The use of DIALOGUE greatly expedites the user interface design of the programs to provide a user-friendly working environment.

2D Metafile (Purpose: Graphic Display)

The other major package used for developing the programs is Domain/Graphics 2D Metafile, which was mainly used to display the process scheme, performance model curves, cost curves, and other graphical output. This graphics package provides not only some primitive options, such as line, circle and box, but also some useful operations on graphics statements. The programs exploited two useful operations (segment and pick-and-identify operations) to display figures and also provide interactive ability.

The 2D Metafile provides the facility to define a group of graphic objects as a segment, and to perform actions on each segment. For example, a process scheme consists of boxes and polygon (unit processes), shaded or unshaded xxx(inflow, underflow, and outflow indications), lines (flows or links), and text (process names), one or several of which can be grouped as an individual graphics segment. This facility allowed the programs to be designed to be interactive.

The 2D Metafile also provides the facility to pick and identify graphic objects on the screen. This allows the programs to locate the position of the mouse cursor on the screen and the graphic object to which the mouse points. This facility immensely contributed to the interactive ability of the programs.

B.2. Program Structure

The programs were mainly written in PASCAL. Figure B.1 shows the logical sequence for one operation of the programs in the working session. The User Interface portion was designed using DIALOGUC.

Since the prototypes are interactive, their program structure does not follow a fixed flow or a hierarchical pattern. The programs can be interrupted or redirected based on user responses, so a traditional program flow chart or a hierarchical tree is not sufficient to explain the whole program structure. Instead, the programs are described by groups of program modules used to implement specific tasks. However, these groups are related to each other, and should not be considered as independent.

Program modules are grouped in the following categories: *Initialization, graphic display, action response, checking and warning, interactive ability, numerical model, message, and data management.* The concept and/or program logic for each group are described below.

Initialization

Modules in this group are used to initialize data, parameters, graphical screen area, interface layout, and message entries. Initialization is done at the initial use of a program or a function. For example, the initialization of the graphical area will not be done until a graphical display is needed.

Graphic Display

The modules in this group do not function independently. They are usually called by other modules to display or modify graphic objects on the screen.

Message

This group is used to provide messages in response to user actions. These messages report the current status of an operation and provide further instructions pertaining to the operation.

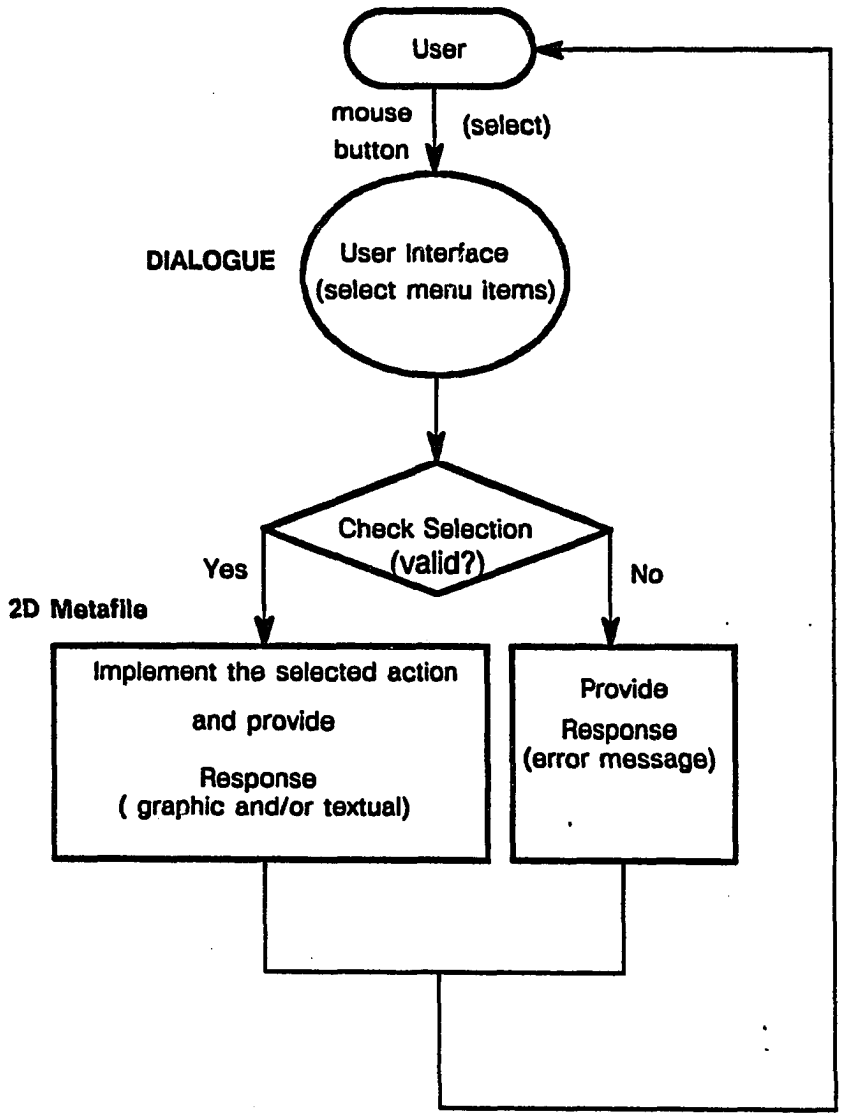


Figure B.1 Logical Sequence For One Operation

Modules related to the display of messages cannot be separated into individual modules as they appear in many places in the programs.

Action Response

This group forms the main body of the programs. The responses to a selected operation can be textual (e.g. messages described earlier), graphic (e.g. curves), or tabular (e.g. list of cost summary).

Checks and Warnings

Many error checks are performed after each action selected by the user. If an error check fails, generally a beep will be sounded with the display of an error message. The error checks include: wrong keystroke, infeasible design, wrong cursor position, change of working design without saving current modified working network, recycle flows, etc. Although most checks do not form an individual module, they do operate independently to help users avoid making mistakes.

Interactive Capability

The interactive ability of the programs was accomplished using two major features: 1) menu items of the user interface, built using DIALOGUE, to detect operation selections, and 2) the two modules written using 2D Metafile, to take control from DIALOGUE to continue the execution of the selected operation. As mentioned in Section B.1., the facility to pick and identify graphic objects makes possible the interactive capability of the programs. Although checks and displays are not included in this group, they are part of the interactive operations.

Numerical Model

Program modules for information and for implementing the mathematical techniques. They are indepent and can be isolated from the prototypes if desired.

Data Management and Operation

The various categories of data (e.g., list of design parameters, plant scheme) for each user were stored in different files or data structures, which were uniquely named based on the user's name.

Operations in the programs may jump from group to group or module to module without following a fixed sequence. The descriptions listed above provide an overview of the design of the programs.

The programs have been demonstrated to several professional environmental engineers. Its user-friendly fashion lets a designer or an analyst easily implement decision making analysis tasks without reading any user manual in advance. The learning time for using the computer aided system is short. Although the problem is complex, the interface guides the user in developing good alternatives. Also, the high resolution Apollo Domain 1280x1024 monitor has provided a good working environment for both the programmer and users.

LIST OF REFERENCES

1.Apollo Computer Inc., "Programming with Domain 2D Graphics Metafile Resource", No. 005097, Rev. 00, Chelmsford, MA, 1985.

2.Apollo Computer Inc., "The DOMAIN/Dialogue User's Guide", No. 004299, Rev. 01, Chelmsford, MA. 1987.

3.Balachandran M., and J. S. Gero, The Noninferior Set Estimation (NISE) Method for Three Objective Problems, Vol. 9, pp. 77-88, 1985.

4.Brill, E. D., Jr., L. D. Hopkins, J. Flach, R. Hariharan, J.-J. Kao, R. D. Morgan, and J. Uber, "Experimental Evaluation of Modeling-To-Generate-Alternatives Approach", Report to National Science Foundation, 1989.

5.Brill, E. D. Jr., The Use of Optimization Models in Public Sector Planning, Management Science 25, 5, pp. 413-422, 1979.

6.Chang, S. Y., E. D. Brill, Jr., and L. D. Hopkins, Use of Mathematical Models to Generate Alternative Solutions to Water Resources Planning Problems, Water Resources Research 18, 1, pp. 58-64, 1982.

7.Chang, S. Y., E. D. Brill, Jr., and L. D. Hopkins, Modeling to Generate Alternatives: A Fuzzy Approach, Fuzzy Sets and System, 9, pp. 137-151, 1983.

8.Chang, S. Y., and S. L. Liaw, Generating Designs for Wastewater Systems, Journal of the Environmental Engineering Division, ASCE, Vol. 111, No. 5, pp. 665-679, Oct. 1985.

9.Cohen, R. M., J. H. May, and H. E. Pople, Jr., An Intelligent Workstation for Electrocenter Design, IEEE Transactions on Systems, Vol. SMC-17, No. 2, Mar./Apr. 1987.

10.Cohon, J. L., and D. H. Marks, A Review and Evaluation of Multiobjective Programming Techniques, Water Resources Research, Vol. 11, No. 2, 1975.

11.Cohon, J. L., Multiobjective Programming and Planning, Academic Press, New York, 1978.

12.Control Data Corporation, "APEX-III Reference Manual. (Version 1.2)," No. 76070000, Rev. G, Minneapolis, Minn., 1979.

13.Das, P., and Y. Y. Haimes, Multiobjective Optimization in Water Quality and Land Management, Water Resources Research, Vol. 15, NO. 6, pp. 1313-1321, 1979.

14.Dick, R. I., and M. T., Suidan, Modeling and Simulation of Clarification and Thickening Processes, Mathematical Modeling for Water Pollution Control Processes, edited by T. M. Keinath, and M. P. Wanielista, Ann Arbor Science Publishers Inc., 1975.

15.Dyksen, W. R. and C. J., Ribbens, Interactive ELLPACK: An Interactive Problem-Solving Environment for Elliptic Partial Differential Equations, ACM Transactions on Mathematical Software, Vol. 13, No. 2, pp. 113-132, June 1987.

16.Ellison, E. F. D., and G. Mitra, UIMP: User Interface for Mathematical Programming, ACM Transactions on Mathematical Software, Vol. 8, No. 3, pp. 229-255, Sept. 1982.

17.Fourer, R., Modeling Languages Versus Matrix Generators for Linear Programming, ACM Transactions on Mathematical SOftware, Vol. 9, No. 2, pp. 143-183, June 1983.

18.Geselbracht, J. J., E. D. Brill, and J. T. Pfeffer, Rule-Based Model of Design Judgment about Sludge Bulking, Journal of Environmental Engineering, ASCE, Vol. 114, No. 1, pp. 54-73, Feb. 1988.

19.Gershon, M., and L. Duckstein, Multiobjective Approaches to River Basin Planning, Journal of Water Resources Planning and Management, ASCE, Vol. 109, No. 1, pp. 13-28, Jan. 1983.

20.Greenberg, H. J., A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models, ACM Transactions on Mathematical Software, Vol. 9, No. 1, pp. 18-56, March 1983.

21.Hwang, C. L., S. R. Paidy, and K. Yoon, Mathematical Programming with Multiple Objectives: A Tutorial, Compt. & Ops. Res., Vol. 7, pp. 5-31, 1980.

22.Kao, J.-J., Generation of Alternative Optima For Nonlinear Programming Problems and An Illustration For Wastewater Treatment Plant Design, M.S. Thesis, University of Illinois at Urbana-Champaign, 1987.

23.Kao, J.-J., J. T. Pfeffer, E. D. Brill, Jr., and J. Geselbracht, Workstation Environment for Wastewater Treatment Design Using AI and Mathematical Models, final report of WRC, 1989.

24.Kshirsagar, S. R., and E. D., Brill, Jr., Ideation and Evaluation Methods Applied to Land-Use Planning, Environment and Planning B: Planning and Design, Vol. 11, pp. 313-324, 1984.

25.Lang, S., Linear Algebra, Adlison-Wesley Publishing Co., Inc., pp. 365-372, 1971.

26.Johnson, L. E., and D. P. Loucks, Interactive Multiobjective Planning Using Computer Graphics, Compt. & Ops. Res., Vol. 7, pp. 89-98,1980.

27.Louie, P. W. F., W. W.-G. Yeh, and N.-S. Hsu, Multiobjective Water Resources Management Planning, Journal of Water Resources Planning and Management, ASCE, Vol. 110, No.1, pp. 39-55,1984.

28.Lucas, C. and G. Mitra, Computer-Assisted Matematical Programming (MOdelling) System: CAMPS, The Computer Journal, Vol. 31, No. 4, pp. 364-375, 1988.

29.Marin, C. M., and M. G., Smith, Water Resources Assessment: A Spatial Equilibrium Approach, Water Resources Research, Vol. 24, No. 6, pp. 793-801, June 1988.

30.Marsten, R. E., Department of Management Information Systems at the University of Arizona at Tucson, XMP User's Guide, 1984.

31.Nakamura, M., A Mathematical Method for Generating and Comparing Alternative Plans: An Application to Regionalization of Wastewater Systems, Ph. D. dissertation at University of Illinois, 1977.

32.Nelder, J. A., and R. Mead, A Simplex Method for Fuction Minimization, Computer Journal, Vol. 7, pp. 308-313, 1965.

33.Paul, J. P., LINGO/PC: Modeling Language for Linear and Integer Programming, OR/MS Today, Vol. 16, No. 2, pp. 24-25, Apr. 1989.

34.Reichard, E. G., Hydrologic Influences on the Potential Benefits of Basinwide Groundwater Management, Water Resources Research, Vol. 23, No. 1, pp. 77-91, Jan. 1987.

35.Sagie, I., Computer-Aided Modeling and Planning (CAMP), ACM Transactions on Mathematical Software, Vol. 12, No. 3, pp. 300-317, Sept. 1985.

36.Shubert, B. O., A Sequential Method Seeking the Global Maximum of A Function, SIAM Journal of Numerical Analysis, Vol. 9, No. 3, Sept. 1972.

37.Starr, M. K., and M. Zeleny, MCDM-State and Future of the arts, in Multiple Criteria Decision Making, edited by M. K. Starr and M. Zeleny, North-Holland, New York, 1977.

38.Tang, C. C., Mathematical Models and Optimization Techniques for Use In Analysis and Design of Wastewater Treatment Systems, Ph. D. dissertation at University of Illinois, 1984.

39.Tang, C. C., E. D. Brill, Jr., and J. T., Pfeffer, Comprehensive Model of Activated Sludge Wastewater Treatment System, Journal of Environmental Engineering, ASCE., Vol. 113, No. 5, pp. 952-969, Oct. 1987.

40.Tecle, A., M. Foget, and L., Duckstein, Multicriterion Selection of Wastewater Management Alternatives, Journal of Water Resources Planning and Management, Vol. 114, No. 4, pp. 383-398, July 1988.

41.Viessman, W., Jr., and M. J. Hammer, Water Supply And Pollution Control, 4th edition, Harper & Row, New York, 1985.

42.Wang, H. F., and M. P. Anderson, Introduction To Groundwater Modeling: Finite Difference and Finite Element Methods, W. H. Freeman and COmpany, 1982.

43.Wagner, B. J., and S. M. Gorelick, Optimal Groundwater Quality Management Under Parameter Uncertainty, Water Resources Research, Vol. 23, No. 7, pp. 1162-1174, July 1987.

44.Willis, R., and P. Liu, Optimization Model for Ground-water Planning, Journal of Water Resources Planning and Management, ASCE, Vol. 110, No. 3, pp. 333-347, 1984.

VITA

Jehng-Jung Kao was born in Keelung, Taiwan, Republic of China, on 17 July 1959. He graduated from Jen-Kuo High School, Taipei, in June 1978. He then enrolled in the environmental engineering program at the National Cheng-Kung University, Tainan, Taiwan in October 1978, and received the degree of Bachelor of Science in Environmental Engineering in June 1982. He served as a second lieutenant in the Army Engineering Corp from October 1982 to August 1984 under a national ROTC program. After leaving the Army, he was employed as a Research Assistant by Institute of Environmental Engineering, National Taiwan University, Taipei, Taiwan. In August 1985, he enrolled in the graduate program at the University of Illinois at Urbana-Champaign in the area of Environmental System analysis and was appointed to a graduate research assistantship. Computer-aided systems for environmental engineering decision making problems was his major research area. He also studied the areas of decision-making processes, knowledge-based database systems and expert systems, optimization, mathematical modeling, computer graphics, user interface, management, and planning.

He has also been employed as a Research Assistant in the Environmental Systems & Modeling Team, U.S. Army Construction Engineering Research Laboratory, Champaign, Illinois from April 1988 to the present. His responsibilities are for maintaining and developing means for uploading the individual databases from an aggregate database environment on a center host. The databases include hazardous waste management (HMIS), environmental impact decision support (EIS), and environment technical information systems (EITS), which are commonly used by Army engineers.