CENTER FOR RESEARCH IN WATER RESOURCES
BUREAU OF ENGINEERING RESEARCH

College of Engineering
The University of Texas at Austin
Austin, Texas 78758-4497

WATER SUPPLY PLANNING USING AN EXPERT GEOGRAPHIC INFORMATION SYSTEM

by

Daene C. McKinney

John F. Burgin

David R. Maidment

Project Number - 04

(September 1, 1991 - August 31, 1993)

Grant Number

14-08-001-G2-48

Technical Report
No. 162
Texas Water Resources Institute
Texas A & M University
College Station, Texas 77843-2118

Technical Memorandum
CRWR 94-2
Center for Research in Water Resources
University of Texas at Austin, PRC 119
Austin, TX 78712

November 1994
(Revised June 1995)

ABSTRACT

An expert geographic information system (expert GIS) for long-term regional water supply planning has been developed. This system has been evaluated through a case study examining a 19-county study region in South Texas with several water supply sources and demand centers. The planning system is comprised of an expert system, which contains the logical rules and expertise of water resources planning experts; a geographic information system, which stores and analyzes spatially distributed water supply and demand data; and a network flow solver, to balance the flows in networks developed by the expert GIS with input from a water resource analyst. Commonly available water demand forecasts and water supply data are used in this new planning tool in an attempt to follow more rapidly the logic of current methods and permit plans to be updated and alternatives to be analyzed. Given annual yields for reservoirs, water demand forecasts and institutional requirements, the expert GIS calculates potential water supply deficits or excesses and suggests efficient and cost effective alternatives for developing additional water supplies in the event that deficits occur. The expert GIS system has been developed so that it can be expanded to include additional constraints and handle large water resources planning regions. Eventually, the system will be capable of analyzing entire river basins, given appropriate information concerning the supply and demand for water. The system has been successfully applied to the TWDB Coastal Bend planning region. The existence of generic categories of rules for regional water planning is evident from this case study. The categories include rules applicable on a statewide basis, a regional basis or a local basis. The local scale rules are specific to individual arcs in the network model representation and need to be entered individually. However, the application of the small sets of statewide and regional rules is sufficient to generate relatively realistic solutions.

TABLE OF CONTENTS

LIST OF TABLES

ACKNOWLEDGMENTS

This research was co-directed by Dr. Daene C. McKinney, Assistant Professor of Civil Engineering, and Dr. David R. Maidment, Professor of Civil Engineering.

Technical assistance from Tony Bagwell and Steve Densmore of the Texas Water Development Board helped to define the logic of producing the Texas Water Plan and subsequently the decision support system developed in this research.

Grants from the Texas Water Development Board (contract IAC94-0100) and the United State Geological Survey through the Texas Water Resources Institute (contract 14-08-001-G2048) helped to support this research.

1.0 INTRODUCTION

A central problem in the field of water resources planning and management is the efficient allocation of water supplies to meet demands. At local planning levels, where there are typically only one or two potential sources of supply and a relatively small number of demands, efficient solutions can be found by simple inspection and common sense. At the regional level, where there are frequently a dozen or more sources of supply and many times that number of demands, the allocation of supplies to meet demands becomes more difficult. In such instances, exhaustive enumeration and the direct comparison of each of the alternatives may eventually lead to efficient solutions. As the size of the planning region expands, the number of alternative allocation possibilities explodes, and simple methods become impractical. This is very commonly the case at the statewide planning level.

The Texas Legislature mandated such statewide planning in 1957 when it created the Texas Water Development Board (TWDB). The board responded in 1961 and in 1969 by producing water plans that describe the state's water resources, quantify future water needs, and propose water supply projects to meet those needs. The scope of planning increased in the 1984 water plan, which proposed conservation and environmental protection initiatives in addition to conventional water supply projects. The current water plan, promulgated by the agency in 1990 [TWDB, 1990] and updated in 1992 [TWDB, 1992], emphasizes improved overall management of the state's existing and future water infrastructure systems and proposes that the state water plan be updated on a regular and predictable basis.

The steps taken by the TWDB in creating a state water plan can be summarized as follows:

(1) Water demands are estimated for the base planning year and for 10-year intervals into the future, out to a planning horizon of 50 years. These include municipal demands for each city of grater than 1,000 population, and agricultural, industrial and other types of water demands which are aggregated by county and estimated for each of Texas' 254 counties.

(2) Available water supplies are estimated for each water supply source: the firm yields of reservoirs, the dependable flow of rivers, and the available yield of groundwater aquifers.

LIST OF FIGURES

(3) A reconciliation process is undertaken in which both supplies and demands are partitioned by county; the total demand and supply are computed for each county in each planning period; and a deficit is registered whenever projected demand exceeds available supply.

(4) In areas of deficit, a search is launched for nearby surplus supply sources that could be allocated to meet the deficit, and new projects necessary to develop those sources are identified and scheduled in each 10-year planning period. This involves a trial-and-error search among many alternatives.

(5) The initial plan so formed is debated at regional meetings throughout the state and adjusted on the basis of input from local officials.

The long history of water planning in Texas has created a solid basis for conceptualization of the problem and considerable operational experience with its solution. The goal of this research is to reformulate the conceptual model using a new set of information engineering tools, in order to improve the understanding of the choices being made in water planning and to increase confidence that good alternatives are being chosen as plans are formulated. The current planning process has potential for improvement in several respects.

The allocation process that has been used in current and previous water plans is heavily dependent upon the expertise and the judgment of a few professionals at the TWDB, and upon outdated Fortran computer programs with complicated data files. Extensive data related to the supply and demand for water in current and future decades are batch processed to identify potential shortfalls, without the aid of geographic display of the plans. Allocations are made on a county-by-county basis within major watersheds, based upon a prioritized list of suppliers for each demander. The list is adjusted incrementally by an expert analyst before each batch run, until demands are satisfied or until there is clear indication of the need for capacity expansion. This is a very tedious and difficult process, and the water allocations that evolve from it are an expression of the analyst's abilities to comprehend the system and improve the solution, rather than the result of a rigorous documentable procedure.

A prime motivation for this research has been a strong desire within the TWDB to automate this process without abandoning the assumptions and philosophies that have resulted in the present set of allocations. Automating this system would define the decision-making methodology, which, in turn, would result in more defensible conclusions and recommendations. It would also facilitate the investigation of alternative economic assumptions and expansion scenarios and allow for comparisons among them in order to ascertain the sensitivity of the current set of allocations to error in these assumptions and scenarios. Furthermore, the effects of political and environmental considerations could be determined in an objective manner secure from the distorting influences of unstated personal bias.

An automated set of procedures to allocate water resources in the state of Texas must take into consideration historical and political institutions as well as the relevant geographic, hydrologic, and economic data. The procedures must be consistent with earlier water planning philosophy and draw from its experience rather than abandon it.

Such a system can be devised based upon the capabilities of:

(1) a geographic information system (GIS) to store, retrieve, manipulate, update, and display spatially related data,

(2) an expert system to implement a set of logical rules that contain the skill of a professional analyst, and

(3) a network balancing system to find least-cost resource allocation solutions.

The development of the system has required several years of programming and testing. Numerous command ancillary computer programs have been written to generate data layers or coverages, create input files, transform and transfer data, and present results. The system was first tested on a small and completely contrived problem, which is presented as an example in Chapter 4. A 19-county case study problem has been undertaken with supply and demand data for the Texas Coastal Bend planning region. The results of applying the new planning system to this problem are presented in Chapter 5.

The solution of this planning problem required a creative approach to analysing geographic data that is more abstract than the digital display of map data. Nominally, this water planning problem involves consideration of five coverages: two for demands (cities and counties), and three for supplies (reservoirs, rivers, and aquifers). The problem is further complicated by the fact that four of these data layers are polygon or area coverages (cities, counties, reservoirs, and aquifers), and the other is a line coverage of rivers. Moving water from one area to another is an ill-defined problem, because the distance between two areas cannot be uniquely determined.

This dilemma was resolved by representing all supplies and demands by geographic points, and the allocation of supplies to demands by areas or straight lines between the points. Thus, each city is represented by the location of its central post office, each county and reservoir by their centroids, river supplies by their point of diversion, and groundwater aquifers by the centroid of the aquifer within each county. In this manner, the distance between each supply and demand point can be approximated, and the elevation of each supply and demand point can be found from digital elevation data. The cost of each allocation can then be estimated, based on the required flow, elevation difference, distance of travel, and type of conveyance system (pipeline, canal, etc.).

Using the point-line layout of supplies and demands, a new solution prototype is constructed combining three computerized methods—GIS, expert systems and a network solver—into a single system. The expert system applies rules which reduce the set of all possible allocations, from supplies to demands, to a set of feasible allocations. The network solver calculates the cost of each allocation in this set and finds the most cost-effective solution to meet the demands from available supplies. This system constitutes a new tool which affords an increased capacity to examine alternative scenarios and enables analysts to better address the scope and complexity of the allocations problem itself. In addition, demonstration of the efficacy of this approach to water resources problems suggests its extension to other spatially distributed planning problems such as electric power distribution, regionalization of wastewater treatment, and emergency services siting. The ability to compare total costs on successive runs affords a new and objective mechanism for determining the relative costs of arbitrarily imposed allocation rules. This is a by-product of creating an automated system in a way that separates the functionality of the data base, the rule base, and the solver.

2.0 LITERATURE REVIEW

2.1 OPTIMIZATION

Operations research methods have been extremely useful for optimizing both the design and management of water delivery systems for over three decades. In particular, linear programming techniques have been utilized effectively in the modeling and solution of complex resource allocation problems. These are most frequently solved with transportation and transshipment algorithms based upon simplex methods. The description of this methodology is standard fare in textbooks in the fields of water resources engineering [Loucks et al., 1981; Buras, 1972] and operations research [Hillier and Lieberman, 1974; Bradley et al., 1977]. Furthermore, it has been demonstrated that many resource allocation problems, as well as many other LP problems, can be represented in terms of single commodity flow within a system of nodes and arcs and solved by network simplex techniques [Jensen and Barnes, 1980]. This affords a significant advantage with respect to the visualization of the problem and also offers data storage and computational advantages compared to standard matrix methods.

The following is a discussion of representative literature from the field of water resources engineering wherein linear programming techniques were employed to solve network flow resource allocations problems.

A raw water supply master plan was prepared for the City of Boulder, Colorado [Brendecke et al., 1989]. The supply system is operated to meet municipal and industrial demands, provide minimum streamflows for Boulder Creek, and generate revenues from hydroelectric turbines installed in raw water and treated water transmission lines. The master plan was developed with the use of three applications of a network optimization tool, which uses the Out-of-Kilter network flow algorithm.

The water rights claims of many Indian reservations in the West are now under adjudication. Lord et al. [1989] sought to (1) develop a conceptual basis for determining Indian water rights; (2) develop an analytical procedure to provide the information needed to resolve water rights conflicts; and (3) apply this analytical procedure to a test case involving the Gila River Basin in Arizona. The methodological core of the research was a set of linked models, encompassing historical, hydrologic, economic, psychological, and institutional elements of the conflict. Hydrologic, institutional, and economic analyses of conjunctive management of surface and groundwater supplies are facilitated by the use of a network optimization model.

Streamflow increases that could be created by vegetation management on forest land along the upper reaches of the Colorado River was examined by Brown et al. [1988]. A network optimization model was used to simulate water flow, storage, use, and loss within the entire Colorado River Basin with and without the flow increases, according to various scenarios incorporating both current and future use levels as well as existing and potential institutional constraints.

A water resource optimization model was developed by Maddaus and McGill [1976] for use in long-range infrastructure planning for water supply and wastewater management. The model included a network analyzer to determine least-cost allocation of available sources of water supply (including reclaimed wastewater) to various demand points subject to certain physical constraints and water management policies, a recosting procedure for nonlinear cost functions, a digital groundwater model for simulating widespread changes in groundwater depth, and a salt balance model for simulating groundwater quality changes with time. The modeling system provided costs for the optimal water resource allocation for various sets of constraints as well as the environmental changes in the groundwater reservoir. The most cost-effective alternative was identified and used to develop a 50-year water supply and wastewater management plan for the Tucson, Arizona, regional area.

Fordham [1972] evaluated simulation as a planning and management tool for water resources in the Truckee and Carson River System in Nevada and California. A simulation model of the two-river system was constructed and then embodied in an optimization algorithm to develop "optimum" operating rules for the system as a whole. Since the demands on the system were incommensurate in economic terms and were greater than the available resource, the problem was resolved into one of allocation of the resource among the various demands. To accomplish this, the problem was formulated as a capacitated flow network and solved using the Out-of-Kilter algorithm. The reservoir releases and diversions from several flow traces were then subject to multiple regression analysis to determine "optimal" operating rules for the five reservoirs and for diversions within the system. Operating rules can be derived by this method which significantly improve overall system operation.

Brown et al. [1972] assessed the importance and the relationship of social-cultural, political and economic inputs to the decision-making process in water resources allocation. A resource allocation model was formulated in terms of network flows; however, it presented problems of assigning value units to political and social inputs and changing decision criteria and was ruled impractical. A new linear programming model was shown to have promise.

A study was performed to determine optimal water resource allocation in the Montana North Central Conservancy District [Foster et al., 1972]. The district covers several river basins and contains numerous existing and proposed facilities (dams, reservoirs, and diversion canals). The study determined the optimal operation method of all these facilities, along with the sizing of the proposed facilities in order to maximize given objective functions. Related efforts in optimal river basin utilization were surveyed, and linear programming was selected as an expedient optimization technique. The problem was formulated by identifying time stages which constitute a repetitive cycle such as a year. With these stages, it was possible to associate operational and capacity variables with network components, which are branches or nodes. Constraint equations were written to reflect network nodal continuity, capacity restrictions, and adjudications such as water rights. A numerical example was considered in which the existing and proposed facilities were aggregated to produce a small, tractable number of facilities. Linear programming was shown to be quite feasible as a decision-making technique for optimum water resource allocation.

The survey of optimization applied to water resource systems just presented shows that linear programming algorithms, in particular network flow algorithms, have been widely used to analyze many water planning problems in the Western United States. When the problem is properly formulated, cost effective solutions can be obtained. One limitation of optimization models is that real problems have many constraints that are difficult to express in the language of optimization and network flows. A second limitation is that optimization seeks a globally-optimal result, while in real planning problems, the participants are often more concerned with optimizing their own local situation than with producing a global optimum.

In this research, we have attempted to overcome the first limitation by using expert system rules as a constraint on the set of feasible networks that can be examined, and to overcome the second limitation by showing the degree of additional cost that is incurred when insistence on a local solution for part of a problem forces departure from the global optimum for the region.

2.2 INTELLIGENT GEOGRAPHIC INFORMATION SYSTEMS

Within the last 10 years, geographic information systems (GIS) have been employed by researchers and practitioners to store, display and relate the large amounts of spatially-based data characteristic of water resources management problems. By coupling GIS to other solution mechanisms, the obvious capabilities of GIS have been extended to facilitate analysis and decision-support for spatially distributed problems. This is often referred to as "intelligent GIS" and the following examples demonstrate recent practice.

Wright and Buehler [1990] demonstrated how the integration of GIS and expert systems technologies can be used to manage land and water resources. They devised a Bayesian ranking system called B-Infer, which is an additional component of the GRASS GIS, whose purpose is to identify good land use plans for military bases.

The Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) engineers and scientists have developed several software systems to support water resources and environmental decision making [Strzepek and Chapra, 1990]. CADSWES advanced decision support systems use high-resolution graphics, artificial intelligence, and GIS together on workstations. Two systems have been developed by CADSWES for governmental planning and management. At the Santa Ana Watershed Project Authority (SAWPA) in California, the QUEST (Quality Evaluation System) system links QUAL2E, EPA's principal stream water quality model, with pre-processors and post-processors. The system enables planners to quickly examine many alternatives for waste load allocation on streams, permitting them to develop a more comprehensive grasp of the impact of their decisions. A multifaceted decision support system was described which automates the U.S. Bureau of Reclamation's 24-month study of Colorado River tributary runoff. The system includes an intelligent user interface for the existing 24-month model, an expert system based on the knowledge of the current master engineers, and direct software links between the river model and forecasting systems. Both of these systems use a simplified GIS as a spatial display device for the river system.

Arnold et al. [1989] proposed a conceptual approach to the development of an intelligent GIS that incorporates knowledge processing capabilities for the surface water problem domain. Special attention was given to object oriented methodology, interfaces between numeric data and symbolic knowledge representation, and to dealing with uncertainty.

While the systems just described all use elements of artificial intelligence and GIS, none of them uses a full-fledged expert system and GIS, which is what has been done in the research reported here. The Nexpert Object expert system and the Arc/Info GIS are coupled to form a synthesized system for relating the spatial objects upon which expert system rules operate, which is directly connected to the corresponding geographic features and their tabular attributes in the GIS. This research is the first time this task has been accomplished in water resources planning with problems of significant spatial scale.

2.3 EXPERT SYSTEMS

Expert systems constitute a segment of computer technology in the broader field of Artificial Intelligence. These systems are characterized by a set of facts or objects and a set of rules, known collectively as a knowledge base, and they have incorporated within them an inference mechanism that allows the sequencing of processing rules to form conclusions [Jackson, 1986]. There are two types of sequencing or chaining of rules: backward and forward. Backward chaining is best seen in the classic example of expert systems—medical diagnosis. If one condition is true, check another condition upon which the first condition depends; if it is true check another, and if it is also true, continue checking and searching back through the knowledge base until a cause is identified. Conversely, forward chaining is the process of determining new conclusions from a given set of facts. For example, given the status of factors critical to the operation of each vehicle in a fleet of vehicles, and given a set of rules about the factors required for a vehicle to operate, an expert system can report which vehicles are inoperable and prepare a list of the needed repairs.

Expert systems that have been carefully designed and developed can perform in a limited domain about as well as a professionally trained human [Jackson, 1986]. The computer system relies primarily on rapid search procedures while a professional relies on more creative and connective mechanisms of associative memory (deduction, intuition, and inference). If the body of knowledge is sufficiently well defined and of moderate size, the efficiency of the search may be unimportant, and search by a machine may in fact be preferable to human reasoning in that there is no chance for oversight, and the logic of the search process is explicitly defined.

A typical expert system has a few hundred rules and definitions, and it usually contains one or more heuristic connections that allow it to cut to the essence of a problem without having to process all of the rules it contains. This closely follows the reasoning process of human experts and speeds up processing if a reliable heuristic exists. In very recent literature, some researchers have explored the utility of expert systems with respect to their ability to capture and define experience-based reasoning for decision support. The following are examples of this work.

Nieuwkamer and Winkelbauer [1992] developed a rule-based expert system named MEXSES for environmental impact assessment of water resources development projects. The system has a link with external models and makes use of their computational power during the inference process. The information in MEXSES on which the impact assessment is based, is stored in the system's knowledge base in the form of production rules. The expert system's inference engine interprets the information in the knowledge base and generates a conclusion about the environmental impact of the problem on the system under consideration. A model-specific program was created to interface with the other programs. The problem of reservoir sedimentation was selected to test the link. The main result of the integration of the reservoir sedimentation model in the hybrid rule base is that the established link demonstrated the feasibility of invoking a numerical model from the MEXSES expert system and using the model results in the reasoning process.

Palmer and Holmes [1988] described a decision support system used to aid in drought decisions. Its components included an expert system, a linear programming model, data base management tools, and computer graphics. The expert system incorporated operator experience and intuition using a rule base developed through interviews with management personnel from the Seattle Water Department. The expert system integrated the other programming techniques into a single system. A linear programming model determined system yield and optimal operating policies for past hydrologic regimes. Data base management and graphics software stored and facilitated the display of over two thousand operating policies to decision-makers. The system provided user-friendly support to help decision-makers explore a wide range of management alternatives.

Greathouse et al. [1989] report on over 40 small- and large-scale decision support systems. These range from data bases with a few obvious rules to systems capable of evaluating conditions of toxic hazard and recommending remediation measures. One of the chief advantages cited by the authors of using expert systems in environmental control is that of attaining consistency in agency response from one site to another. The systems allow for the distribution of scarce expertise and will find many uses such as report generation, emergency response assistance, hazard identification, planning and training.

When expert systems were first introduced about 10 years ago, great promise was held out for the intelligence it was thought that they would bring to problems involving complicated logic. Much of that promise faded away when it became clear that the class of problems to which expert systems can be applied is quite small, and that trying to apply expert systems to very general problems was of limited utility because there are often exceptions to rules and special cases which are sufficiently influential that they govern the final result. Experience with the use of expert systems suggests that they should be applied to problems where

(1) the logical rules that should govern the system can be explicitly identified and there really is no other way of expressing this logical system.

(2) there are a very large number of repeated applications of the same type of logical systems.

The water planning problem studied here generally satisfies these conditions because planning rules and guidelines are stated in a logical, even legal, language that is very difficult to quantify by other means, such as by writing optimization codes. Also, in Texas, the whole state is conceptually represented in a consistent way for water planning so a very wide area of repeated applications is possible.

3.0 METHODOLOGY

Developing systematic water plans for Texas has a long history which dates back to the drought of the 1950's. That event created the recognition in the state that systematic water planning was needed, led to the collection of water use data from all major water users beginning in the early 1960's, and to the development of plans for large-scale state water projects in the late 1960's and early 1970's. However, voters did not approve large scale water transfer schemes to transfer water from East Texas to the High Plains because of their high cost, and later Texas Water Plans during the 1980's and 1990's have been confined to more modest projects for water transfers within regions, and, in particular, within river basins.

The differing legal doctrines governing ground and surface water withdrawals make it much more difficult to evolve balanced plans for the conjunctive utilization of groundwater and surface water than for surface water alone. The state has more than 4500 water supply entities such as river authorities, water districts and cities, which have an uncounted number of individual water supply facilities. There is a very large number of water supply contracts between suppliers and end-use demanders, and sometimes between suppliers and wholesalers, and then between wholesalers and end-use demanders.

In this report are presented two applications of the automated planning system: (1) a special study of a hypothetical problem connecting four supplies with three demands, devised by Texas Water Development Board planning staff to typify common planning complexities and to better ascertain the sensitivity of the basic model too changes in rules or constraints; and (2) a case-study of a 19-county region in South Texas which typifies the overall planning problem.

3.1. THE TWDB WATER PLANNING PROCESS

The history of water planning in Texas has led to the creation of a simplified numerical planning scheme which identifies the principal features of the planning problem without getting immersed in the endless details relating to each particular water supply system. A number of assumptions are made.

3.1.1 Planning Horizon

A planning horizon of 50 years is adopted, beginning with a base year and continuing in 10-year intervals. For example, if 1990 is the base year, then calculations are done for conditions in years 1990, 2000, 2010, 2020, 2030, and 2040. It is inevitable that economic and population projections made over such a long horizon are uncertain. Indeed, one of the motivations for having an automated planning system is to be able to update the plans on a frequent basis as economic conditions change.

3.1.2 Annual Yield Estimates

All planning estimates are for total annual quantities in each planning year. For example, for a city, the quantity considered is the annual volume of water demand for the city, not the demand in the peak day, which is an important design criterion for city water treatment systems. The focus is on the overall water balance of the state rather than the supply and treatment system of a particular water user. Ratios of monthly to annual demand are used in estimating firm yields of reservoirs, but the planning number is still an annual yield figure. The effects of drought on demand are included by increasing the expected demand by a factor derived from historical demand data during normal and drought weather conditions.

3.1.3 Municipal Demand

The annual municipal water demand for each city greater than 1000 population is calculated as the product of population and an annual water use per capita estimate based on historical water use data. Population forecasts are made by a separate procedure which considers demographic and economic factors in each region of the state. In some Texas Water Plans, the effect of water conservation has been included by adjusting the water use per capita estimate. Municipal demand for cities of less than 1000 population and for rural residents is included in the corresponding county water demand estimates. Where a city lies across a county boundary, the portion of the city's demand in each county is estimated to allow later accounting of total demands by county.

3.1.4 Agricultural Demand

The annual irrigation water demand is calculated by taking the product of the irrigated area and an irrigation water use per unit area, determined using climatic data and a soil water balance. Crop water use estimates are done seperately for each crop. The area of each crop irrigated in each county is found by making surveys, so that an annual irrigation water demand totalled across all crops can be calculated for each county. Trends in the irrigated area from historical surveys are used to project the irrigation demand over the planning horizon in each region of the state. Irrigation water demand is an important quantity because it constitutes more than 80% of the consumptive water use in the state, that is, the water withdrawn from surface and groundwater sources that is lost to evapotranspiration. Livestock water use is based on estimates of animal populations in each county multiplied by water use per animal.

3.1.5 Industrial and Other Demands

Annual demands are computed by county for industrial water use divided into categories using the SIC, or Standard Industrial Classification, codes. Water demands are found by multiplying the units of production or dollars of value added in production, by the water use per unit of production or dollar of value added. Special demand estimates are made for very large industrial water users including steam-electric power, oil and gas refining, petrochemicals production, and mining.

3.1.6 Demand Scenarios

The total demand is aggregated for Texas' 254 counties by totalling the municipal, agricultural and industrial demands for all water users in each county. Several demand projection scenarios are constructed based on lower or higher estimates of economic and population growth in the state, normal and dry weather conditions , and alternative levels of water use efficiencies. A water allocation between supplies and demands produced by the planning system described in this report relates to a particular demand scenario.

3.1.7 Physical Water Supplies

The planning process uses physical estimates of available water supplies rather than contracted estimates. For example, a reservoir system may have a firm yield which is totally committed based on contracts for future supplies, but some of these contracts may be for industries or facilities as yet not constructed, so that the reservoir actually has additional supply capacity available beyond its present water delivery volumes. It is this physical capacity for water supply which is the focus of the planning process.

3.1.8 Naturalized Flows

The flows in the principal rivers in Texas have been affected by reservoir construction and by withdrawals along the river. Naturalized flows have been reconstructed for the principal river systems, and these flows are used for allocation of water for direct withdrawal from rivers, taking into account the simultaneous withdrawals of upstream and downstream users. A water adjudication process has been carried out in several of the major river basins where historical water rights that were not being fully utilized have been reallocated to newer water users.

3.1.9 Firm Yield of Reservoirs

The firm yield of a reservoir is the mean annual demand which can be supplied from the reservoir throughout the critical drought of record. All allocations for uninterruptable supplies are based on the firm yield. In some cases, contracts for additional supplies which can be interrupted during droughts are also made, but these secondary supplies are not considered in the planning process at the statewide level.

3.1.10 Dependable Yield of Groundwater Systems

The dependable yield of a groundwater system is equal to its mean annual recharge rate. This rate is, however, a rather elusive quantity, because the rate of recharge of some aquifers is significantly influenced by the degree to which they are pumped. Some Texas aquifers are being mined, that is, their levels are being progressively lowered by pumping in excess of annual recharge. Groundwater studies are made of each aquifer to determine its dependable supply rate. For planning purposes, the aquifers are divided by counties into separate supply sources. Groundwater availability is determined as a combination of dependable yields and managed withdrawals depending on the aquifer. One of the extensions desired by the TWDB to its present planning methods is the capacity to construct a grid over each aquifer and to evaluate the effect of pumping in each county on the overall water balance of the aquifer. Although that is not accomplished in the automated planning system presented here, the fact that this system is based on a GIS layout of the data, and that groundwater models can be connected to GIS, means that at a later time this detailed groundwater planning feature could be added to the planning system.

3.1.11 Water Allocation

The heart of the water planning process is the procedure of water allocation in which available supplies are matched with demand requirements. This is done by an allocation or matching process in which individual water sources, such as a particular reservoir or aquifer supply, are allocated to particular demands such as a city or agricultural demand in a county. This is done under the constraints that the supply allocated from a particular source cannot exceed its available capacity, and the requirements of each demand must be met. After the allocation is made, areas of deficit are located where allocated supplies are less than demand requirements, and a search is launched for additional supply sources which could meet these demands. Conceptually, a region of analysis is specified around the deficit area, potential or existing supply sources with additional unallocated capacity are identified, and cost estimation of each potential allocation among these additional sources is done to identify the most cost-effective solution. This solution has ripple effects, because a supply allocated to one demand is then no longer available to meet demands elsewhere. The current planning process involves many iterations of this process, trying at each iteration to arrive at a more reasonable and cost-effective solution from the overall viewpoint. Such a solution may not satisfy local interests, however, so compromise is required.

The water planning system in this report helps clarify the process of water allocation by using rules to describe the logic of how the allocation is actually being carried out, rather than relying solely on the intuitive judgment of the planners. The automated system allows simultaneous rather than sequential determination of allocations so that the overall cost-effective allocations are obtained. The 19-county case study region in South Texas used in this study was chosen to surround the city and region of Corpus Christi for which water supply shortages are projected in the coming decades, so allocations of supply from more distant sources are needed.

3.2 MODEL CONCEPTUALIZATION

In the water allocation problem, areally distributed entities (reservoirs, aquifer, counties, cities, etc.) are conceptualized as lumped parameter systems whose properties are assigned to representative points lying within their geographic boundaries. Thus, the real planning problem which is described in geographic space by areas, lines and points, is represented simply by points and lines, or more specifically by a directed graph; that is, by a set of nodes and arcs (directed line segments) between them. The requirements of spatial topology call for the following rules:

(1) a node is a vertex or point.

(2) an arc is a link joining a pair of nodes.

(3) a path is an ordered sequence of arcs in which the initial node of each arc is the terminal node of the preceding arc in the sequence and all of the nodes in the sequence are distinct.

(4) for an arc (i,j), node i is called the "from-node" and node j is called the "to-node".

(5) arcs can join only at nodes.

As Chen [1990] shows, a complete directed graph can consist of many disconnected subgraphs of nodes and arcs, which is how the water allocation problem actually exists. There are many local water supply systems rather than a single large interconnected system. In fact, extension of infrastructure with new pipelines or canals is sometimes used to connect previously disconnected systems in order to make better overall use of water supplies.

In the conceptual water allocation model, we distinguish two types of demands: city (>1000 population) and county demands; and three types of supplies: reservoirs, aquifers and river diversions. This classification does not allow for requirements for instream flows and for bay and estuary flows so that there are additional demands for natural resources management that need to be included later. There are at least three levels of abstraction at which this conceptual model can be expressed: (1) a geographic representation, (2) a functional representation, and (3) a planning representation, as illustrated in Figure 3.1 and described below.

In the *geographic* representation, the features making up the water system, such as cities, counties, rivers, etc., are represented in their natural GIS format, that is, as points, lines, polygons, etc., in various data layers. The geographic representation is suitable for displaying digital maps of the problem, but is limited in its use for planning because of the disconnection of features between data layers.

In the *functional* representation, the geographic features are abstracted into a node-link network, where each node represents a particular area feature (e.g., representing a county by its centroid) or a particular control point on a linear feature such as a river diversion point or a point where instream flow requirements are defined. The functional representation is a detailed schematic diagram of the physical elements in the water system. All of the elements in the functional representation are actual physical items, either those that presently exist or those that could in the future be constructed. The yields of supply sources, the demands of end users, and the capacities of transmission facilities can be defined in the functional representation, as can the costs of constructing new facilities and of transporting water through existing systems.

Finally, progressing to the *planning* representation, a further abstraction of the functional representation is made in which a new network of allocations between supply and demand nodes is defined. In the planning representation, no new node locations are defined, but what is defined are allocation arcs (and perhaps new "planning" nodes), directly connecting node locations in the functional representation. In other words, a particular arc in the planning representation is a water allocation between two node locations that may, in the functional representation, require transmission of water along several links and through various kinds of nodes. A node location in the functional representation that represents a city serving as a wholesaler may, in the planning representation, be both a supply node and a demand node, such that the total demand required for the wholesaler is the sum of its own demand and the demands of the entities it serves. Its supply capacity is equal to the sum of the supplies coming into it. Thus, the "demand" requirement for a particular city node in the planning representation can be greater than the physical demand required by that city in the functional representation. It is in the creation of the planning representation that the legal, environmental and institutional constraints may be brought to bear, such as limitations on interbasin water transfers, rules about water supply districts supplying the cities within their region, and so on.

For implementation in a GIS, the planning representation has the significant advantage that all supplies and demands are represented as points, and they can be collected into a single data layer. Moreover, a particular allocation plann can be displayed as a geographic coverage of lines between points of supply and demand, which is a very useful mechanism for planners to visualize the planned allocation.

Ignoring for the moment the particular way in which entities are represented in a GIS—an expert system, data files, or a network flow algorithm—let us consider real water systems and the way in which they can be symbolized. That is, we want to create abstractions of water entities at various levels, so as to clarify the basic nature of the problem we are examining. Once an adequate conceptual model that reflects the elements and issues we wish to consider in the real system is constructed, then we can turn to the available technologies and discuss how to represent the conceptual model by means of various kinds of computer programming and software tools. It may occur that the practicalities of that process will force compromises on the conceptual model.

3.2.1 Water Entities

Water entities are defined as geographic features which store, transmit, or use water (see Table 3.1). There are two types of each kind of entity. Storage entities consist of surface storage facilities (reservoirs or lakes), and subsurface storage facilities (aquifers). Transmission entities consist of surface channels (rivers, canals), and pipelines. Usage entities consist of cities greater than 1000 population, and counties.

Natural resource entities such as bays, estuaries, fish hatcheries, endangered species habitats, and the like, typically require that water be released for their use or maintenance. They can be either real usage entities (bay and estuary) requiring water to be released that is then lost from the system, or they can be transmission entities upon which instream flow requirements are defined.

Each of the storage and usage entities is assumed to be a spatially discrete areal feature (at the geographic and functional representation levels) whose properties can be attached to a point located within its actual (planning level) geographic extent, though not necessarily at its centroid. These are then referred to as storage and usage nodes, respectively. Each of the transmission entities is a linear feature which can be symbolized by a straight line or a set of straight lines connecting actual geographic points on the feature.

3.2.2 Nodes

The above discussion implies that there are five kinds of nodes. These nodes and their respective symbolic representations are presented in Table 3.1. In the event that a particular geographic entity, such as an aquifer, is too large to be represented as a single entity, it can be broken into several entities connected by a natural transmission system as shown in Figure 3.2.

3.2.3 Arcs

All arcs are directed arcs, that is, they have a from-node and a to-node. A supply is a combination of a node and an outgoing arc as in Figure 3.3a. A demand is a combination of a node and an incoming arc as in Figure 3.3b. The instream flow at a particular point on a river would thus be symbolized as in Figure 3.4a, while bay and estuary flow would appear as in Figure 3.4b at the end of the river transmission path.

3.2.4 Allocations

An allocation is a physical transfer of water between two nodes in the network. The nodes may be close together or far apart, and the allocated water may be transmitted through several nodes between its origin and its destination. Thus, we have a supply node, a demand node, and an allocation which passes between them. The allocation is a defined quantity (and sometimes defined quality) of flow. These relationships are depicted in Figure 3.5. The cost C associated with an allocation is a function of the volume of water moving through the transmission entities. An allocation arc is described by 5 attributes as shown in Table 3.2.

3.2.5 Wholesalers

A water entity that both supplies and demands water functions as a wholesaler. Thus, the conceptualization of a large city having a supply from a reservoir that then serves a smaller neighboring city might look like Figure 3.6. In the planning network representation, the supplier-wholesaler-demander relationship is modeled as in Figure 3.8.

3.2.6 Allocation network

An allocation X_{ij} is the amount of water allocated from node i to node j. This is symbolized by an arc (i, j) between two nodes of the network. Allocations are characterized by the amount of water transferred between them W, by its quality q and the transport cost $C(i, j, W)$ which is a function of the amount of water shipped along the arc and the route of the arc. If we assume that demander j pays supplier i the cost $C(i, j, W)$ of delivering amount of water W, then we can represent the allocation as

$$X_{ij} = X(i, j, W, q, C)$$

The allocation of water from a supply node i must not exceed the available supply S_i at that node, or

$$\sum_i X_{ij} \leq S_i$$

where S_i is the capacity of the i-th supply. Similarly, the allocation of water to a demand node j must be less than the amount D_j demanded at that node, or

$$\sum_j X_{ij} \geq D_j$$

3.2.7 Cost apportionment

Suppose a particular allocation X_{ij} involves a cost to a supplier of C_s and a cost to a demander of C_d simply to achieve the transfer of water and in addition the demander has to pay the supplier P for the water received. The total real cost in economic terms is $C_s + C_d$ but the suppliers cost is $C_s - P$ and the demanders cost is $C_s + P$ as shown in Figure 3.8.

The total net cost or benefit to the suppliers of all such allocations is

$$C_i = \sum_i (C_s - P)_{ij}$$

where $(C_s - P)_{ij}$ is the net cost or benefit of allocation X_{ij}. Similarly, the total net cost to the demander is

$$C_j = \sum_j (C_s + P)_{ij}$$

(3) A network flow solver to balance the flows on the resulting network in order to satisfy the demands at minimum cost.

3.3.1 Geographic Information System

The first step is to load all of the pertinent data into the GIS. Items that are alike are stored together in the data base and related to the whole by their geographic coordinates through the process of creating a GIS coverage. A coverage is a tabular organization of like items, (e.g., points, lines, and polygons) wherein the geographic features appear as rows in the table and the characteristics of the features, known as attributes, appear as columns. A coverage is analogous to a layer in a multi-layer thematic map. The number of coverages and the number of items in each coverage is virtually unlimited. For example, a single coverage might detail municipal boundaries; another might identify the location of water meters; several might be employed to describe the extent of vegetation communities and soil types. The themes are countless and depend upon the kinds of data available and the types of problems that are at hand. Whereas the coverages are composed of elemental items such as points, lines, and polygons, the elements themselves may have attached to them other attributes such as identification numbers, names, dates, physical quantities, status flags, etc. GIS capabilities with respect to selective retrieval and depiction are obvious, but of equal importance is their capacity to be utilized as data base managers to efficiently store, modify, update, and relate large amounts of information.

Initially, all the data are stored in three GIS coverages (see Table 3.4). Two point coverages, SUPPLY and DEMAND, are defined to contain the raw data describing the supply and demand information. A line coverage, PARC, describing the potential arcs or links by which water may be delivered is generated from the information in the SUPPLY and DEMAND coverages and from external definitions and assumptions. The first and second group of attributes of the PARC coverage are inherited from the endpoints of the allocation arc, i.e., the corresponding entry in the SUPPLY and DEMAND coverages. The third group of attributes in the PARC coverage are evaluated separately in the GIS, expert system, or network flow solver as needed. This third group of attributes contains the intrinsic qualities of an allocation arc such as its length, capacity, unit transport cost, flow, and feasibility. Figure 3.9 depicts these tables and their structure. The GIS affords powerful capabilities with respect to manipulating the data within coverages. Several functions aid in the creation and maintenance of the data sets. Other functions are available for computing distance, area, line intersections, polygon overlays, etc. Selective depiction of the data contained within the coverages is also facilitated by the GIS.

characterized by the amount of water transferred between them W, by its quality q and the transport cost $C(i, j, W)$ which is a function of the amount of water shipped along the arc and the route of the arc. If we assume that demander j pays supplier i the cost $C(i, j, W)$ of delivering amount of water W, then we can represent the allocation as

$$X_{ij} = X(i, j, W, q, C)$$

The allocation of water from a supply node i must not exceed the available supply S_i at that node, or

$$\sum_i X_{ij} \leq S_i$$

where S_i is the capacity of the i-th supply. Similarly, the allocation of water to a demand node j must be less than the amount D_j demanded at that node, or

$$\sum_j X_{ij} \geq D_j$$

3.2.7 Cost apportionment

Suppose a particular allocation X_{ij} involves a cost to a supplier of C_s and a cost to a demander of C_d simply to achieve the transfer of water and in addition the demander has to pay the supplier P for the water received. The total real cost in economic terms is $C_s + C_d$ but the suppliers cost is $C_s - P$ and the demanders cost is $C_s + P$ as shown in Figure 3.8.

The total net cost or benefit to the suppliers of all such allocations is

$$C_i = \sum_i (C_s - P)_{ij}$$

where $(C_s - P)_{ij}$ is the net cost or benefit of allocation X_{ij}. Similarly, the total net cost to the demander is

$$C_j = \sum_j (C_s + P)_{ij}$$

(3) A network flow solver to balance the flows on the resulting network in order to satisfy the demands at minimum cost.

3.3.1 Geographic Information System

The first step is to load all of the pertinent data into the GIS. Items that are alike are stored together in the data base and related to the whole by their geographic coordinates through the process of creating a GIS coverage. A coverage is a tabular organization of like items, (e.g., points, lines, and polygons) wherein the geographic features appear as rows in the table and the characteristics of the features, known as attributes, appear as columns. A coverage is analogous to a layer in a multi-layer thematic map. The number of coverages and the number of items in each coverage is virtually unlimited. For example, a single coverage might detail municipal boundaries; another might identify the location of water meters; several might be employed to describe the extent of vegetation communities and soil types. The themes are countless and depend upon the kinds of data available and the types of problems that are at hand. Whereas the coverages are composed of elemental items such as points, lines, and polygons, the elements themselves may have attached to them other attributes such as identification numbers, names, dates, physical quantities, status flags, etc. GIS capabilities with respect to selective retrieval and depiction are obvious, but of equal importance is their capacity to be utilized as data base managers to efficiently store, modify, update, and relate large amounts of information.

Initially, all the data are stored in three GIS coverages (see Table 3.4). Two point coverages, SUPPLY and DEMAND, are defined to contain the raw data describing the supply and demand information. A line coverage, PARC, describing the potential arcs or links by which water may be delivered is generated from the information in the SUPPLY and DEMAND coverages and from external definitions and assumptions. The first and second group of attributes of the PARC coverage are inherited from the endpoints of the allocation arc, i.e., the corresponding entry in the SUPPLY and DEMAND coverages. The third group of attributes in the PARC coverage are evaluated separately in the GIS, expert system, or network flow solver as needed. This third group of attributes contains the intrinsic qualities of an allocation arc such as its length, capacity, unit transport cost, flow, and feasibility. Figure 3.9 depicts these tables and their structure. The GIS affords powerful capabilities with respect to manipulating the data within coverages. Several functions aid in the creation and maintenance of the data sets. Other functions are available for computing distance, area, line intersections, polygon overlays, etc. Selective depiction of the data contained within the coverages is also facilitated by the GIS.

3.3.2 Expert System

The data structures used in the GIS and expert system are different and yet they have some striking similarities. Since the GIS is a relational database and the expert system is an object-oriented expert system shell, the transfer of data between them requires a translation of the data structures. The GIS database is organized in tabular form: three coverages with attribute tables, each with multiple rows and multiple columns. Each coverage in the GIS corresponds to a class of objects in the expert system. Each of the rows in the coverage attribute table becomes an object in a class and all of the GIS tabular attributes become properties of the associated object in the expert system. Objects inherit their property types and in some cases their property values from their parent class. These two data models, relational data base and object oriented, have a direct correspondence. As the data are transferred from the GIS to the expert system, it is necessary to translate from relational database structures to object-oriented data structures. The mapping of items into the object oriented data structures is controlled by data import specifications prior to the actual transfer. This translation is quite straightforward and is handled routinely by the data import/export features of the GIS and expert system software. The object oriented data structure is depicted in Figure 3.11.

Central to the expert system is a production system which consists of a rule-set, a rule-interpreter, and working memory. Working memory is examined and modified by this production system as it applies the rule-interpreter to the rule-set. In backward chaining, the rules are triggered by an initiating suggested hypothesis which, in conjunction with the backward chaining of successive hypotheses controls the activation and selection of subsequent rules at each cycle of logic processing.

In the water allocation problem, the rules have the effect of modifying the set of values associated with the attributes of the PARC objects. By this process, the cost and feasibility of individual objects (arcs) within the PARC class may be adjusted by the actions of rules that refer to the class as a whole. Rules may also refer to individual objects or to sub-classes of objects.

After the data represented in the class of potential arcs have been processed in the expert system, the modified data set is transferred back to the GIS and the PARC coverage is updated with the updated information.

3.3.3 Network Flow Solver

The next step is to find the least cost set of flows on the network of allowable arcs remaining after the expert system rule processing. This solution must satisfy the water demands without exceeding either the available supply or the arc capacities. This is a classic problem in Operations Research known as the transportation problem and is easily conceptualized in the network form as depicted in Figure 3.11.

The nodes on the left-hand side of the network shown in Figure 3.11 represent the sources of supply, and the nodes at the right represent the demands. The lines between nodes indicate the allowable arcs, and the information in brackets and parentheses indicates the costs and constraints for the problem. This characterization of the transportation problem may be formulated as a linear programming problem as follows:

Minimize $$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} \le s_i \qquad \forall_i$$

$$\sum_{i=1}^{m} x_{ij} = d_j \qquad \forall_j$$

$$x_{ij} \ge 0 \qquad \forall_i, \forall_j$$

A slightly more general network flow model can more accurately represent the water allocation problem we are considering here. In particular, intermediate nodes can be used to portray transshipment points in the network. Transshipment arises when we consider run-of-river sources and large municipalities acting as wholesalers of water to smaller demanders. The depiction of the network is presented in Figure 3.12. This mathematical programming model can be solved by network simplex techniques as described by Jensen and Barnes [1981].

Once the network flow problem is solved, the flow on each arc is returned to the GIS PARC coverage and inspection is performed to determine if any assumptions of the planning process have been violated. If necessary, the problem can be resolved using the flows from previous iterations to update the assumptions. This step may be necessary because the unit transport cost is itself a function of the decision variable, flow.

Table 3.1 Water Entities or Features Which Store, Transmit, and Use Water

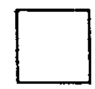
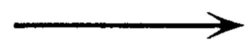
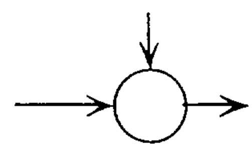
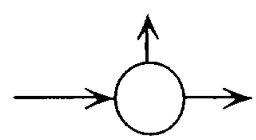
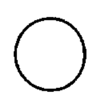
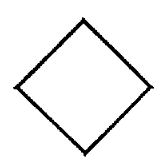
Water Entities	Description	Symbol
Storage Entities	Surface storage (reservoir or lake)	
	Subsurface storage (aquifer)	
Transmission Entities	Natural channel (river, canal, subsurface leakage)	
	Pipeline	
	Return flow	
	Diversion	
Usage Entities	City (pop. > 1000)	
	County	
	Natural resources	

Table 3.2 Attributes of a Water Allocation From a Supply i to a Demand j

Attribute	Symbol	Units
Source node (origin)	S	(x-y coords)
Demand node (destination)	D	(x-y coords)
Amount	W	(ac-ft/yr)
Quality (perhaps several constituents)	q	(mg/l)
Cost	C	($)

Table 3.3 Attributes of a Water Allocation Problem

Set	Symbol	Characteristics
Water supplies or sources	$s_1 \cdots s_N$	$a_1 \cdots a_I$
Water demands	$d_1 \cdots d_M$	$b_1 \cdots b_J$
Potential arcs	$P_{11} \cdots P_{NM}$	$c_1 \cdots c_k$
Allocations	$X_{11} \cdots X_{NM}$	

Table 3.4 Coverages in the Water Allocation GIS Data Base

Coverage	Symbol	Attributes
SUPPLY	$S \doteq \{s_1, s_2, \cdots s_n\}$	$A \doteq \{a_1, a_2, \cdots a_i\}$
DEMAND	$D \doteq \{d_1, d_2, \cdots d_m\}$	$B \doteq \{b_1, b_2, \cdots b_j\}$
PARC	$P \doteq \{p_{11}, p_{12}, \cdots, p_{1m}, \cdots, p_{n1}, \cdots, p_{nm}\}$	$C \doteq \{a_1, \cdots, a_i, b_1, \cdots, b_j, e_1, \cdots, e_k\}$

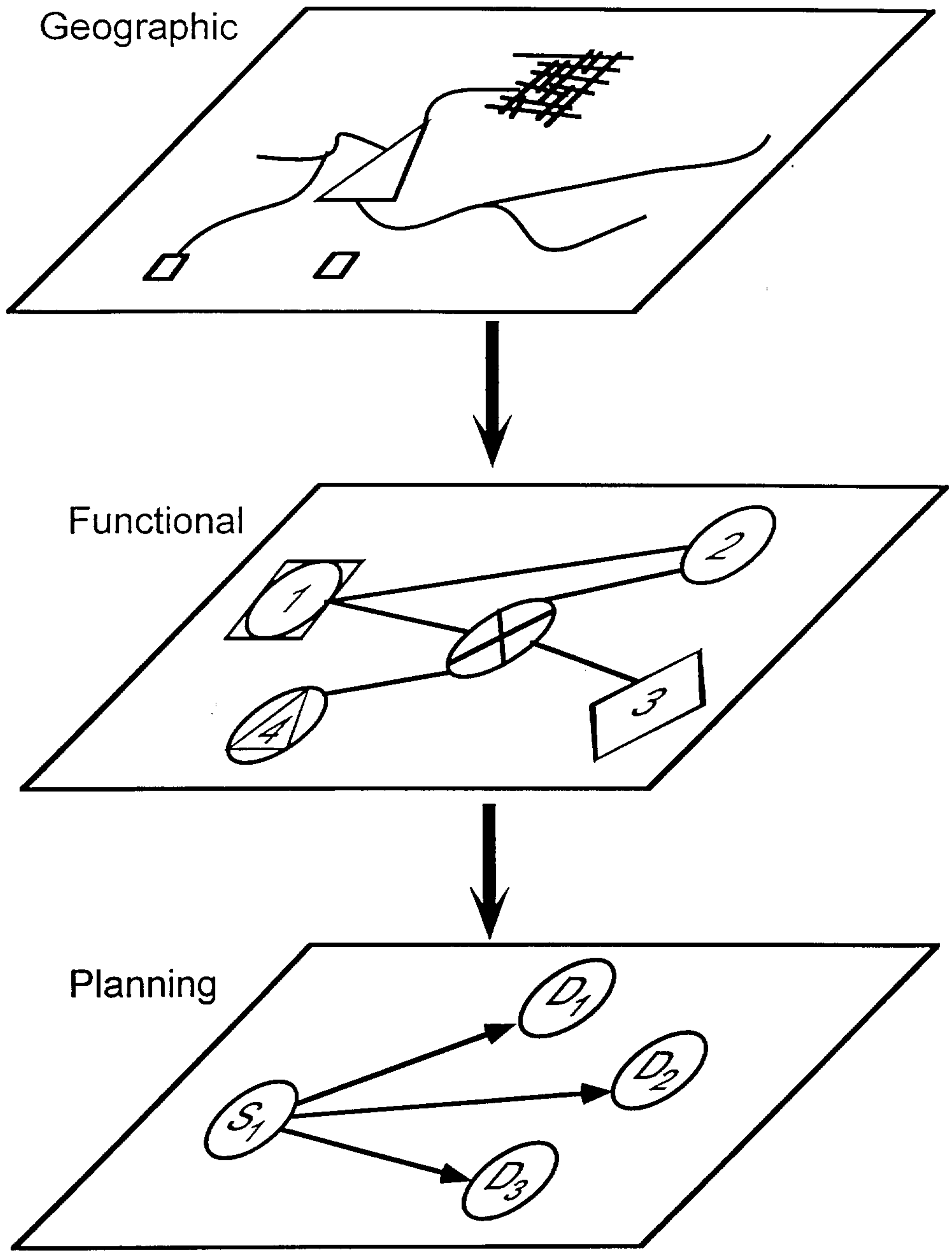


Figure 3.1 Levels of abstraction in the water allocation problem.

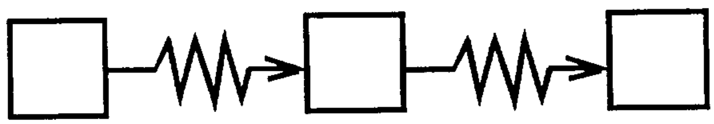


Figure 3.2 Functional level representation of a water entity decomposed into three component parts that have natural transmission routes between them.

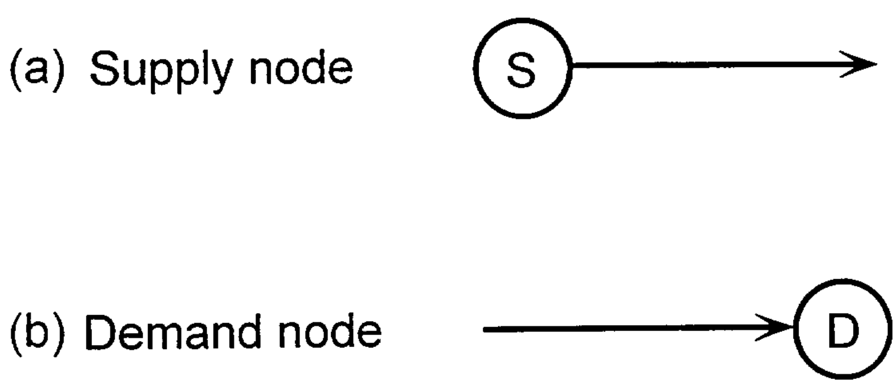


Figure 3.3 (a) Functional level representation of a supply node; and (b) functional level representation of a demand node.

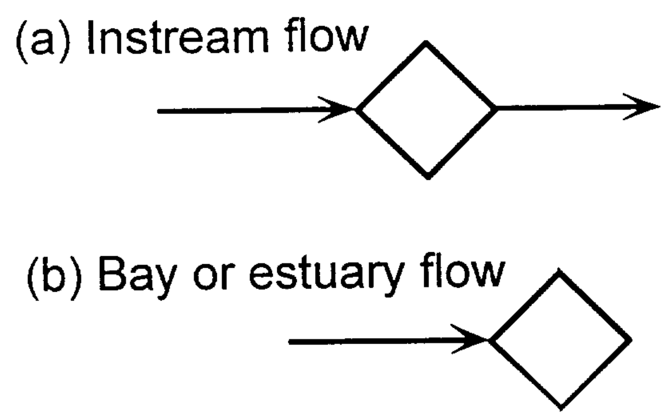


Figure 3.4 Functional level representation of a natural resources node, (a) represents an instream flow, and (b) represents a bay or estuary release.

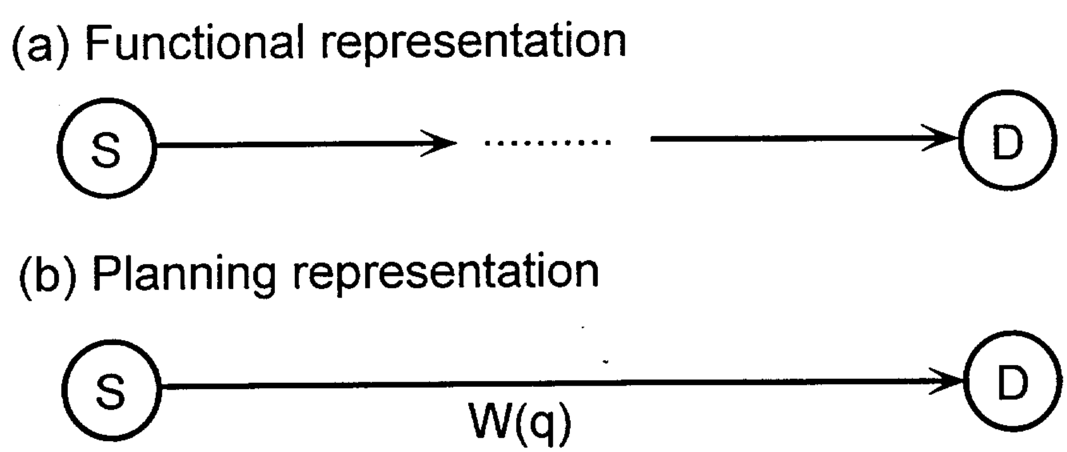


Figure 3.5 Representation of an allocation, *W(q)*, (a) functional representation, and (b) planning representation

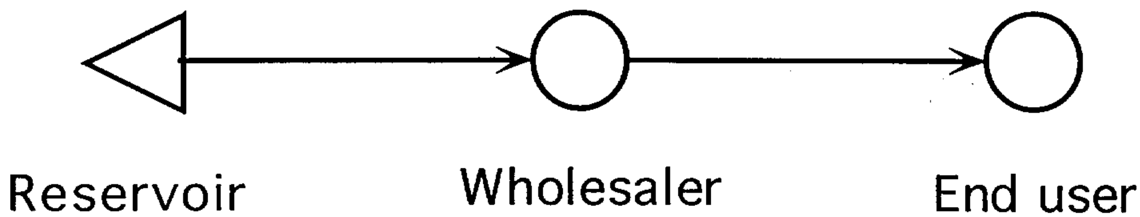


Figure 3.6 Functional representation of a supplier, a wholesaler, and a demander of water.

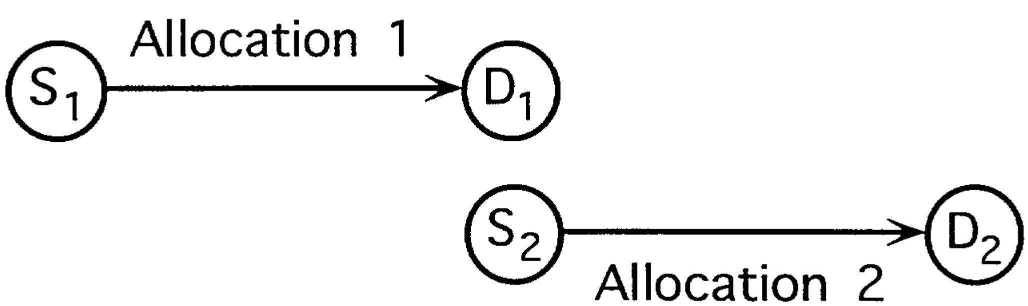


Figure 3.7 Planning representation of a supplier, a wholesaler, and a demander of water.

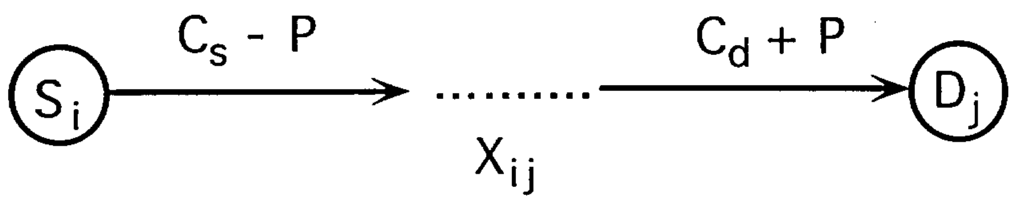


Figure 3.8 Representation of supplier, wholesaler, and demander costs for delivering water.

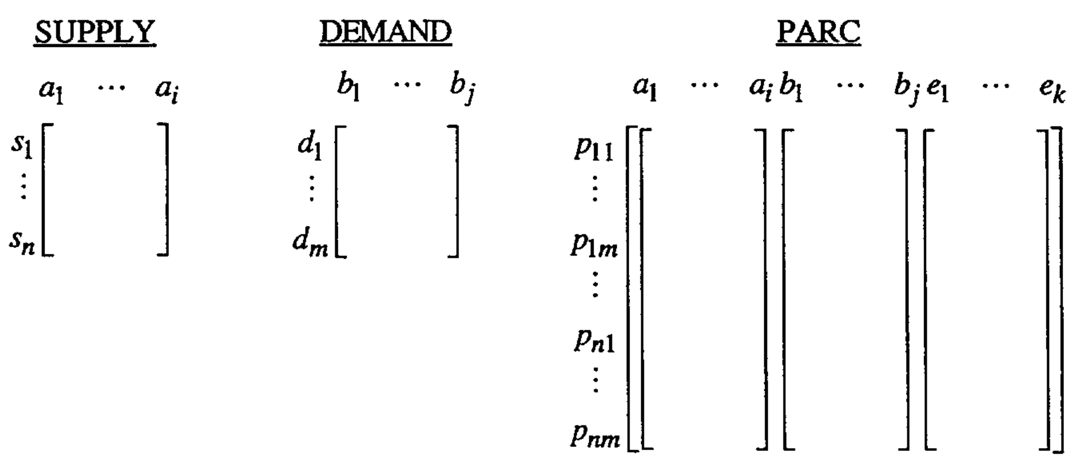


Figure 3.9 Water allocation problem GIS coverages.

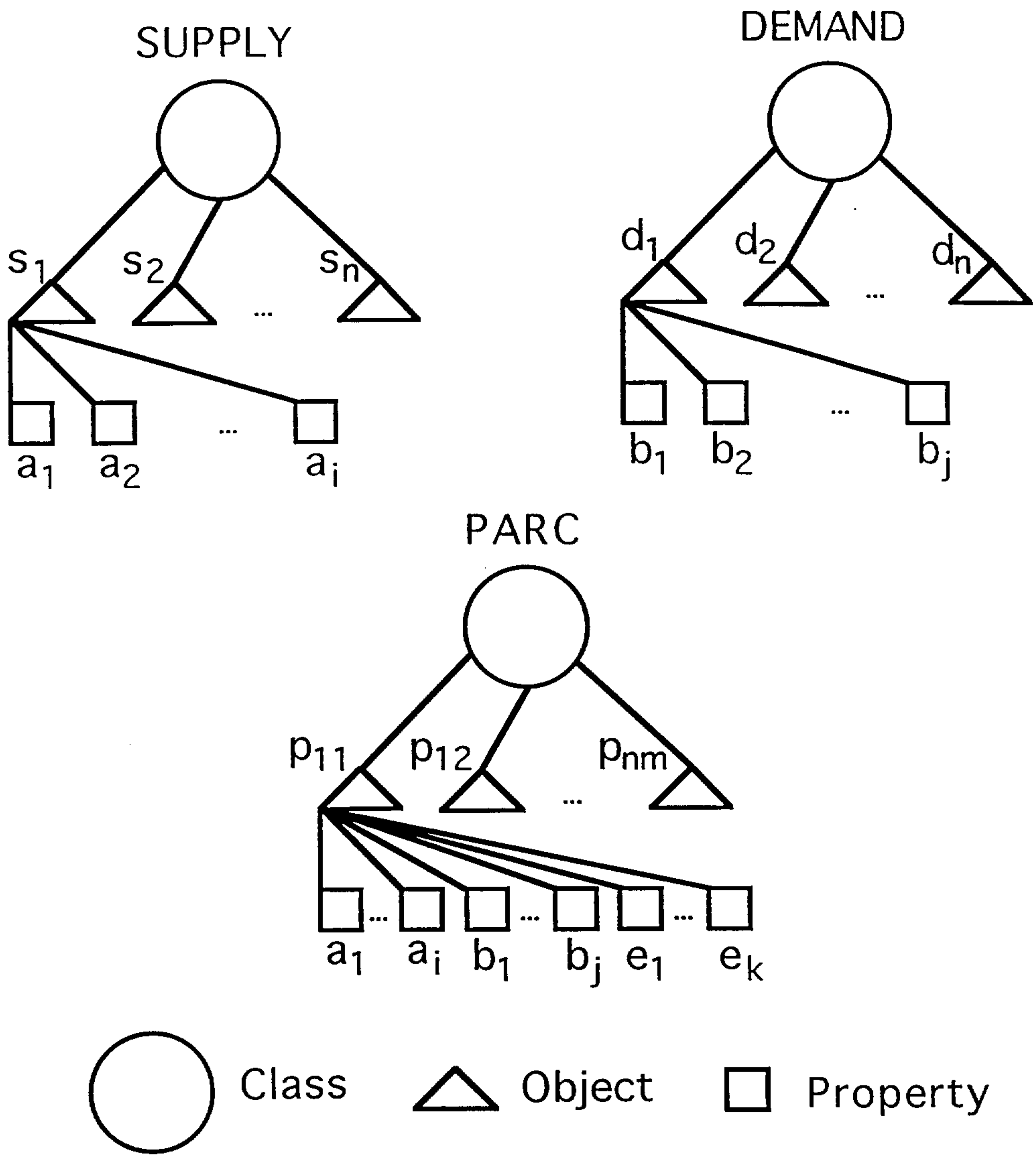


Figure 3.10 Object oriented data model in the expert system.

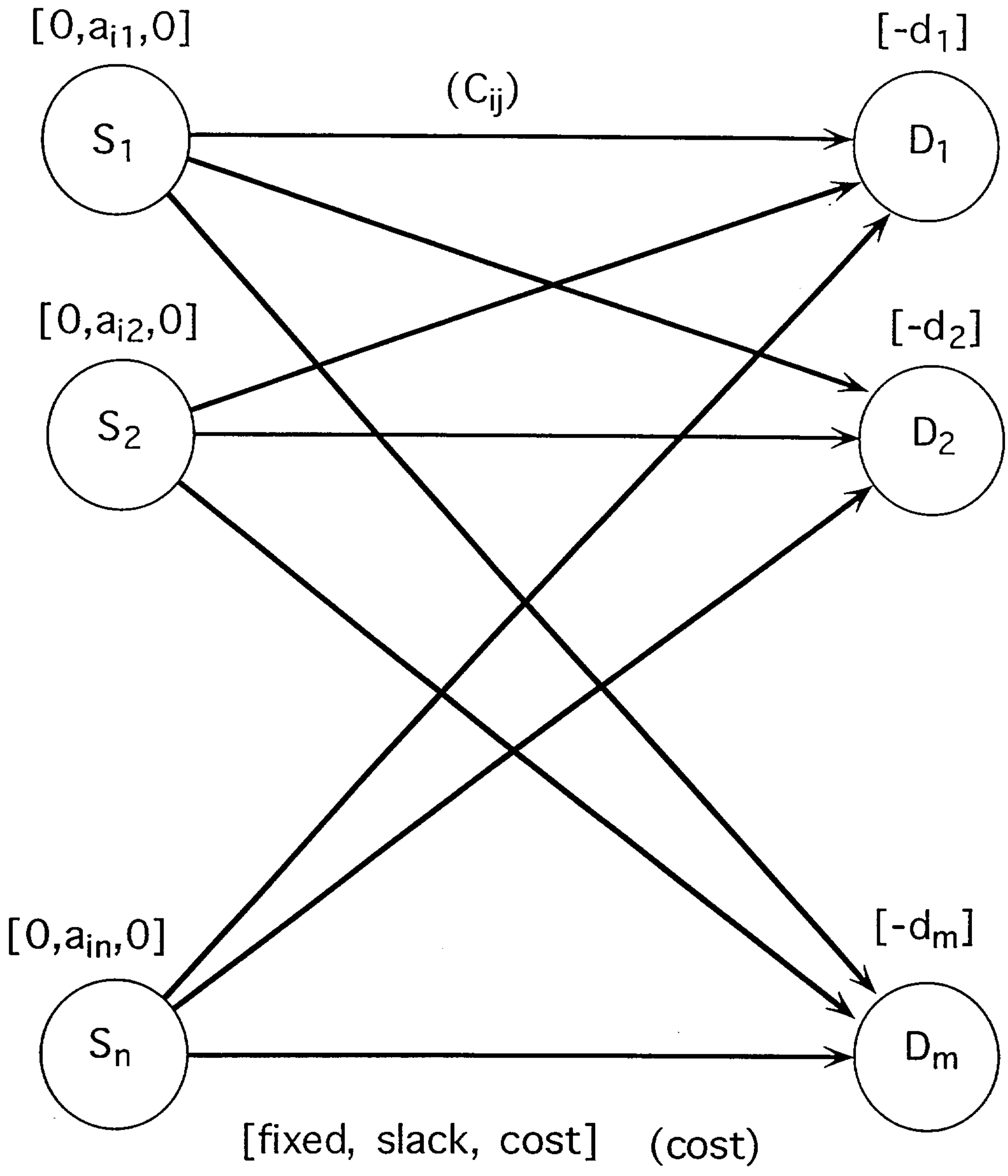


Figure 3.11 Network representation of a transportation problem.

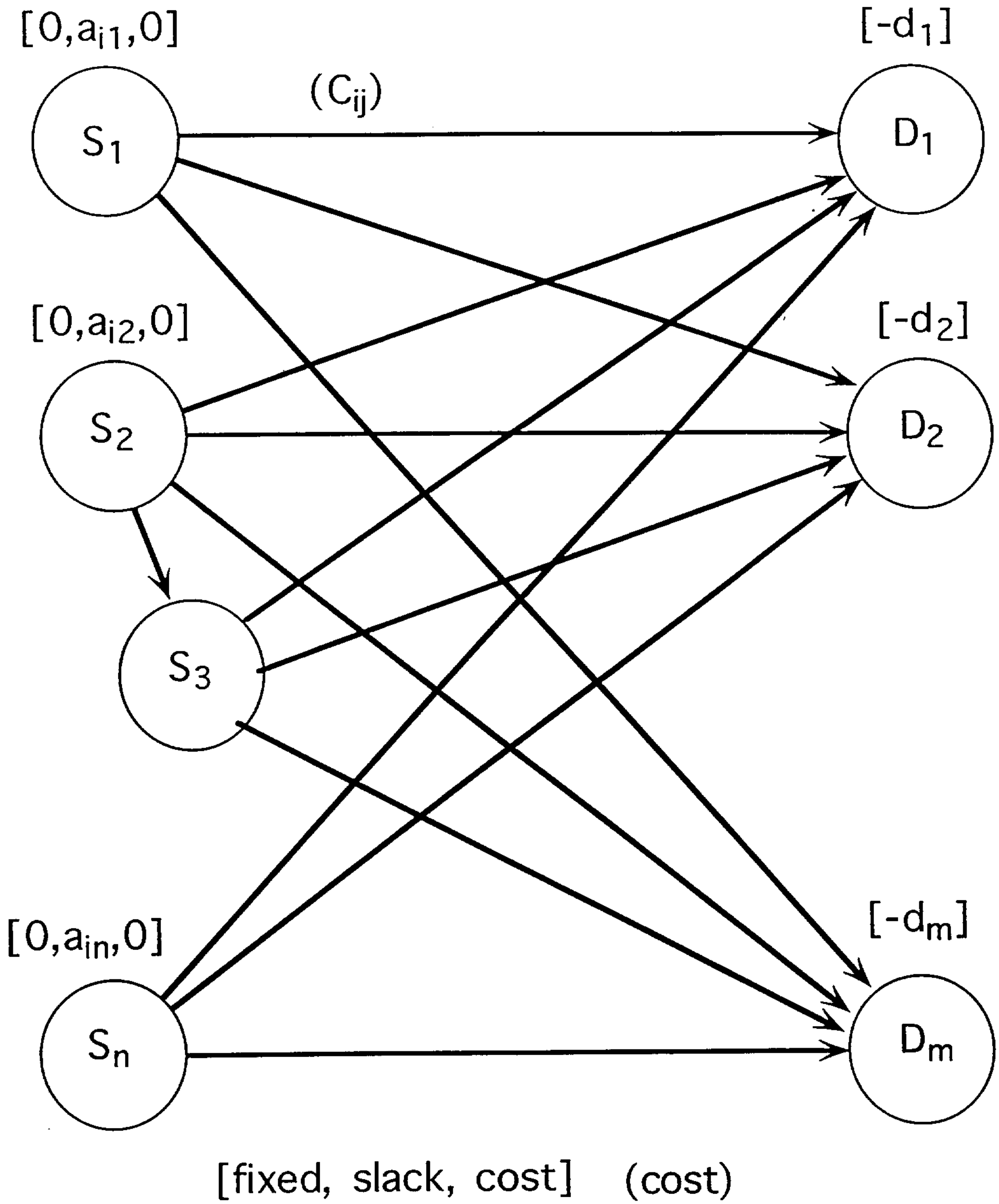


Figure 3.12 General network representation of a transshipment problem.

4.0 SOLUTION OF AN EXAMPLE PROBLEM

4.1 Introduction

From the description presented in Chapter 3 of the conceptual model underlying the planning system, it is obvious that this model is complicated, both by the nature of the planning problem itself, and by the way in which this problem is represented in the three different software systems (GIS, expert system, network flow algorithm) whose interconnected operation is used to solve the planning problem.

Recognizing these complications, Texas Water Development Board planners created a simplified example problem on which solution strategies could be attempted without inducing all the additional complexity found in real systems with a large number of supply and demand nodes. In this chapter, the solution of this problem is described, and the application of the planning system to the 19-county South Texas study region is presented in Chapter 5.

The example problem is depicted in Figure 4.1, with pertinent data listed in Table 4.1. There are four potential sources of supply (the Up river and Down river diversions, the reservoir and the well field) and three demands (small towns #1 and #2, and the large city between them). The data pertaining to these entities can be converted into the GIS coverages SUPPLY and DEMAND based upon the structures in Tables 4.2 and 4.3. A third coverage can be generated from the combination of supply and demand data. This coverage, named PARC for "potential arc," consists of the set of all possible links or arcs between supplies and demands. These are the arcs of the network flow model, and ultimately, when the network is "solved", the flow in each arc represents the allocation of water from its source to its demand. The structure of the data for the PARC coverage is shown in Table 4.4. The formulation of the GIS coverages takes the problem from the geographic representation of the problem to a functional representation as shown in Figure 3.1. At this level of representation, the different supplies and demands are clearly identified and the various possible links between them indicated.

The first nine attributes in the PARC coverage are inherited from the source and demand coverages. The lower bound is almost always set to zero. The upper bounds are initially set to the maximum amount that could ever be expected to flow in the arc, which, in most cases, is the lesser of: (1) the full capacity of the supply end of the arc; or (2) the full demand of the demand end of the arc. The unit cost for each arc represents the cost to transport a unit of flow in the arc and must be determined from an appropriate cost function. Flow is set initially to the upper bound and feasibility is set to 0, meaning that the arc is feasible. These values are decision variables in later steps of the solution procedure.

The GIS provides a great deal of flexibility for augmenting or modifying the structures and the data set. For example, if it becomes important to identify those items in the source coverage that fall within the boundaries of an aquifer recharge zone, the GIS facilitates this determination with its polygon overlay functions and the information can be incorporated into the individual records of the coverage appropriately. Similarly, the demand coverage can be easily aggregated or disaggregated with respect to demands of particular types or water quality requirements using the features of the GIS database manager. These features are not needed for solution of the example problem, however.

Once the SUPPLY, DEMAND, and PARC coverages are defined, the data is transferred to the expert system for rule processing. Through this translation process the three GIS coverages, SUPPLY, DEMAND, and PARC, are converted to classes of objects in the expert system. The expert system then invokes a set of logical rules that result in the modification of some attributes of the class PARC's objects. These rules are derived from the expertise and experience of planning professionals within the TWDB and the authors. Some of these rules will eliminate obviously unacceptable transfer links or arcs. Other rules eliminate or penalize certain arcs due to their political or environmental attributes, while still others impose the effects of water rights requirements. Some simple examples are shown in Table 4.5.

In the first example rule, all potential arcs with unit transport cost greater than 100 are eliminated from the network. In the second example, arcs are eliminated that connect demands in one water control and improvement district (WCID) to supplies in another. The third rule penalizes links that send water from west to east. The infeasibility flag is set equal to the number of the rule that caused the allocation to be considered infeasible. This information is used in post-processing to analyze the sensitivity of the resulting solution to changes in the rule set.

After applying the rules, the expert system then transfers the information back to the GIS and the coverages are updated with modified cost and infeasibility information. At that time the coverages are transferred from the GIS to the network generator to create an input data set for the network solver.

A network model is a natural tool for the abstraction of this problem. The nodes of the network represent supply or demand points in the system and the arcs represent the pipelines, canals, rivers, etc. necessary to transport water from one point to another.

Capacities and unit transport costs are incorporated into the node and arc elements and the linkages are configured to closely model transportation relationships that occur in practical operational systems. Run-of-river and demander-wholesaler relationships are easily portrayed in the network diagram. Figure 4.3 displays the network representation of the example problem defined in Figures 4.1 and 4.2. The numbers shown in the square brackets near each supply node are (1) any fixed flow that must pass through a supply, (2) the firm yield of the supply and (3) the costs in place of the supply, respectively. The numbers shown in the square brackets to the right of each demand node are the demands for that node. The numbers shown in the parentheses on each arc are the unit transportation costs along the arc.

A critical element in the success of this method is the automatic and accurate determination of the costs on all of the arcs that link the potential sources to the demands. It is, in large part, these costs that determine the resulting solution. The costs of source water are represented in the network on those links that run from the master source node to the individual source nodes. These costs are reported by the suppliers and are a part of the raw data set. More problematical are the transport costs. There are as many of these as there are potential arcs in the network; that is to say there are too many of these to deal with individually. The costs must be determined automatically as the potential arcs are set up. This is accomplished in a cost function subroutine that has been designed in a modular fashion so that it can be revised and expanded as improvements warrant. It may be that the perfect transport cost function will never be developed, but by making things modular, improvements can be incorporated into the procedure without necessitating wholesale restructuring of the data base in the GIS or the rule set in the expert system.

Initially, the cost subroutine was a very simple function of distance, thus allowing the other parts of the system to be developed and tested without having to wait on the perfection of the transport cost function. A more complex subroutine has been developed incorporating TWDB pipe cost analysis techniques [TWDB, 1967, 1977]. Unfortunately, the unit transport cost of an arc is a function of the flow in the arc, which is not known *a priori*. As a first approximation, the unit transport cost on an arc is computed assuming that the arc is flowing at full capacity. The system is then solved as a linear programming problem by the network simplex algorithm [Jensen and Barnes, 1981]. The resulting flows are then compared to the assumed flows. If a discrepancy is present, the new flows are used to update unit transport cost estimates and the system is then re-solved. This process is continued until convergence is achieved.

As an alternative to the network simplex method of solving the network flow problem, the out-of-kilter algorithm has also been coded and tested. Comparison indicates that network simplex is typically 30 percent faster than the Out-of-Kilter method.

If a feasible solution to the network flow problem exists, flows on each arc (allocations) are determined such that total system cost is minimized and all demands are met. However, some rule sets may so severely constrain the network flow problem that a feasible solution can not be found. In this case, the rule set must be examined and constraints relaxed to allow a feasible solution to be found. This process provides valuable information to the analyst regarding the nature of the rules imposed on the system and their overall effect on system design and allocations.

The final allocations and costs are transferred back to the GIS, which affords facilities to present the results in map form for easier interpretation. Also within the GIS, provision exists to compare consecutive runs thereby enabling the analyst to see the effects of changes to the rules or changes to the network itself.

4.2 RESULTS FROM SOLUTION OF THE EXAMPLE PROBLEM

The problem depicted in Figure 4.2 attempts to capture in microcosm the kinds of issues that appear in larger scale water allocation problems. In this example (1) the distances and elevations may favor one solution over another; (2) there is not enough inexpensive groundwater to serve every demand; (3) the large city is in a position to act as a water wholesaler to the smaller towns; (4) the capacity at Source S3 is dependent upon the amount taken from S2; and (5) institutional considerations such as water rights might impose constraints of their own.

The solution of this problem begins by transforming the raw data shown in Figure 4.2 into GIS SUPPLY, DEMAND, and PARC coverages. The attribute tables for these coverages are shown in Tables 4.6, 4.7, and 4.8, respectively.

The TWDB programs PIPE-D [TWDB, 1977] and PIPE-X [TWDB, 1977] have been incorporated into the system to estimate pipe diameter requirements and unit transport costs. The method relies on the pipeline end elevations, transport length, a set of empirical equations to determine a cost per thousand gallons of flow in the reach, and an assumed flow rate in the line. Then the GIS coverages are transferred to the expert system and converted into classes of objects with properties corresponding to the GIS attributes.

Next, a set of logical rules is invoked by the expert system which results in the modification of the cost and feasibility properties of selected objects in class PARC. The information is then transferred back to the GIS and the coverages are modified appropriately.

A network generator program translates the GIS SUPPLY, DEMAND, and PARC coverage information into an ASCII input data file structured according to the input requirements of the network simplex program. Then the solver is run. If a network flow solution exists, an ASCII output file containing the resulting solution is written and then used to update the GIS PARC coverage the results are displayed.

The assumed values of flow are examined and if necessary the problem is restarted with adjusted initial flows and their associated unit costs.

4.2.1 Rule Set No. 1 - No Rules

The results of the first case, where no rules are imposed on the system and all allocation arcs are included as feasible in the network flow solver, are shown in Table 4.9 and Figure 4.4. Water is allocated from supply S1 to demand D1, from S2 and S4 to D2, and from S2 to D3. The total system cost to meet all demands is $239,900.

4.2.2 Rule Set No. 2 - Distance Rules

The results of the next case, where a distance rule is applied, are shown in Table 4.10 and Figure 4.5, the expert system applies a rule that eliminates arcs with unit cost greater than $100 per AF. Although this removes two arcs from the analysis, the allocations are unchanged from the previous case. Again the total cost is $239,900.

4.2.3 Rule Set No. 3 - Distance and WCID Rules

In the third case, no water is allowed to be transferred out of a water control and improvement district number 100. The results of this solution are shown in Table 4.11 and Figure 4.6. This forces the inexpensive groundwater (source S4) to be allocated to the small town (demand D1). Other allocations adjust as necessary to meet demands and the resulting total cost is $261,000.

4.2.4 Rule Set No. 4 - Penalized Distance and WCID Rules

The final case, the results of which are shown in Table 4.12 and Figure 4.7, demonstrates the effect of a adding a rule that introduces a cost reduction on arcs that transfer water from a wholesaler (city D2) to another demand location (demand locations D1 or D3). Total cost is $256,500. Note that the inexpensive groundwater from source S4 goes unallocated.

Although much more is needed with respect to capturing the knowledge and experience of experts, these brief examples demonstrate that the pieces of the system can be made to work in concert in such a way as to arrive at solutions similar to those of a professional analyst.

Table 4.1 Example Problem Data

Water Entity	Symbol	Supply (AF)	Demand (AF)	Cost in place ($/AF)	Elevation (ft AMSL)
Up river	S_1	500	–	10	1100
Reservoir	S_2	3500	–	20	800
Down river	S_3	$3500-\sum_j X_{2j}$	–	25	750
Well Field	S_4	1100	–	1	1100
Small town #1	D_1	–	500	–	1000
Large city	D_2	–	2000	–	850
Small town #2	D_3	–	1000	–	700

Table 4.2 SUPPLY Coverage Attributes

Description	Symbol
Identification Number	SUPSRC_ID
Transfer Flag	XSRC
Water District Id #	WCIDS
Latitude	LATS
Longitude	LONS
Elevation	ELEVS
Capacity	EXTQS
Cost in place	CIPL
Placename	NAMES

Table 4.3 DEMAND Coverage Attributes

Description	Symbol
Identification Number	DEMAND_ID
Transfer Flag	XDEM
Water District Id #	WCIDD
Latitude	LATD
Longitude	LOND
Elevation	ELEVD
Capacity	EXTQD
Cost in place	CIPL
Placename	NAMED

Table 4.4 PARC Coverage Attributes

Description	Symbol
Identification Number	PARC_ID
Source latitude	LATS
Source longitude	LONS
Source elevation	ELEVS
Source water district id	WCIDS
Demand latitude	LATD
Demand longitude	LOND
Demand elevation	ELEVD
Demand water district id	WCIDD
Lower bound	LOWB
Upper bound	UPPB
Unit transport cost	COST
Amount of flow	FLOW
Infeasibility	NFEAS

Table 4.5 Example Expert System Rules

Rule	Conditions	Hypothesis	Action
1.	If PARC object COST > 100	Allocation too expensive	Set NFEAS = 1
2.	IF PARC object WCIDS ≠ WCIDD	Transfer of water outside district	Set NFEAS = 2
3.	If PARC object LONS > LOND	Transfer of water from west to east	Set NFEAS = 3

Table 4.6 Example Problem SUPPLY Coverage

SUPSRC_ID	XSRC	WCIDS	LATS	LONS	ELEVS	EXTQS	CIPL	NS
1	0	800	30.10	99.06	1100.	500	10	S1
2	0	700	30.05	99.06	800.	3500	20	S2
3	2	700	30.00	99.06	750.	0	25	S3
4	0	100	30.09	99.02	900.	1100	1	S4

Table 4.7 Example Problem DEMAND Coverage

SUPSRC_ID	XDEM	WCIDD	LATD	LOND	ELEVD	EXTQD	ND
1	0	100	30.10	99.00	1000.	500	D1
2	1	200	30.05	99.00	850.	2000	D2
3	0	300	30.00	99.00	700	1000	D3

Table 4.8 Example Problem PARC Coverage

ID #	Lower Bound	Upper Bound	Cost	Flow	Infeasibility
S001D001	0	500	79	500	0
S001D002	0	500	45	500	0
S001D003	0	500	60	500	0
S002D001	0	500	101	500	0
S002D002	0	3500	52	3500	0
S002D003	0	1000	60	1000	0
S003D001	0	500	107	500	0
S003D002	0	3500	58	3500	0
S003D003	0	1000	60	1000	0
S004D001	0	500	90	500	0
S004D002	0	1100	45	1100	0
S004D003	0	1000	60	1000	0
D002D001	0	500	95	500	0
D002D003	0	1000	60	1000	0

* Columns for LATS, LONS, ELEVS, WCIDS, LATD, LOND, ELEVD, and WCIDD are not shown.

Table 4.9 Example Problem Updated PARC Coverage - Rule Set 1

ID #	Lower Bound	Upper Bound	Cost ($/AF)	Flow (AF)	Feasibility
S001D001	0	500	79	500	0
S001D002	0	500	45	0	0
S001D003	0	500	60	0	0
S002D001	0	500	101	0	0
S002D002	0	3500	52	900	0
S002D003	0	1000	60	1000	0
S003D001	0	500	107	0	0
S003D002	0	3500	58	0	0
S003D003	0	1000	60	0	0
S004D001	0	500	90	0	0
S004D002	0	1100	45	1100	0
S004D003	0	1000	60	0	0
D002D001	0	500	95	0	0
D002D003	0	1000	60	0	0

Table 4.10 Example Problem Updated PARC Coverage - Rule Set 2

ID #	Lower Bound	Upper Bound	Cost ($/AF)	Flow (AF)	Feasibility
S001D001	0	500	79	500	0
S001D002	0	500	45	0	0
S001D003	0	500	60	0	0
S002D001	0	500	101	–	2
S002D002	0	3500	52	900	0
S002D003	0	1000	60	1000	0
S003D001	0	500	107	–	2
S003D002	0	3500	58	0	0
S003D003	0	1000	60	0	0
S004D001	0	500	90	0	0
S004D002	0	1100	45	1100	0
S004D003	0	1000	60	0	0
D002D001	0	500	95	0	0
D002D003	0	1000	60	0	0

Table 4.11 Example Problem Updated PARC Coverage - Rule Set 3

ID #	Lower Bound	Upper Bound	Cost ($/AF)	Flow (AF)	Infeasibility
S001D001	0	500	79	0	0
S001D002	0	500	45	500	0
S001D003	0	500	60	0	0
S002D001	0	500	101	–	2
S002D002	0	3500	52	1500	0
S002D003	0	1000	60	1000	0
S003D001	0	500	107	–	2
S003D002	0	3500	58	0	0
S003D003	0	1000	60	0	0
S004D001	0	500	90	500	0
S004D002	0	1100	45	–	3
S004D003	0	1000	60	–	3
D002D001	0	500	95	0	0
D002D003	0	1000	60	0	0

Table 4.12 Example Problem Updated PARC Coverage - Rule Set 4

ID #	Lower Bound	Upper Bound	Cost ($/AF)	Flow (AF)	Infeasibility
S001D001	0	500	79	0	0
S001D002	0	500	45	500	0
S001D003	0	500	60	0	0
S002D001	0	500	101	–	2
S002D002	0	3500	52	2000	0
S002D003	0	1000	60	1000	0
S003D001	0	500	107	–	2
S003D002	0	3500	58	0	0
S003D003	0	1000	60	0	0
S004D001	0	500	90	0	0
S004D002	0	1100	45	–	3
S004D003	0	1000	60	–	3
D002D001	0	500	**10**	500	0
D002D003	0	1000	**10**	0	0

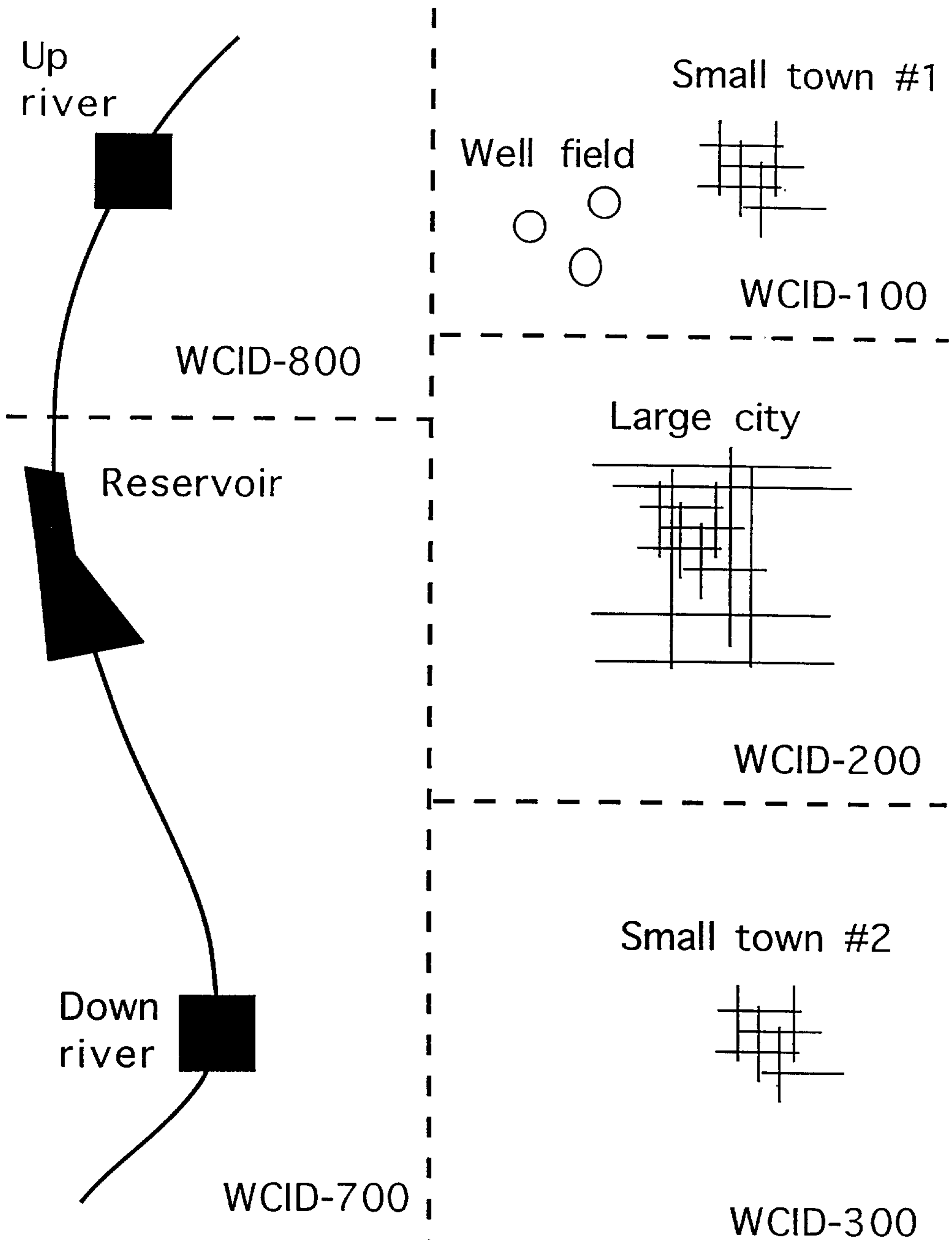


Figure 4.1 Geographic representation of example problem.

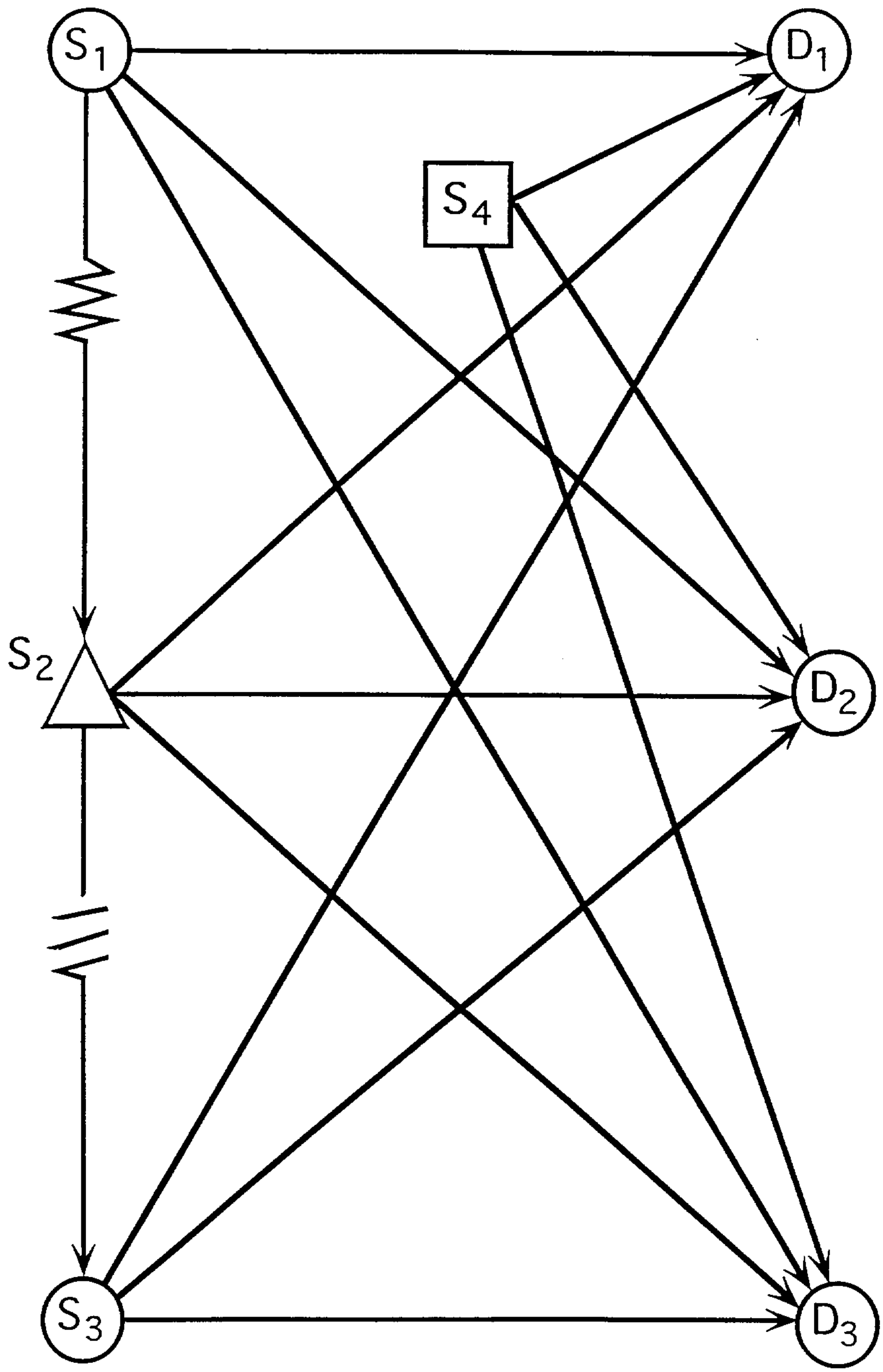


Figure 4.2 Functional representation of example problem.

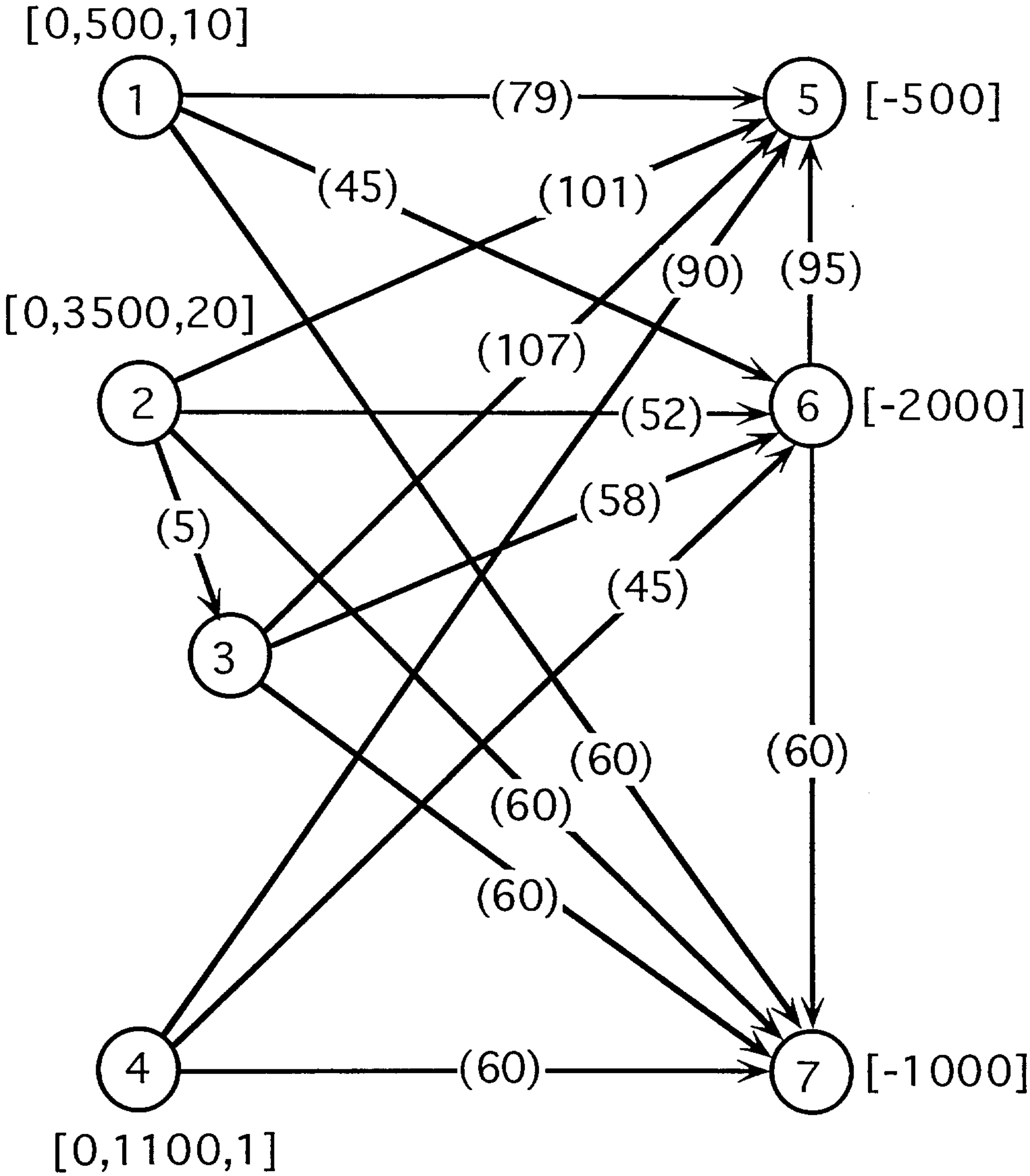


Figure 4.3 Planning representation of example problem.

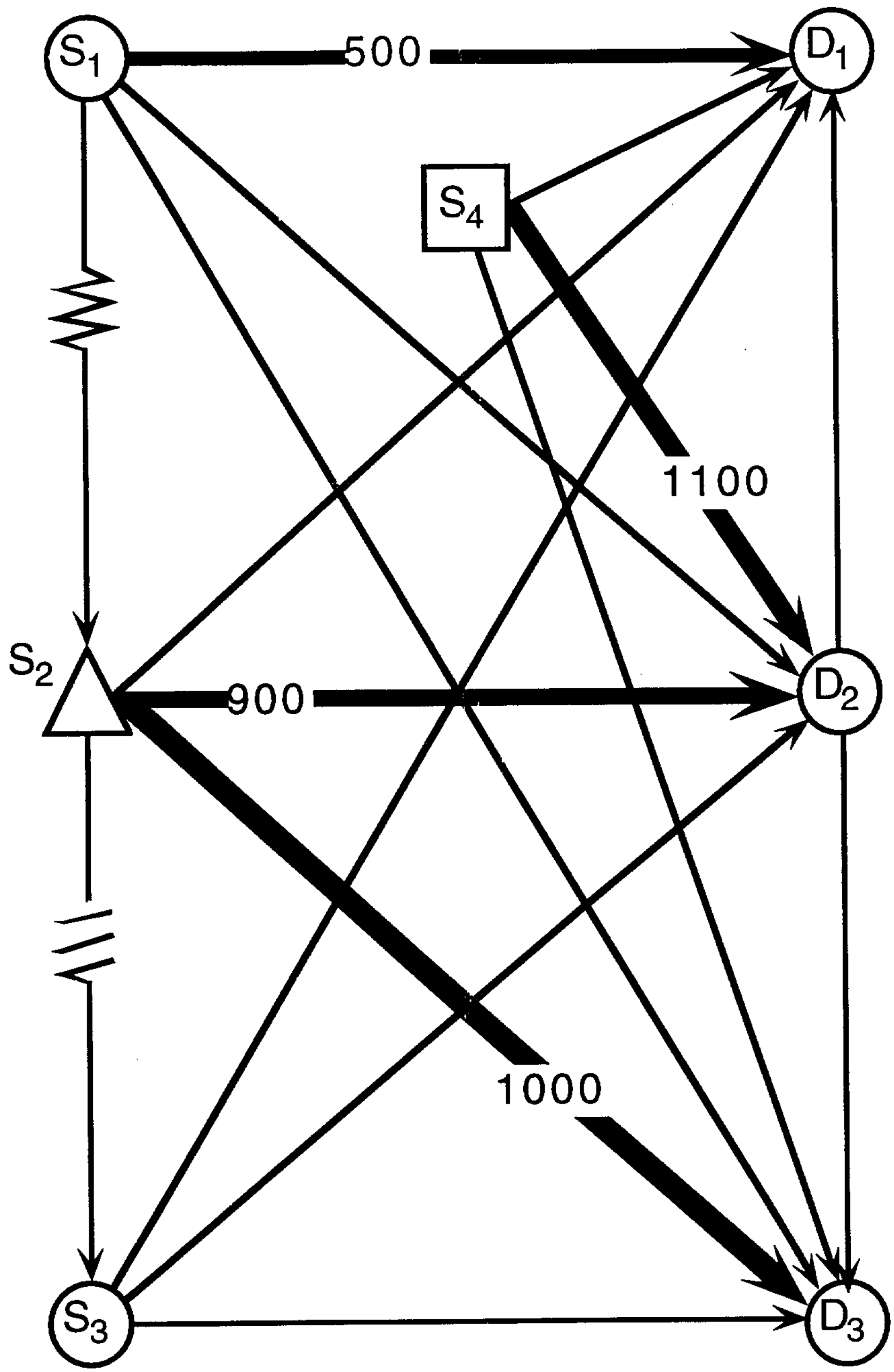


Figure 4.4 Solution of example problem - Rule set 1.

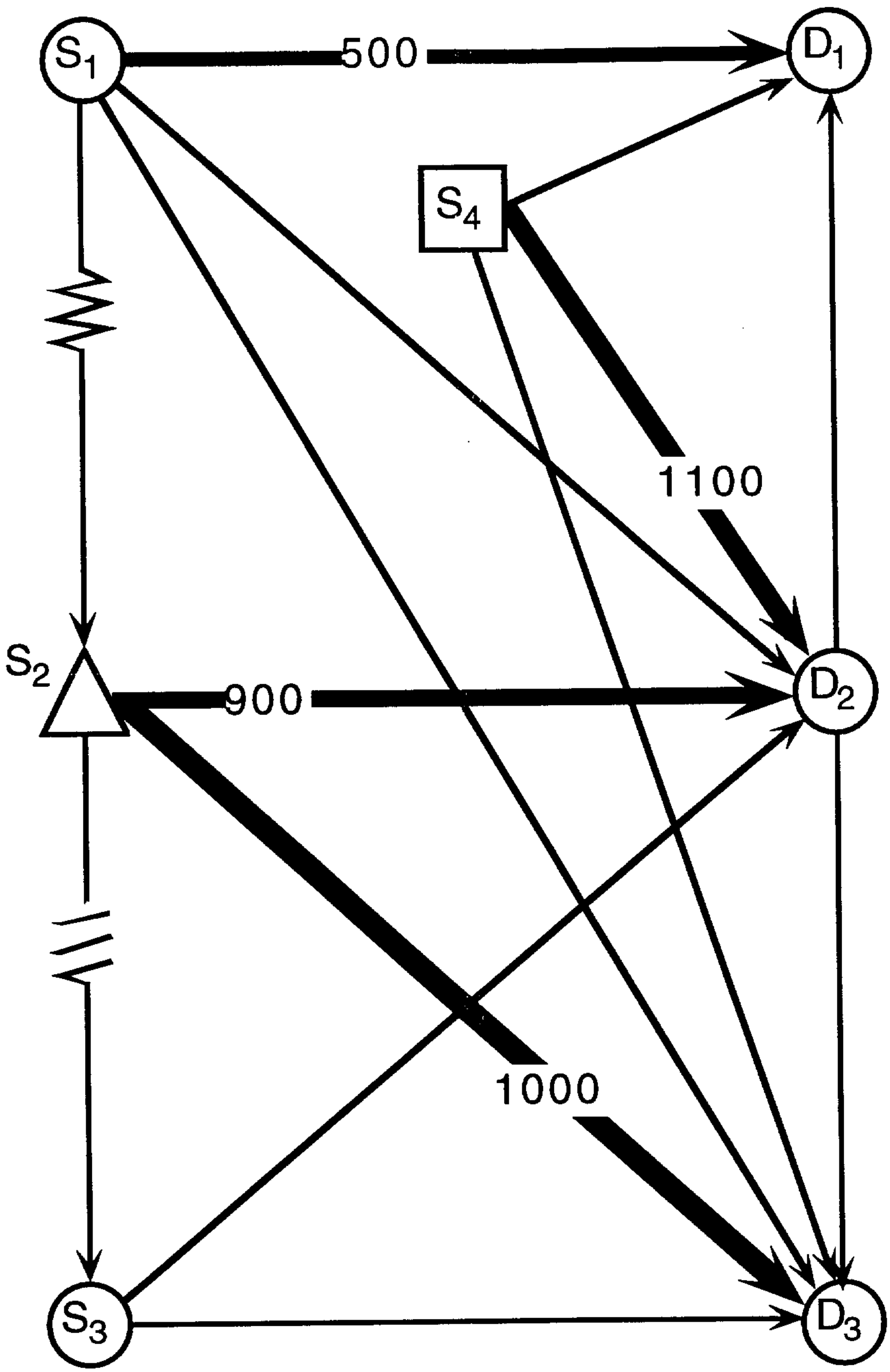


Figure 4.5 Solution of example problem - Rule set 2.

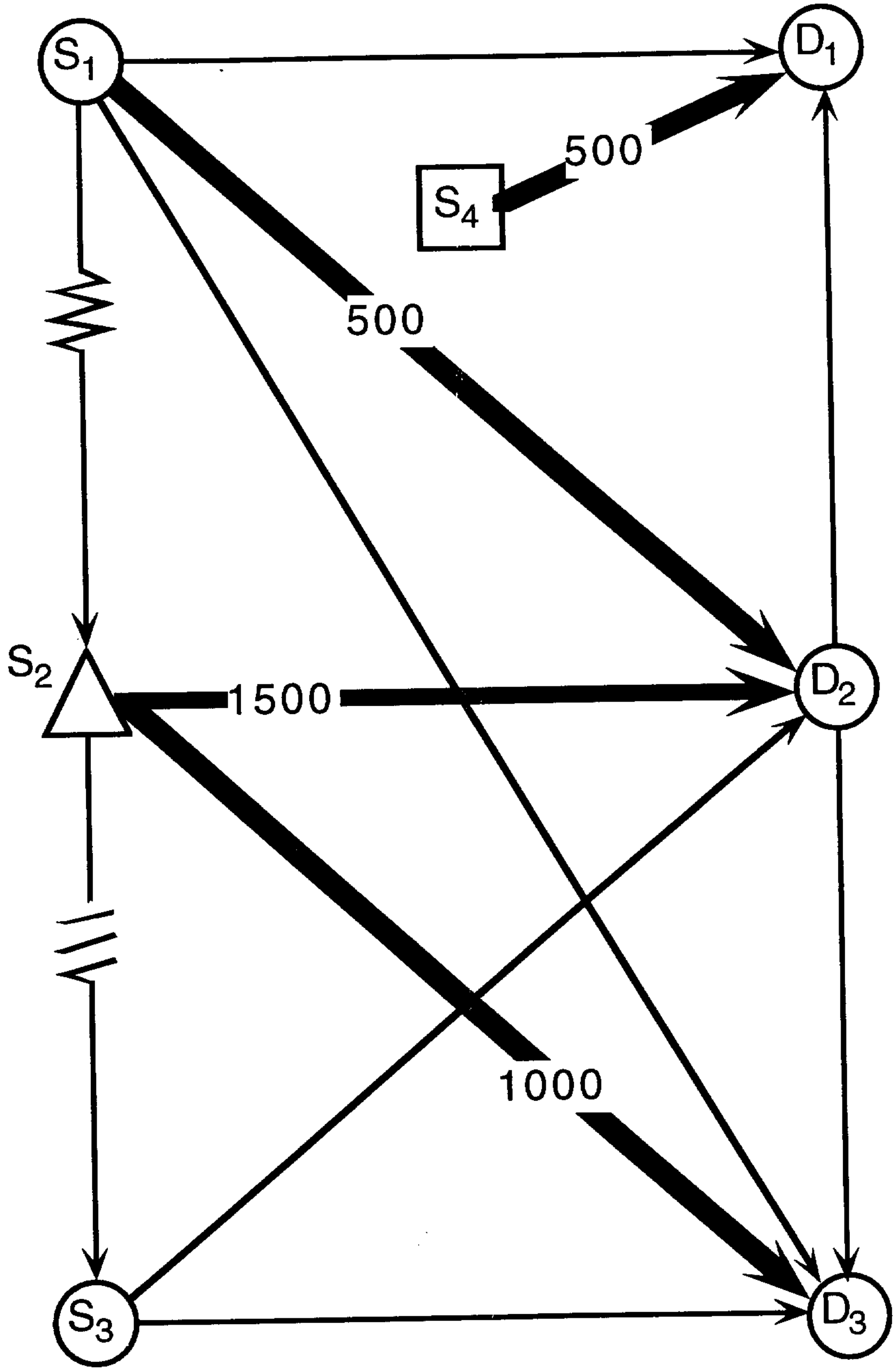


Figure 4.6 Solution of example problem - Rule set 3.

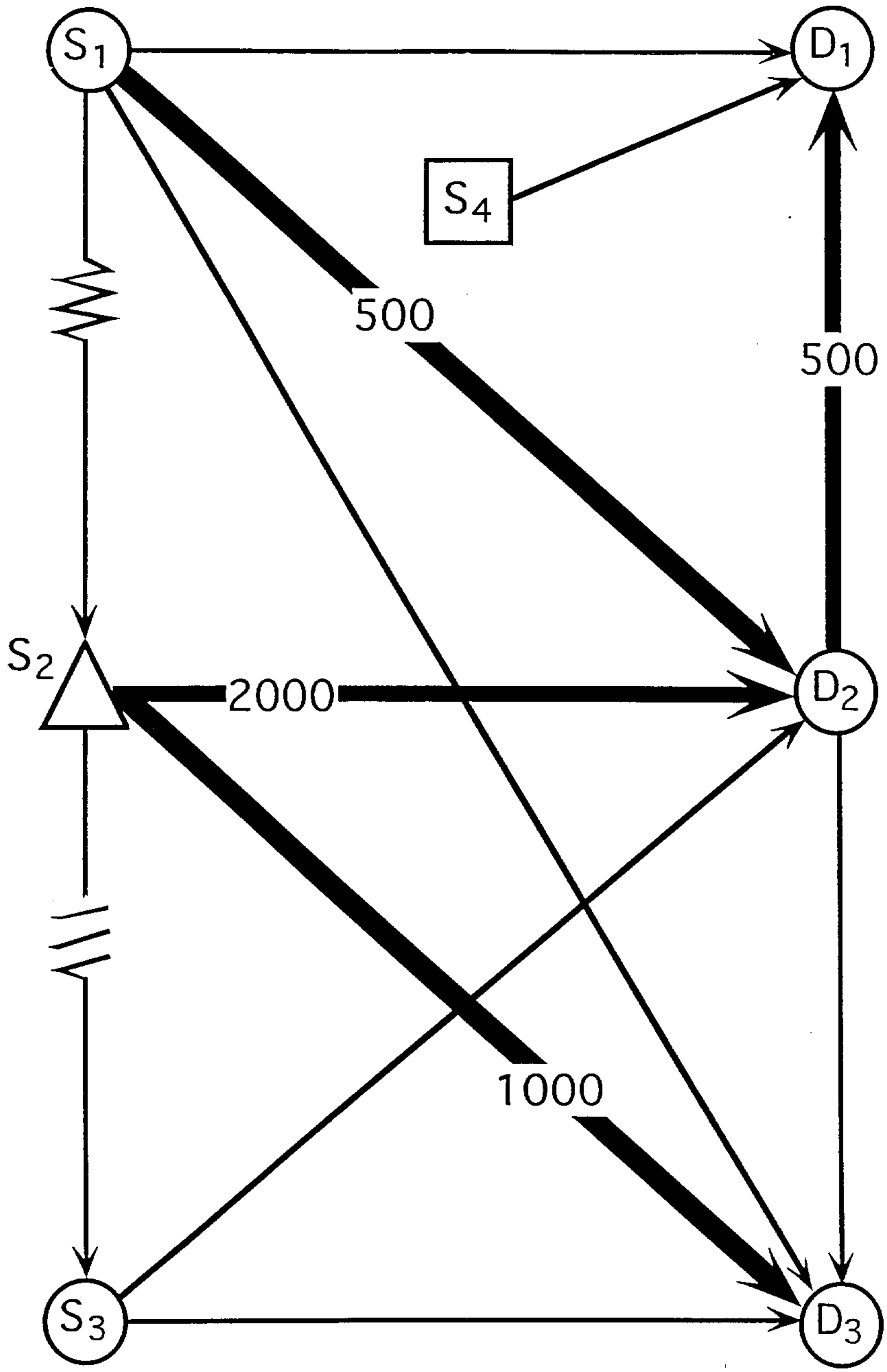


Figure 4.7 Solution of example problem - Rule set 4.

5.0 CASE STUDY - CORPUS CHRISTI, TEXAS REGION

5.1 INTRODUCTION

The fifteen separate water planning regions considered by the Texas Water Development Board in the 1992 update of the Texas Water Plan (TWDB, 1992, Fig 2-1, p. 8) are depicted in Figure 5.1. These conform to uniform service regions for state government regulatory and services purposes as required by the General Appropriations Act of the 72nd Legislature. Region 10, the Coastal Bend, is located in the central coastal plains and is comprised of the following 19 counties: Aransas, Bee, Brooks, Calhoun, DeWitt, Duval, Goliad, Gonzales, Jackson, Jim Wells, Kenedy, Kleberg, Lavaca, Live Oak, McMullen, Nueces, Refugio, San Patricio, and Victoria.

The physiography of the region is generally flat, with grassland in the humid northeast tending to brush country in the subhumid southwest. Rainfall ranges from 40 inches per year in the eastern counties to 24 inches per year in the western counties [Larkin and Bomar, 1983]. Surface water supplies exist in the regulated rivers which flow from the northwest to southeast across the region emptying into the Gulf of Mexico. These rivers include the Navidad and Lavaca in the Lavaca basin, the San Antonio, San Marcos and Guadalupe in the Guadalupe basin, and the Medina, Frio and Nueces in the Nueces basin. In addition, numerous creeks rise within the region and are exploited by agricultural users as local supplies. Surface supplies in the region are drawn directly from the rivers and are also developed in several moderate sized reservoirs, the most significant being Choke Canyon Reservoir and Lake Corpus Christi on the Nueces River. Surface water resources in Region 10 are estimated to contribute a firm yield exceeding 474,000 acre-ft per year in the year 2040.

The region also has extensive groundwater supplies. The Gulf Coastal Aquifer provides good quality and relatively inexpensive domestic water to small cities and towns in every county of the planning region. Moreover, water supplies from the Carrizo-Wilcox, Sparta Sand, and Queen City Aquifers are available to the more inland counties. Typically, ground water is one-half to one-fourth the cost of surface water within this region, and consequently, it is the demanders' first choice of supply. Groundwater resources are expected to exceed 310,000 acre-ft/year in the year 2040.

The potential sources of water supply in Region 10 for the year 2040 are listed in Table 5.1 (A more detailed list of the data appears in the Appendix). These data were extracted from information provided by the TWDB in data files: ALLOC40, RESDATA, and GWDATA [Steve Densmore, TWDB, personal commun., 1992]. Costs for groundwater were developed from information provided by TWDB staff concerning average well depths and drilling costs per foot for each aquifer in the region. Costs for the surface water for each reservoir were estimated by amortizing the costs of construction, adding the costs of operation and dividing by the firm yield. These costs of source supplies do not reflect the same degree of detailed investigation that is contained in the other data. The category 'SHORTAGE' that appears for seven counties in the table is a device employed by TWDB planners that allows flexibilities with respect to satisfying certain, local demands. As indicated in the table, there is a large supply (999,999 AF/Y) of 'shortage water' but it is available at a very high price (999 $/AF). This is the last water allocated to any purpose and prevents infeasibility in the model while identifying those demands that are, in fact, not really being met. Another modelling abstraction occurs with respect to the location of supplies that are, in the real world, widely dispersed: groundwater, shortage, and local supplies are collapsed into point entities and placed at the centroid of their respective counties.

Although 45 cities and towns are identified as having population above 1000, the city of Corpus Christi is the only large metropolitan center in the region. The present (1990) population of the region is 707,791 and is expected to rise to 1,297,523 by the year 2040. This will to result in a domestic demand of 238,283 acre-ft. per year. Farming, ranching, oil production, refining, and metals manufacturing add significantly, increasing the total demand for water in the year 2040 to 712,839 acre-ft per year. Table 5.2 lists the demand data for the year 2040, identifying 98 water demanders within the region.

As before, the data were provided by the TWDB. The particular file, NTR11, lists population estimates and water demand projections for several categories of use by cities, towns and counties in Texas. For the purpose of this investigation, three categories other than cities were created by lumping the appropriate categories found in the raw data. These are: county aggregated industrial, county aggregated agricultural, and county aggregated other domestic (not otherwise accounted for as a specified city), and these appear in the table as CA_county, CB_county, and CO_county. Such demands are distributed in the real world but are collapsed to point entities and assumed to be located at the centroid of their respective counties for modelling purposes.

Given the 44 potential sources of supply and 98 demanders, how should the water resources of this region be allocated? What other considerations, besides geography and economy, apply? Ultimately, can a list of rules be devised and implemented to capture the heuristic aspects of the planning process? The following sections outline one attempt to answer these questions.

5.2 PLAN 0 - NO RULES

This plan serves as a baseline for comparison. There were no modifications made to the initial network of potential arcs by the expert system prior to LP optimization. The network consisted of 143 nodes and 4356 arcs. The LP solution was completed in 11 seconds running on a SUN Sparc 2 Workstation. The solution puts flow on 130 arcs at a total cost of $40,000,000 and is displayed in Figure 5.2. More specific detail is provided in Table 5.3.

The resulting distribution of the region's water supplies is generally reasonable. There are, however, some allocations that do not fit present planning criteria. Many cases appear where large industrial and agricultural users obtain groundwater resources that traditionally would have been allocated to smaller municipalities. Moreover, there are cases where groundwater demanders import groundwater from another county in preference to using groundwater available in their own county. Also, there are instances where local and shortage supplies are utilized beyond county boundaries and by non-agricultural users. This has the consequent effect of forcing the intended and appropriate users of these supplies to be allocated water from somewhere else.

5.3 PLAN 4 - STATEWIDE RULES

In this plan basic rules are applied in the expert system that modify the network of potential arcs. These rules rectify modeling problems associated with locating aggregated supplies at the centroid of counties and implement basic statewide planning concepts. The network was pared of 1030 infeasible arcs and transportation costs were reset to zero on 70 others. The LP solution was completed in 7 seconds. The allocation employs 129 arcs at a total cost of $38,000,000 and is displayed in Figure 5.3. Detail is provided in Table 5.4. The following rules were applied in this scenario:

(1) Within county groundwater transport costs reset to zero. In the model, groundwater is located at the centroid of each county, resulting in a transport cost that does not actually exist.

(2) Within county local supply transport costs set to zero. As above, actual users do not need to transport this water any significant distance.

(3) Allocation of 'Local' supply to non-agricultural users infeasible. By definition, this water is only available to agricultural demanders.

(4) Local supply export infeasible. Again, by its definition, this source is not available outside of the county to which it is accounted.

(5) Shortage supply export infeasible. As above, this supply is local by defintion.

The allocations that evolve from implementing these rules exhibit the expected effects and represent a substantial improvement over the previous allocations; however, there remain general patterns of allocations that do not conform to present practice and philosophy. For example, given the types and scale of agricultural operations in this region, it is uncommon for their water supplies to be drawn from outside the users' county. Similarly, in practice in this region, industrial users do not import groundwater from other counties (precluding its use by the smaller cities and towns in the distant source counties and consequently forcing those communities to more expensive alternatives). These atypical patterns can be easily removed by including rules that prohibit such allocations. Furthermore, there are several instances where large demanders tap an entire groundwater supply dry. While this may be the optimum allocation of resources, there is a political consideration that goes beyond it in real life: no single demander is allowed to consume all of the cheapest water. Again, a rule or sequence of rules can be written to prevent such allocations. Last, there is a concern for the very small demanders. Several allocations to very small cities call for distant transport to meet these demands, when in reality, there is no economic possibility for such a transfer. Rules must be included to reserve nearby, inexpensive water supplies for these users. The rules to implement these corrections are discussed in the next section.

5.4 PLAN 9 - STATEWIDE AND REGIONAL RULES

This plan incorporates the planning considerations specific to this region. It also applies rules to specific supply sources and demanders. All previous rules apply plus the following:

(6) Groundwater import by industrial users infeasible. Prevents the dislocation of inexpensive supplies.

(7) Groundwater import by agricultural users infeasible. Agricultural users must rely on other local or shortage supplies.

(8) County Other Municipal Users dedicated up to 20% of the preferred (cheapest) source of supply. Reserves a portion of the Gulf Coast Aquifer to small demanders.

(9) Individual groundwater users limited to 40% of supply. Forces distribution of cheap supplies among several users.

(10) Twenty percent surcharge on cost in place applied to groundwater exports. Discourages groundwater transport.

(11) Surface water import by agricultural users infeasible. Dispersed, small and medium demanders cannot support this type of infrastructure.

(12) Coletto Creek to municipal and agricultural users infeasible. This reservoir is dedicated to industrial use.

(13) Jackson County agricultural users not restricted by rule 9. Jackson County agricultural demands require special consideration.

(14) Tilden to Gulf Coast, Queen City and Sparta Sand infeasible. This city is not located over these aquifers.

The network was pared of 2098 infeasible arcs and costs were modified on 2586 arcs. The lower bound was reset on 19 arcs to implement rule 8, and the upper bound was reset on 2619 arcs to implement rule 9. LP run time was 4 seconds and 147 arcs were assigned flow at a total cost of $53,000,000. The resulting allocations are displayed in Figure 5.4 and detailed in Table 5.5.

The allocations derived from this plan closely resemble the present Texas Water Plan devised by TWDB experts. The remaining difference involves the underutilization of water from surface supplies, particularly from Choke-Corpus Reservoir. In the 1990 Water Plan [TWDB,1992] this water is allocated to medium-sized and large municipal users in the nearby counties. Additional rules that favor such connections were needed to more closely match this kind of allocation logic. The results of applying these rules is discussed in the next section.

5.5 PLAN 10 - STATEWIDE, REGIONAL, AND PARC RULES

This plan further refines the planning considerations to include specifications of the attributes at the arc level. Again, all previous rules apply as well as these additional:

(15) Coletto Creek to CA_Goliad transport cost reset to zero. Corrects modeling error.

(16) Choke-Corpus to Aransas Co. municipalities discount applied. Encourages municipal use of this supply source.

(17) Choke-Corpus to Bee Co. municipalities discount applied.

(18) Choke-Corpus to Jim Wells Co. municipalities discount applied.

(19) Choke-Corpus to Kleberg Co. municipalities discount applied.

(20) Choke-Corpus to Live Oak Co. municipalities discount applied.

(21) Choke-Corpus to Nueces Co. municipalities discount applied.

(22) Choke-Corpus to San Patricio Co. municipalities discount applied.

After applying the rules, 28 additional arcs were modified to implement the cost reductions. The LP solver indicated an optimum feasible solution in 4 seconds involving 145 arcs for a total cost of $50,000,000. Figure 5.5 and Table 5.6 display the results. The desired effect is evident in the allocations from the largest surface supply in the region, Choke-Corpus with bay. Under this scenario, several of the cities in the targeted counties are receiving water from the Choke-Corpus reservoir system, and the resource is now fully utilized. The industrial demanders who were formerly allocated water experience reductions from this source and have been connected to other surface sources of supply.

5.6 ASSESSMENT OF THE COASTAL BEND REGION CASE STUDY

A detailed comparison of the allocations made by the Automated Allocations System (AAS) and the 1990 Texas Water Plan is presented in Table 5.7. Many individual discrepancies can be found, yet overall the plans are quite similar. The instances where run of river supplies are allocated from supply Guadalupe in preference to supply San Antonio are balanced later by allocations from supply San Antonio in preference to supply Guadalupe. This is a caused by the equal cost of each supply and the proximate locations designated for them. If finer definition in these allocation recommendations is desired, it may be achieved with the designation of several run of river supply locations for each. Other individual discrepancies may be addressed by writing additional rules at the Parc level, but such an effort, while it will improve the agreement between AAS and TWDB, is a move away from the goal of an automated system.

Table 5.1 Potential Supply Sources for Region 10

ID#	NAME	YIELD (AF/Y)	COST ($/AF)
1	GULF_COAST__ARANSAS	400	31
2	CARRIZO-WILCOX__BEE	394	24
3	GULF_COAST__BEE	14577	31
4	GULF_COAST__BROOKS	14577	31
5	GULF_COAST__CALHOUN	2940	31
6	LOCAL__CALHOUN	12600	10
7	SHORTAGE__CALHOUN	999999	999
8	GULF_COAST__DE_WITT	15866	31
9	GULF_COAST__DUVAL	23970	31
10	GULF_COAST__GOLIAD	12809	31
11	CARRIZO-WILCOX__GONZALES	19840	16
12	GULF_COAST__GONZALES	2083	39
13	QUEEN_CITY__GONZALES	6104	44
14	SPARTA_SAND__GONZALES	16340	49
15	LOCAL__GONZALES	4200	10
16	GULF_COAST__JACKSON	28343	39
17	SHORTAGE__JACKSON	999999	999
18	GULF_COAST__JIM_WELLS	11370	31
19	SHORTAGE__JIM_WELLS	999999	999
20	GULF_COAST__KENEDY	9550	31
21	GULF_COAST__KLEBERG	17088	31
22	GULF_COAST__LAVACA	38123	39
23	CARRIZO-WILCOX__LIVE_OAK	2399	24
24	GULF_COAST__LIVE_OAK	5242	31
25	LOCAL__LIVE_OAK	760	10
26	SHORTAGE__LIVE_OAK	999999	999
27	CARRIZO-WILCOX__MCMULLEN	7909	24
28	GULF_COAST__MCMULLEN	1838	31
29	QUEEN_CITY__MCMULLEN	1105	44
30	SPARTA_SAND__MCMULLEN	600	49
31	SHORTAGE__MCMULLEN	999999	999
32	GULF_COAST__NUECES	3254	31
33	LOCAL__NUECES	950	10
34	SHORTAGE__NUECES	999999	999
35	GULF_COAST__REFUGIO	7768	31
36	GULF_COAST__SAN_PATRICIO	5228	31
37	SHORTAGE__SAN_PATRICIO	999999	999
38	GULF_COAST__VICTORIA	41130	39
39	TEXANA	75000	43
40	CUERO_I&II	52000	49
41	GUADALUPE_RIVER	79000	40
42	COLETO_CREEK	12500	59
43	SAN_ANTONIO_RIVER	25000	40
44	CHOKE-CORPUS_w_bay	230549	43

Table 5.2 Region 10 Demand Data for the Year 2040

ID#	NAME	DEMAND (AF/Y)	ID#	NAME	DEMAND (AF/Y)
1	ROCKPORT	2324	51	CO_KENEDY	49
2	CO_ARANSAS	4426	52	CB_KENEDY	1821
3	CA_ARANSAS	554	53	KINGSVILLE	9179
4	CB_ARANSAS	107	54	CO_KLEBERG	2028
5	BEEVILLE	3730	55	CA_KLEBERG	2574
6	CO_BEE	2921	56	CB_KLEBERG	2612
7	CA_BEE	5	57	HALLETTSVILLE	831
8	CB_BEE	2590	58	SHINER	746
9	FALFURRIAS	1372	59	YOAKUM	915
10	CO_BROOKS	769	60	CO_LAVACA	1865
11	CA_BROOKS	18	61	CA_LAVACA	6933
12	CB_BROOKS	1690	62	CB_LAVACA	15216
13	POINT_COMFORT	237	63	GEORGE_WEST	592
14	PORT_LAVACA	3213	64	THREE_RIVERS	485
15	SEADRIFT	398	65	CO_LIVE_OAK	795
16	CO_CALHOUN	2219	66	CA_LIVE_OAK	16113
17	CA_CALHOUN	94914	67	CB_LIVE_OAK	4960
18	CB_CALHOUN	23235	68	TILDEN	76
19	CUERO	1831	69	CO_MCMULLEN	155
20	YORKTOWN	535	70	CB_MCMULLEN	4626
21	YOAKUM	550	71	BISHOP	848
22	CO_DE_WITT	1182	72	CORPUS_CHRISTI	119046
23	CA_DE_WITT	7212	73	PORT_ARANSAS	2161
24	CB_DE_WITT	3070	74	ROBSTOWN	2820
25	BENAVIDES	747	75	CO_NUECES	3834
26	FREER	1227	76	CA_NUECES	54448
27	SAN_DIEGO	1294	77	CB_NUECES	4354
28	CO_DUVAL	448	78	REFUGIO	439
29	CB_DUVAL	5506	79	WOODSBORO	278
30	GOLIAD	671	80	CO_REFUGIO	406
31	CO_GOLIAD	921	81	CB_REFUGIO	940
32	CA_GOLIAD	16000	82	MATHIS	1262
33	CB_GOLIAD	1934	83	ARANSAS_PASS	2010
34	GONZALES	2932	84	GREGORY	725
35	NIXON	653	85	INGLESIDE	1789
36	CO_GONZALES	2593	86	ODEM	636
37	CA_GONZALES	2672	87	PORTLAND	2980
38	CB_GONZALES	6775	88	SINTON	1416
39	EDNA	1573	89	TAFT	827
40	GANADO	418	90	TAFT_SOUTHWEST	407
41	CO_JACKSON	1338	91	CO_SAN_PATRICIO	3540
42	CA_JACKSON	66	92	CA_SAN_PATRICIO	28008
43	CB_JACKSON	61120	93	CB_SAN_PATRICIO	4672
44	ALICE	9410	94	BLOOMINGTON	515
45	ORANGE_GROVE	403	95	VICTORIA	16243
46	PREMONT	1351	96	CO_VICTORIA	4125
47	CO_JIM_WELLS	2560	97	CA_VICTORIA	79827
48	CA_JIM_WELLS	347	98	CB_VICTORIA	14985
49	CB_JIM_WELLS	4652			
50	SARITA	14			

Table 5.3 PLAN 0 - No Rules

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__ARANSAS	31		400
CA_ARANSAS		0	293
CB_ARANSAS		0	107
			400
CARRIZO-WILCOX__BEE	24		394
CA_BEE		0	5
CB_BEE		0	389
			394
GULF_COAST__BEE	31		14577
BEEVILLE		38	3730
CO_BEE		0	2921
CB_BEE		0	2201
CA_GOLIAD		26	5725
			14577
GULF_COAST__BROOKS	31		14577
FALFURRIAS		53	1372
CO_BROOKS		0	769
CA_BROOKS		0	18
CB_BROOKS		0	1690
PREMONT		61	1351
KINGSVILLE		32	9179
BISHOP		70	198
			14577
GULF_COAST__CALHOUN	31		2940
CO_CALHOUN		0	2219
CA_CALHOUN		0	721
			2940
LOCAL__CALHOUN	10		12600
CA_CALHOUN		0	12600
			12600
SHORTAGE__CALHOUN	999		999999
			0

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__DE_WITT	31		15866
CO_DE_WITT		0	1182
CA_DE_WITT		0	7212
CB_DE_WITT		0	3070
CA_GOLIAD		30	321
VICTORIA		25	4081
			15866
GULF_COAST__DUVAL	31		23970
BENAVIDES		68	747
FREER		61	1227
SAN_DIEGO		55	1294
CO_DUVAL		0	448
CB_DUVAL		0	5506
ALICE		25	5599
CA_LIVE_OAK		20	8352
PORTLAND		40	797
			23970
GULF_COAST__GOLIAD	31		12809
CO_GOLIAD		0	921
CA_GOLIAD		0	9954
CB_GOLIAD		0	1934
			12809
CARRIZO-WILCOX__GONZ	16		19840
CUERO		48	1831
GONZALES		41	2932
NIXON		91	653
CA_GONZALES		0	2672
GANADO		92	37
HALLETTSVILLE		70	831
YOAKUM		75	915
VICTORIA		31	6262
CA_VICTORIA		26	3707
			19840
GULF_COAST__GONZALES	39		2083
EDNA		60	1573
BLOOMINGTON		84	510
			2083

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
QUEEN_CITY__GONZALES	44		6104
YOAKUM		88	550
CB_GONZALES		0	5549
BLOOMINGTON		84	5

			6104
SPARTA_SAND__GONZALE	49		16340
PORT_LAVACA		42	3213
YORKTOWN		88	535
GOLIAD		85	671
SHINER		84	746
VICTORIA		31	5900

			11065
LOCAL__GONZALES	10		4200
CO_GONZALES		0	2593
CB_GONZALES		0	1226
GANADO		92	381

			4200
GULF_COAST__JACKSON	39		28343
CO_JACKSON		0	1338
CA_JACKSON		0	66
CB_JACKSON		0	26939

			28343
SHORTAGE__JACKSON	999		999999

			0
GULF_COAST__JIM_WELL	31		11370
ALICE		25	3811
CO_JIM_WELLS		0	2560
CA_JIM_WELLS		0	347
CB_JIM_WELLS		0	4652

			11370
SHORTAGE__JIM_WELL	999		999999

			0

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__KENEDY	31		9550
CO_KENEDY		0	49
CB_KENEDY		0	1821
			1870
GULF_COAST__KLEBERG	31		17088
CO_KLEBERG		0	2028
CA_KLEBERG		0	2574
CB_KLEBERG		0	2612
BISHOP		72	650
PORT_ARANSAS		68	2161
CB_NUECES		45	3998
ARANSAS_PASS		67	2010
INGLESIDE		65	1055
			17088
GULF_COAST__LAVACA	39		38123
CB_JACKSON		14	14109
CO_LAVACA		0	1865
CA_LAVACA		0	6933
CB_LAVACA		0	15216
			38123
CARRIZO-WILCOX__LIVE	24		2399
CO_LIVE_OAK		0	795
CA_LIVE_OAK		0	1604
			2399
GULF_COAST__LIVE_OAK	31		5242
CA_LIVE_OAK		0	282
CB_LIVE_OAK		0	4960
			5242
LOCAL__LIVE_OAK	10		760
CA_LIVE_OAK		0	760
			760
SHORTAGE__LIVE_OAK	999		999999
			0

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
CARRIZO-WILCOX__MCMU	24		7909
CA_LIVE_OAK		26	5115
TILDEN		172	76
CO_MCMULLEN		0	155
CB_MCMULLEN		0	2563
			7909
GULF_COAST__MCMULLEN	31		1838
ORANGE_GROVE		91	403
THREE_RIVERS		82	485
CB_MCMULLEN		0	950
			1838
QUEEN_CITY__MCMULLEN	44		1105
CB_MCMULLEN		0	1105
			1105
SPARTA_SAND__MCMULLE	49		600
GEORGE_WEST		76	592
CB_MCMULLEN		0	8
			600
SHORTAGE__MCMULLEN	999		999999
			0
GULF_COAST__NUECES	31		3254
SARITA		396	14
CO_NUECES		0	2884
CB_NUECES		0	356
			3254
LOCAL__NUECES	10		950
CO_NUECES		0	950
			950

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__NUECES	999		999999

			0
GULF_COAST__REFUGIO	31		7768
ROCKPORT		47	1100
CO_ARANSAS		37	4426
CA_ARANSAS		78	261
POINT_COMFORT		115	237
SEADRIFT		90	398
CO_REFUGIO		86	406
CB_REFUGIO		61	940

			7768
GULF_COAST__SAN_PATR	31		5228
CO_SAN_PATRICIO		0	3540
CB_SAN_PATRICIO		0	1688

			5228
SHORTAGE__SAN_PATR	999		999999

			0
GULF_COAST__VICTORIA	39		41130
CO_VICTORIA		0	4125
CA_VICTORIA		0	22020
CB_VICTORIA		0	14985

			41130
TEXANA	43		75000
CA_CALHOUN		21	54928
CB_JACKSON		14	20072

			75000
CUERO_I&II	49		52000

			0
GUADALUPE	40		79000
CA_CALHOUN		19	24900
CA_VICTORIA		13	54100

			79000

Table 5.3 PLAN 0 - No Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
COLETO_CR	59		12500

			0
SAN_ANTONIO	40		25000
CA_CALHOUN		25	1765
CB_CALHOUN		23	23235

			25000
CHOKE-CORPUS_w_bay	43		230549
ROCKPORT		54	1224
CORPUS_CHRISTI		25	119046
ROBSTOWN		49	2820
CA_NUECES		18	54448
REFUGIO		92	439
WOODSBORO		106	278
MATHIS		62	1262
GREGORY		82	725
INGLESIDE		57	734
ODEM		78	636
PORTLAND		53	2183
SINTON		57	1416
TAFT		71	827
TAFT_SOUTHWEST		89	407
CA_SAN_PATRICIO		23	28008
CB_SAN_PATRICIO		40	2984

			217437

Table 5.4 PLAN 4 - Statewide Rules

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__ARANSAS	31		400
CA_ARANSAS		0	293
CB_ARANSAS		0	107

			400
CARRIZO-WILCOX__BEE	24		394
CB_BEE		0	394

			394
GULF_COAST__BEE	31		14577
BEEVILLE		0	3730
CO_BEE		0	2921
CA_BEE		0	5
CB_BEE		0	2196
CA_GOLIAD		26	5725

			14577
GULF_COAST__BROOKS	31		14577
FALFURRIAS		0	1372
CO_BROOKS		0	769
CA_BROOKS		0	18
CB_BROOKS		0	1690
ALICE		53	554
CB_NUECES		50	3404
INGLESIDE		67	1789

			9596
GULF_COAST__CALHOUN	31		2940
POINT_COMFORT		0	237
PORT_LAVACA		0	86
SEADRIFT		0	398
CO_CALHOUN		0	2219

			2940
LOCAL__CALHOUN	10		12600
CB_CALHOUN		0	12600

			12600
SHORTAGE__CALHOUN	999		999999

			0

Table 5.4 PLAN 4 - Statewide Rules (Continued).

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__DE_WITT	31		15866
CUERO		0	1831
YORKTOWN		0	535
YOAKUM		0	550
CO_DE_WITT		0	1182
CA_DE_WITT		0	7212
CB_DE_WITT		0	3070
CA_GOLIAD		30	992
CB_VICTORIA		24	494

			15866
GULF_COAST__DUVAL	31		23970
BENAVIDES		0	747
FREER		0	1227
SAN_DIEGO		0	1294
CO_DUVAL		0	448
CB_DUVAL		0	5506
ALICE		25	6799
CA_LIVE_OAK		20	7949

			23970
GULF_COAST__GOLIAD	31		12809
GOLIAD		0	671
CO_GOLIAD		0	921
CA_GOLIAD		0	9283
CB_GOLIAD		0	1934

			12809
CARRIZO-WILCOX__GONZ	16		19840
CA_VICTORIA		26	19840

			19840
GULF_COAST__GONZALES	39		2083
GONZALES		0	2083

			2083
QUEEN_CITY__GONZALES	44		6104
GONZALES		0	849
CO_GONZALES		0	2593
CA_GONZALES		0	2662

			6104

Table 5.4 PLAN 4 - Statewide Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SPARTA_SAND__GONZALE	49		16340
PORT_LAVACA		42	3127
NIXON		0	653
CA_GONZALES		0	10
CB_GONZALES		0	2575
CB_VICTORIA		27	5335

			11700
LOCAL__GONZALES	10		4200
CB_GONZALES		0	4200

			4200
GULF_COAST__JACKSON	39		28343
EDNA		0	1573
GANADO		0	418
CO_JACKSON		0	1338
CA_JACKSON		0	66
CB_JACKSON		0	24948

			28343
SHORTAGE__JACKSON	999		999999

			0
GULF_COAST__JIM_WELL	31		11370
ALICE		0	2057
ORANGE_GROVE		0	403
PREMONT		0	1351
CO_JIM_WELLS		0	2560
CA_JIM_WELLS		0	347
CB_JIM_WELLS		0	4652

			11370
SHORTAGE__JIM_WELL	999		999999

			0
GULF_COAST__KENEDY	31		9550
SARITA		0	14
CO_KENEDY		0	49
CB_KENEDY		0	1821

			1884

Table 5.4 PLAN 4 - Statewide Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__KLEBERG	31		17088
KINGSVILLE		0	9179
CO_KLEBERG		0	2028
CA_KLEBERG		0	2574
CB_KLEBERG		0	2612
CO_NUECES		49	695

			17088
GULF_COAST__LAVACA	39		38123
CB_JACKSON		14	11617
HALLETTSVILLE		0	831
SHINER		0	746
YOAKUM		0	915
CO_LAVACA		0	1865
CA_LAVACA		0	6933
CB_LAVACA		0	15216

			38123
CARRIZO-WILCOX__LIVE	24		2399
GEORGE_WEST		0	592
CO_LIVE_OAK		0	795
CA_LIVE_OAK		0	1012

			2399
GULF_COAST__LIVE_OAK	31		5242
THREE_RIVERS		0	485
CA_LIVE_OAK		0	557
CB_LIVE_OAK		0	4200

			5242
LOCAL__LIVE_OAK	10		760
CB_LIVE_OAK		0	760

			760
SHORTAGE__LIVE_OAK	999		999999

			0

Table 5.4 PLAN 4 - Statewide Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
CARRIZO-WILCOX__MCMU	24		7909
CA_LIVE_OAK		26	6595
TILDEN		0	76
CO_MCMULLEN		0	155
CB_MCMULLEN		0	1083

			7909
GULF_COAST__MCMULLEN	31		1838
CB_MCMULLEN		0	1838

			1838
QUEEN_CITY__MCMULLEN	44		1105
CB_MCMULLEN		0	1105

			1105
SPARTA_SAND__MCMULLE	49		600
CB_MCMULLEN		0	600

			600
SHORTAGE__MCMULLEN	999		999999

			0
GULF_COAST__NUECES	31		3254
BISHOP		0	848
PORT_ARANSAS		0	2161
ROBSTOWN		0	245

			3254
LOCAL__NUECES	10		950
CB_NUECES		0	950

			950
SHORTAGE__NUECES	999		999999

			0

Table 5.4 PLAN 4 - Statewide Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__REFUGIO	31		7768
ROCKPORT		47	1018
CO_ARANSAS		37	4426
CA_ARANSAS		78	261
REFUGIO		0	439
WOODSBORO		0	278
CO_REFUGIO		0	406
CB_REFUGIO		0	940
			7768
GULF_COAST__SAN_PATR	31		5228
MATHIS		0	1262
ARANSAS_PASS		0	1371
GREGORY		0	725
ODEM		0	636
TAFT		0	827
TAFT_SOUTHWEST		0	407
			5228
SHORTAGE__SAN_PATR	999		999999
			0
GULF_COAST__VICTORIA	39		41130
BLOOMINGTON		0	515
VICTORIA		0	16243
CO_VICTORIA		0	4125
CA_VICTORIA		0	11091
CB_VICTORIA		0	9156
			41130
TEXANA	43		75000
CA_CALHOUN		21	50445
CB_JACKSON		14	24555
			75000
CUERO_I&II	49		52000
			0

Table 5.4 PLAN 4 - Statewide Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GUADALUPE	40		79000
CA_CALHOUN		19	30104
CA_VICTORIA		13	48896

			79000
COLETO_CR	59		12500

			0
SAN_ANTONIO	40		25000
CA_CALHOUN		25	14365
CB_CALHOUN		23	10635

			25000
CHOKE-CORPUS_w_bay	43		230549
ROCKPORT		54	1306
CORPUS_CHRISTI		25	119046
ROBSTOWN		49	2575
CO_NUECES		45	3139
CA_NUECES		18	54448
ARANSAS_PASS		59	639
PORTLAND		53	2980
SINTON		57	1416
CO_SAN_PATRICIO		43	3540
CA_SAN_PATRICIO		23	28008
CB_SAN_PATRICIO		40	4672

			221769

Table 5.5 PLAN 9 - Statewide & Regional Rules

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__ARANSAS	31		400
ROCKPORT		0	53
CO_ARANSAS		0	80
CA_ARANSAS		0	160
CB_ARANSAS		0	107
			400
CARRIZO-WILCOX__BEE	24		394
BEEVILLE		0	157
CO_BEE		0	6
CA_BEE		0	5
CB_BEE		0	157
			325
GULF_COAST__BEE	31		14577
BEEVILLE		0	3573
CO_BEE		0	2915
CB_BEE		0	2433
ARANSAS_PASS		57	1372
INGLESIDE		55	1789
CO_SAN_PATRICIO		43	2495
			14577
GULF_COAST__BROOKS	31		14577
FALFURRIAS		0	1372
CO_BROOKS		0	769
CA_BROOKS		0	18
CB_BROOKS		0	1690
KINGSVILLE		38	2344
			6193
GULF_COAST__CALHOUN	31		2940
POINT_COMFORT		0	237
SEADRIFT		0	398
CO_CALHOUN		0	1129
CB_CALHOUN		0	1176
			2940
LOCAL__CALHOUN	10		12600
CB_CALHOUN		0	12600
			12600

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__CALHOUN	999		999999
CB_CALHOUN		0	9459
			9459
GULF_COAST__DE_WITT	31		15866
CUERO		0	1831
YORKTOWN		0	535
YOAKUM		0	550
CO_DE_WITT		0	1182
CA_DE_WITT		0	6346
CB_DE_WITT		0	3070
VICTORIA		31	2352
			15866
GULF_COAST__DUVAL	31		23970
BENAVIDES		0	747
FREER		0	1227
SAN_DIEGO		0	1294
CO_DUVAL		0	448
CB_DUVAL		0	5506
ALICE		31	7249
CORPUS_CHRISTI		23	2668
PORT_ARANSAS		53	1706
CO_NUECES		42	145
PORTLAND		46	2980
			23970
GULF_COAST__GOLIAD	31		12809
CO_ARANSAS		48	1239
CO_CALHOUN		65	1090
GOLIAD		0	671
CO_GOLIAD		0	921
CA_GOLIAD		0	5123
CB_GOLIAD		0	1934
VICTORIA		38	1831
			12809

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
CARRIZO-WILCOX__GONZ	16		19840
PORT_LAVACA		45	3213
GONZALES		0	2932
NIXON		0	653
CO_GONZALES		0	2177
CA_GONZALES		0	1839
CB_GONZALES		0	2019
VICTORIA		34	7007
			19840
GULF_COAST__GONZALES	39		2083
CO_GONZALES		0	416
CA_GONZALES		0	833
CB_GONZALES		0	556
			1805
QUEEN_CITY__GONZALES	44		6104
			0
SPARTA_SAND__GONZALE	49		16340
			0
LOCAL__GONZALES	10		4200
CB_GONZALES		0	4200
			4200
GULF_COAST__JACKSON	39		28343
EDNA		0	1573
GANADO		0	418
CO_JACKSON		0	1338
CA_JACKSON		0	66
CB_JACKSON		0	24948
			28343
SHORTAGE__JACKSON	999		999999
			0

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__JIM_WELL	31		11370
ALICE		0	2161
ORANGE_GROVE		0	403
PREMONT		0	1351
CO_JIM_WELLS		0	2560
CA_JIM_WELLS		0	347
CB_JIM_WELLS		0	4548
			11370
SHORTAGE__JIM_WELL	999		999999
CB_JIM_WELLS		0	104
			104
GULF_COAST__KENEDY	31		9550
SARITA		0	14
CO_KENEDY		0	49
CB_KENEDY		0	1821
			1884
GULF_COAST__KLEBERG	31		17088
KINGSVILLE		0	6835
CO_KLEBERG		0	2028
CA_KLEBERG		0	2574
CB_KLEBERG		0	2612
CO_NUECES		55	3039
			17088
GULF_COAST__LAVACA	39		38123
HALLETTSVILLE		0	831
SHINER		0	746
YOAKUM		0	915
CO_LAVACA		0	1865
CA_LAVACA		0	6933
CB_LAVACA		0	15216
			26506
CARRIZO-WILCOX__LIVE	24		2399
GEORGE_WEST		0	592
THREE_RIVERS		0	230
CA_LIVE_OAK		0	618
CB_LIVE_OAK		0	959
			2399

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__LIVE_OAK	31		5242
THREE_RIVERS		0	255
CO_LIVE_OAK		0	795
CA_LIVE_OAK		0	2096
CB_LIVE_OAK		0	2096

			5242
LOCAL__LIVE_OAK	10		760
CB_LIVE_OAK		0	760

			760
SHORTAGE__LIVE_OAK	999		999999
CB_LIVE_OAK		0	1145

			1145
CARRIZO-WILCOX__MCMU	24		7909
TILDEN		0	76
CB_MCMULLEN		0	3163
ROBSTOWN		57	2126
MATHIS		63	625
SINTON		63	1416
TAFT		73	503

			7909
GULF_COAST__MCMULLEN	31		1838
CO_MCMULLEN		0	155
CB_MCMULLEN		0	735
MATHIS		65	637
ARANSAS_PASS		63	311

			1838
QUEEN_CITY__MCMULLEN	44		1105
CB_MCMULLEN		0	442

			442
SPARTA_SAND__MCMULLE	49		600
CB_MCMULLEN		0	240

			240

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__MCMULLEN	999		999999
CB_MCMULLEN		0	46

			46
GULF_COAST__NUECES	31		3254
BISHOP		0	848
PORT_ARANSAS		0	455
CO_NUECES		0	650
CB_NUECES		0	1301

			3254
LOCAL__NUECES	10		950
CB_NUECES		0	950

			950
SHORTAGE__NUECES	999		999999
CB_NUECES		0	2103

			2103
GULF_COAST__REFUGIO	31		7768
ROCKPORT		53	2271
CO_ARANSAS		43	3107
REFUGIO		0	439
WOODSBORO		0	278
CO_REFUGIO		0	406
CB_REFUGIO		0	940
ARANSAS_PASS		61	327

			7768
GULF_COAST__SAN_PATR	31		5228
GREGORY		0	725
ODEM		0	636
TAFT		0	324
TAFT_SOUTHWEST		0	407
CO_SAN_PATRICIO		0	1045
CB_SAN_PATRICIO		0	2091

			5228
SHORTAGE__SAN_PATR	999		999999
CB_SAN_PATRICIO		0	2581

			2581

Table 5.5 PLAN 9 - Statewide & Regional Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__VICTORIA	39		41130
BLOOMINGTON		0	515
VICTORIA		0	5053
CO_VICTORIA		0	4125
CA_VICTORIA		0	16452
CB_VICTORIA		0	14985
			41130
TEXANA	43		75000
CA_CALHOUN		21	38828
CB_JACKSON		14	36172
			75000
CUERO_I&II	49		52000
CA_DE_WITT		41	866
CA_VICTORIA		28	26338
			27204
GUADALUPE	40		79000
CA_CALHOUN		19	41963
CA_VICTORIA		13	37037
			79000
COLETO_CR	59		12500
			0
SAN_ANTONIO	40		25000
CA_CALHOUN		25	14123
CA_GOLIAD		32	10877
			25000
CHOKE-CORPUS_w_bay	43		230549
CA_ARANSAS		87	394
CA_LIVE_OAK		41	13399
CORPUS_CHRISTI		25	116378
ROBSTOWN		49	694
CA_NUECES		18	54448
CA_SAN_PATRICIO		23	28008
			213321

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__ARANSAS	31		400
ROCKPORT		0	53
CO_ARANSAS		0	80
CA_ARANSAS		0	160
CB_ARANSAS		0	107

			400
CARRIZO-WILCOX__BEE	24		394
BEEVILLE		0	157
CO_BEE		0	6
CA_BEE		0	5
CB_BEE		0	157

			325
GULF_COAST__BEE	31		14577
BEEVILLE		0	3573
CO_BEE		0	2915
CB_BEE		0	2433
CORPUS_CHRISTI		34	3635

			12556
GULF_COAST__BROOKS	31		14577
FALFURRIAS		0	1372
CO_BROOKS		0	769
CA_BROOKS		0	18
CB_BROOKS		0	1690

			3849
GULF_COAST__CALHOUN	31		2940
POINT_COMFORT		0	237
SEADRIFT		0	398
CO_CALHOUN		0	1129
CB_CALHOUN		0	1176

			2940
LOCAL__CALHOUN	10		12600
CB_CALHOUN		0	12600

			12600

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__CALHOUN	999		999999
CB_CALHOUN		0	9459

			9459
GULF_COAST__DE_WITT	31		15866
CUERO		0	1831
YORKTOWN		0	535
YOAKUM		0	550
CO_DE_WITT		0	1182
CA_DE_WITT		0	6346
CB_DE_WITT		0	3070
VICTORIA		31	2352

			15866
GULF_COAST__DUVAL	31		23970
BENAVIDES		0	747
FREER		0	1227
SAN_DIEGO		0	1294
CO_DUVAL		0	448
CB_DUVAL		0	5506
ALICE		31	5160
CORPUS_CHRISTI		23	9588

			23970
GULF_COAST__GOLIAD	31		12809
CO_CALHOUN		65	940
GOLIAD		0	671
CO_GOLIAD		0	921
CA_GOLIAD		0	5123
CB_GOLIAD		0	1934
VICTORIA		38	3220

			12809
CARRIZO-WILCOX__GONZ	16		19840
PORT_LAVACA		45	3213
GONZALES		0	2932
NIXON		0	653
CO_GONZALES		0	2177
CA_GONZALES		0	2672
CB_GONZALES		0	2575
VICTORIA		34	5618

			19840

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__GONZALES	39		2083
CO_GONZALES		0	416

			416
QUEEN_CITY__GONZALES	44		6104

			0
SPARTA_SAND__GONZALE	49		16340

			0
LOCAL__GONZALES	10		4200
CB_GONZALES		0	4200

			4200
GULF_COAST__JACKSON	39		28343
EDNA		0	1573
GANADO		0	418
CO_JACKSON		0	1338
CA_JACKSON		0	66
CB_JACKSON		0	24948

			28343
SHORTAGE__JACKSON	999		999999

			0
GULF_COAST__JIM_WELL	31		11370
ALICE		0	2161
ORANGE_GROVE		0	403
PREMONT		0	1351
CO_JIM_WELLS		0	2560
CA_JIM_WELLS		0	347
CB_JIM_WELLS		0	4548

			11370
SHORTAGE__JIM_WELL	999		999999
CB_JIM_WELLS		0	104

			104

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GULF_COAST__KENEDY	31		9550
SARITA		0	14
CO_KENEDY		0	49
CB_KENEDY		0	1821
			1884
GULF_COAST__KLEBERG	31		17088
KINGSVILLE		0	6835
CO_KLEBERG		0	2028
CA_KLEBERG		0	2574
CB_KLEBERG		0	2612
			14049
GULF_COAST__LAVACA	39		38123
HALLETTSVILLE		0	831
SHINER		0	746
YOAKUM		0	915
CO_LAVACA		0	1865
CA_LAVACA		0	6933
CB_LAVACA		0	15216
			26506
CARRIZO-WILCOX__LIVE	24		2399
GEORGE_WEST		0	251
THREE_RIVERS		0	230
CA_LIVE_OAK		0	959
CB_LIVE_OAK		0	959
			2399
GULF_COAST__LIVE_OAK	31		5242
THREE_RIVERS		0	255
CO_LIVE_OAK		0	795
CA_LIVE_OAK		0	2096
CB_LIVE_OAK		0	2096
			5242
LOCAL__LIVE_OAK	10		760
CB_LIVE_OAK		0	760
			760

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__LIVE_OAK	999		999999
CB_LIVE_OAK		0	1145

			1145
CARRIZO-WILCOX__MCMU	24		7909
KINGSVILLE		42	2344
TILDEN		0	76
CB_MCMULLEN		0	3163

			5583
GULF_COAST__MCMULLEN	31		1838
CO_MCMULLEN		0	155
CB_MCMULLEN		0	735

			890
QUEEN_CITY__MCMULLEN	44		1105
CB_MCMULLEN		0	442

			442
SPARTA_SAND__MCMULLE	49		600
CB_MCMULLEN		0	240

			240
SHORTAGE__MCMULLEN	999		999999
CB_MCMULLEN		0	46

			46
GULF_COAST__NUECES	31		3254
BISHOP		0	848
CO_NUECES		0	650
CA_NUECES		0	455
CB_NUECES		0	1301

			3254
LOCAL__NUECES	10		950
CB_NUECES		0	950

			950

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
SHORTAGE__NUECES	999		999999
CB_NUECES		0	2103

			2103
GULF_COAST__REFUGIO	31		7768
CO_CALHOUN		65	150
REFUGIO		0	439
WOODSBORO		0	278
CO_REFUGIO		0	406
CB_REFUGIO		0	940

			2213
GULF_COAST__SAN_PATR	31		5228
TAFT_SOUTHWEST		0	1
CO_SAN_PATRICIO		0	1045
CA_SAN_PATRICIO		0	2091
CB_SAN_PATRICIO		0	2091

			5228
SHORTAGE__SAN_PATR	999		999999
CB_SAN_PATRICIO		0	2581

			2581
GULF_COAST__VICTORIA	39		41130
BLOOMINGTON		0	515
VICTORIA		0	5053
CO_VICTORIA		0	4125
CA_VICTORIA		0	16452
CB_VICTORIA		0	14985

			41130
TEXANA	43		75000
CA_CALHOUN		21	38828
CB_JACKSON		14	36172

			75000
CUERO_I&II	49		52000
CA_DE_WITT		41	866
CA_VICTORIA		28	15855

			16721

Table 5.6 PLAN 10 - Statewide, Regional, & Parc Rules (Continued)

SUPPLY / DEMAND	COST IN PLACE ($/AF)	TR.COST ($/AF)	AMOUNT (AF/Y)
GUADALUPE	40		79000
CA_CALHOUN		19	31480
CA_VICTORIA		13	47520

			79000
COLETO_CR	59		12500
CA_GOLIAD		0	10877

			10877
SAN_ANTONIO	40		25000
CA_ARANSAS		78	394
CA_CALHOUN		25	24606

			25000
CHOKE-CORPUS_w_bay	43		230549
ROCKPORT		13	2271
CO_ARANSAS		12	4346
ALICE		12	2089
GEORGE_WEST		23	341
CA_LIVE_OAK		41	13058
CORPUS_CHRISTI		6	105823
PORT_ARANSAS		15	2161
ROBSTOWN		12	2820
CO_NUECES		11	3184
CA_NUECES		18	53993
MATHIS		15	1262
ARANSAS_PASS		14	2010
GREGORY		20	725
INGLESIDE		14	1789
ODEM		19	636
PORTLAND		13	2980
SINTON		14	1416
TAFT		17	827
TAFT_SOUTHWEST		22	406
CO_SAN_PATRICIO		10	2495
CA_SAN_PATRICIO		23	25917

			230549

Table 5.7 Comparison of Allocations

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB	1992 WATER PLAN (AF/Y)
ROCKPORT	2324		
GULF_COAST__ARANSAS	53	CHOKE-CORPUS w bay	2324
CHOKE-CORPUS_w_bay	2271		
CO_ARANSAS	4426		
GULF_COAST__ARANSAS	80	GULF COAST	274
CHOKE-CORPUS_w_bay	4346	CHOKE-CORPUS w bay	4152
CA_ARANSAS	554		
GULF_COAST__ARANSAS	160	GULF COAST	122
SAN_ANTONIO	394	CHOKE-CORP	432
CB_ARANSAS	107		
GULF_COAST__ARANSAS	107	GULF COAST	3
		CHOKE-CORP	14
		LOCAL SUP6	90
BEEVILLE	3730		
CARRIZO-WILCOX__BEE	157	CHOKE-CORPUS w bay	3730
GULF_COAST__BEE	3573		
CO_BEE	2921		
CARRIZO-WILCOX__BEE	6	GULF COAST	2400
GULF_COAST__BEE	2915	CHOKE-CORPUS w bay	311
		GULF COAST	210
CA_BEE	5		
CARRIZO-WILCOX__BEE	5	GULF COAST	5
CB_BEE	2590		
CARRIZO-WILCOX__BEE	157	GULF COAST	121
GULF_COAST__BEE	2433	GULF COAST	1155
		GULF COAST	422
		LOCAL SUP6	739
		GULF COAST	153
FALFURRIAS	1372		
GULF_COAST__BROOKS	1372	GULF COAST	1372
CO_BROOKS	769		
GULF_COAST__BROOKS	769	GULF COAST	769
CA_BROOKS	18		
GULF_COAST__BROOKS	18	GULF COAST	18
CB_BROOKS	1690		
GULF_COAST__BROOKS	1690	GULF COAST	62
		GULF COAST	495
		GULF COAST	524
		LOCAL SUP6	609
POINT_COMFORT	237		
GULF_COAST__CALHOUN	237	TEXANA	237

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB	1992 WATER PLAN (AF/Y)
PORT_LAVACA	3213		
CARRIZO-WILCOX__GONZ	3213	CUERO I&II	3213
SEADRIFT	398		
GULF_COAST__CALHOUN	398	GULF COAST	398
CO_CALHOUN	2219		
GULF_COAST__CALHOUN	1129	GULF COAST	100
GULF_COAST__GOLIAD	940	TEXANA	140
GULF_COAST__REFUGIO	150	GULF COAST	350
		CUERO I&II	1606
		CANYON	8
		OTHER	15
CA_CALHOUN	94914		
TEXANA	38828	GULF COAST	1722
GUADALUPE	31480	GULF COAST	200
SAN_ANTONIO	24606	TEXANA	39959
		GULF COAST	41
		CUERO I&II	7945
		SAN ANTONI	25000
		GUADALUPE	19631
		CANYON	270
		GUADALUPE	146
CB_CALHOUN	23235		
GULF_COAST__CALHOUN	1176	GULF COAST	32
LOCAL__CALHOUN	12600	OTHER	1
SHORTAGE__CALHOUN	9459	GULF COAST	2
		GULF COAST	1976
		OTHER	6
		GUADALUPE	11285
		LOCAL SUP2	11908
		LOCAL SUP6	615
		LOCAL SUP6	2
		OTHER	3
CUERO	1831		
GULF_COAST__DE_WITT	1831	GULF COAST	1831
YORKTOWN	535		
GULF_COAST__DE_WITT	535	GULF COAST	535
YOAKUM	550		
GULF_COAST__DE_WITT	550	GULF COAST	550
		GULF COAST	915
CO_DE_WITT	1182		
GULF_COAST__DE_WITT	1182	GULF COAST	168
		GULF COAST	4
		GULF COAST	875
		GULF COAST	135
CA_DE_WITT	7212		
GULF_COAST__DE_WITT	6346	GULF COAST	20
CUERO_I&II	866	GULF COAST	47
		LOCAL SUP2	145
		CUERO I&II	7000

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB	1992 WATER PLAN (AF/Y)
CB_DE_WITT	3070		
GULF_COAST__DE_WITT	3070	GULF COAST	36
		GULF COAST	347
		GULF COAST	1
		GULF COAST	67
		GULF COAST	1
		GULF COAST	335
		GULF COAST	666
		LOCAL SUP2	225
		LOCAL SUP2	1156
		GULF COAST	40
		GULF COAST	196
BENAVIDES	747		
GULF_COAST__DUVAL	747	GULF COAST	747
FREER	1227		
GULF_COAST__DUVAL	1227	GULF COAST	1227
SAN_DIEGO	1294		
GULF_COAST__DUVAL	1294	GULF COAST	1084
		CHOKE-CORPUS w bay	210
CO_DUVAL	448		
GULF_COAST__DUVAL	448	GULF COAST	85
		GULF COAST	363
CB_DUVAL	5506		
GULF_COAST__DUVAL	5506	GULF COAST	407
		LOCAL SUP6	137
		GULF COAST	8
		GULF COAST	97
		GULF COAST	3095
		GULF COAST	407
		LOCAL SUP6	1355
GOLIAD	671		
GULF_COAST__GOLIAD	671	GULF COAST	671
CO_GOLIAD	921		
GULF_COAST__GOLIAD	921	GULF COAST	329
		GULF COAST	467
		GULF COAST	125
CA_GOLIAD	16000		
GULF_COAST__GOLIAD	5123	GULF COAST	1690
COLETO_CR	10877	COLETO	12500
		CANYON	1810
CB_GOLIAD	1934		
GULF_COAST__GOLIAD	1934	GULF COAST	280
		GULF COAST	496
		LOCAL SUP2	660
		GULF COAST	3
		GULF COAST	495
GONZALES	2932		
CARRIZO-WILCOX__GONZ	2932	GUADALUPE R. W CUERO	2932

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB 1992 WATER PLAN	(AF/Y)
NIXON	653		
CARRIZO-WILCOX__GONZ	653	CARRIZO - WILCOX	653
CO_GONZALES	2593		
CARRIZO-WILCOX__GONZ	2177	CARRIZO - WILCOX	21
GULF_COAST__GONZALES	416	CARRIZO - WILCOX	1300
		GULF COAST	22
		OTHER	50
		QUEEN CITY	122
		SPARTA	200
		GUADALUPE R. W CUERO	393
		CANYON	700
CA_GONZALES	2672		
CARRIZO-WILCOX__GONZ	2672	CARRIZO -	674
		OTHER	20
		QUEEN CITY	125
		SPARTA	30
		CANYON	200
		LOCAL SUP2	100
		GUADALUPE	700
		RETURN FL	823
CB_GONZALES	6775		
CARRIZO-WILCOX__GONZ	2575	CARRIZO -	39
LOCAL__GONZALES	4200	CARRIZO -	22
		CARRIZO -	1710
		CARRIZO -	119
		OTHER	120
		QUEEN CITY	335
		SPARTA	245
		LOCAL SUP2	600
		LOCAL SUP2	3585
EDNA	1573		
GULF_COAST__JACKSON	1573	TEXANA	1573
GANADO	418		
GULF_COAST__JACKSON	418	TEXANA	418
CO_JACKSON	1338		
GULF_COAST__JACKSON	1338	GULF COAST	200
		TEXANA	202
		GULF COAST	700
		TEXANA	111
		GULF COAST	125
CA_JACKSON	66		
GULF_COAST__JACKSON	66	GULF COAST	66
CB_JACKSON	61120		
GULF_COAST__JACKSON	24948	GULF COAST	17050
TEXANA	36172	GULF COAST	304
		LOCAL SUP2	1598
		GULF COAST	25000
		GULF COAST	350
		LOCAL SUP2	1313
		LOCAL SUP6	77
		GULF COAST	52
		GULF COAST	4272
		LOCAL SUP2	921
		LOCAL SUP2	139

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB 1992 WATER PLAN	(AF/Y)
ALICE	9410		
GULF_COAST__DUVAL	5160	CHOKE-CORPUS w bay	9410
GULF_COAST__JIM_WELL	2161		
CHOKE-CORPUS_w_bay	2089		
ORANGE_GROVE	403		
GULF_COAST__JIM_WELL	403	GULF COAST	403
PREMONT	1351		
GULF_COAST__JIM_WELL	1351	GULF COAST	1351
CO_JIM_WELLS	2560		
GULF_COAST__JIM_WELL	2560	GULF COAST	297
CA_JIM_WELLS	347		
GULF_COAST__JIM_WELL	347	CHOKE-CORP	347
CB_JIM_WELLS	4652		
GULF_COAST__JIM_WELL	4548	GULF COAST	1048
SHORTAGE__JIM_WELL	104	GULF COAST	227
		LOCAL SUP6	13
		GULF COAST	78
		GULF COAST	270
		GULF COAST	1837
		GULF COAST	227
		LOCAL SUP2	952
SARITA	14		
GULF_COAST__KENEDY	14	GULF COAST	14
CO_KENEDY	49		
GULF_COAST__KENEDY	49	GULF COAST	49
CB_KENEDY	1821		
GULF_COAST__KENEDY	1821		
KINGSVILLE	9179		
GULF_COAST__KLEBERG	6835	GULF COAST	170
CARRIZO-WILCOX__MCMU	2344	CHOKE-CORPUS w bay	9009
CO_KLEBERG	2028		
GULF_COAST__KLEBERG	2028	GULF COAST	1700
		CHOKE-CORPUS w bay	328
CA_KLEBERG	2574		
GULF_COAST__KLEBERG	2574	GULF COAST	51
		GULF COAST	2500
		CHOKE-CORP	23
CB_KLEBERG	2612		
GULF_COAST__KLEBERG	2612	GULF COAST	542
		GULF COAST	500
		GULF COAST	341
		LOCAL SUP2	100
		LOCAL SUP2	1129

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB 1992 WATER PLAN	(AF/Y)
HALLETTSVILLE	831		
GULF_COAST__LAVACA	831	GULF COAST	831
SHINER	746		
GULF_COAST__LAVACA	746	GULF COAST	746
YOAKUM	550		
GULF_COAST__DE_WITT	550	GULF COAST	550
		GULF COAST	915
CO_LAVACA	1865		
GULF_COAST__LAVACA	1865	GULF COAST	5
		GULF COAST	20
CA_LAVACA	6933		
GULF_COAST__LAVACA	6933	GULF COAST	933
		GULF COAST	6000
CB_LAVACA	15216		
GULF_COAST__LAVACA	15216	GULF COAST	76
		GULF COAST	12235
		GULF COAST	381
		LOCAL SUP2	762
		LOCAL SUP2	1718
		GULF COAST	1
		GULF COAST	5
		GULF COAST	38
GEORGE_WEST	592		
CARRIZO-WILCOX__LIVE	251	GULF COAST	592
CHOKE-CORPUS_w_bay	341		
THREE_RIVERS	485		
CARRIZO-WILCOX__LIVE	230	CHOKE-CORPUS w bay	485
GULF_COAST__LIVE_OAK	255		
CO_LIVE_OAK	795		
GULF_COAST__LIVE_OAK	795	GULF COAST	695
		CHOKE-CORPUS w bay	100
CA_LIVE_OAK	16113		
CARRIZO-WILCOX__LIVE	959	CARRIZO -	15000
GULF_COAST__LIVE_OAK	2096	GULF COAST	927
CHOKE-CORPUS_w_bay	13058	CHOKE-CORP	186
CB_LIVE_OAK	4960		
CARRIZO-WILCOX__LIVE	959	CARRIZO -	53
GULF_COAST__LIVE_OAK	2096	CARRIZO -	2346
LOCAL__LIVE_OAK	760	GULF COAST	172
SHORTAGE__LIVE_OAK	1145	GULF COAST	1229
		GULF COAST	398
		LOCAL SUP2	55
		LOCAL SUP2	707
TILDEN	76		
CARRIZO-WILCOX__MCMU	76	CARRIZO - WILCOX	76
CO_MCMULLEN	155		
GULF_COAST__MCMULLEN	155	CARRIZO - WILCOX	155

Table 5.7 Comparison of Allocations (Continued)

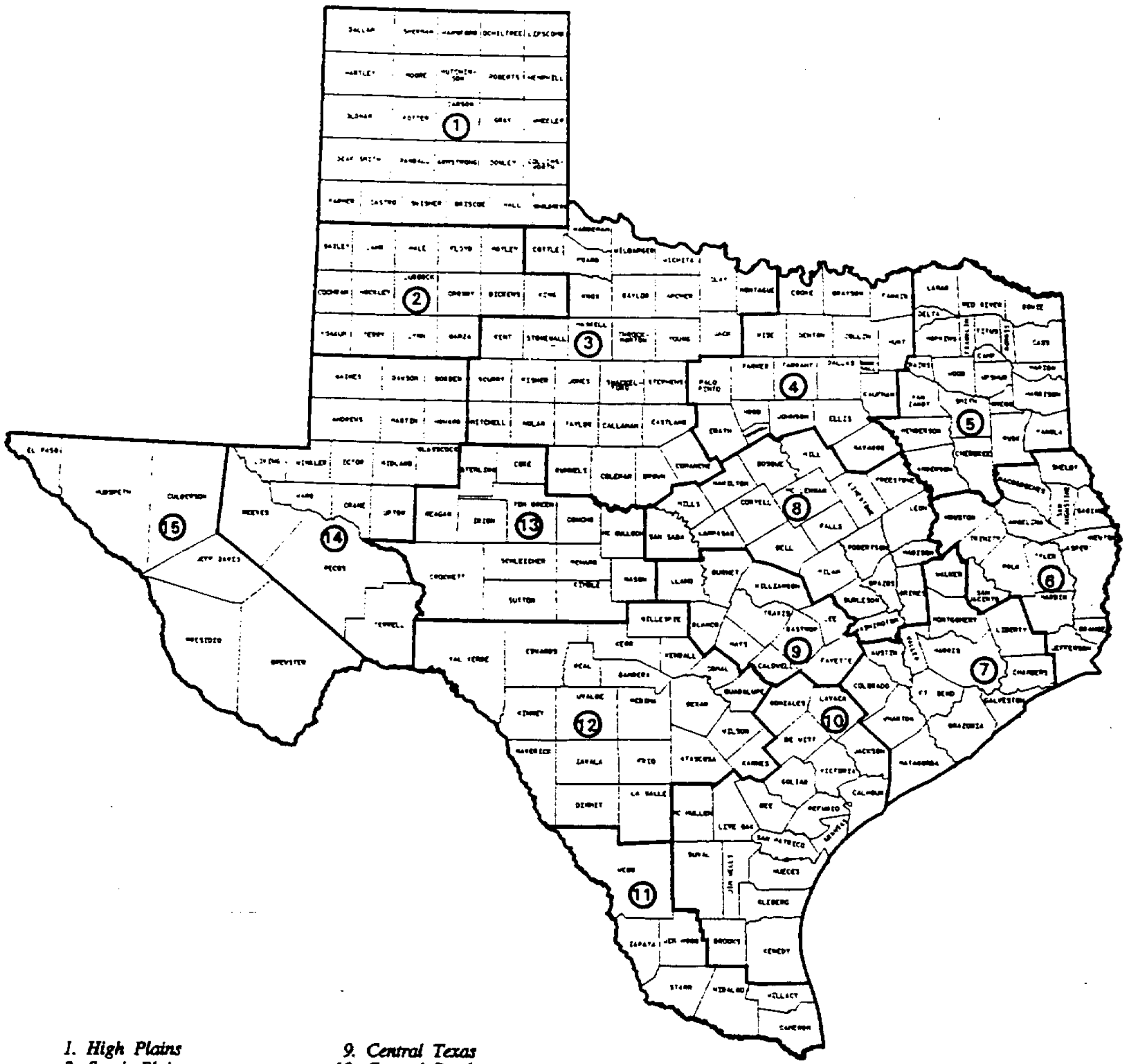
AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB 1992 WATER PLAN	(AF/Y)
CB_MCMULLEN	4626		
CARRIZO-WILCOX__MCMU	3163	CARRIZO -	165
GULF_COAST__MCMULLEN	735	GULF COAST	215
QUEEN_CITY__MCMULLEN	442	GULF COAST	215
SPARTA_SAND__MCMULLE	240	LOCAL SUP6	1237
SHORTAGE__MCMULLEN	46		
BISHOP	848		
GULF_COAST__NUECES	848	CHOKE-CORPUS w bay	848
CORPUS_CHRISTI	119046		
GULF_COAST__BEE	3635	CHOKE-CORPUS w bay	5587
GULF_COAST__DUVAL	9588	CHOKE-CORPUS w bay	113459
CHOKE-CORPUS_w_bay	105823		
PORT_ARANSAS	2161		
CHOKE-CORPUS_w_bay	2161	CHOKE-CORPUS w bay	108
		CHOKE-CORPUS w bay	2053
ROBSTOWN	2820		
CHOKE-CORPUS_w_bay	2820		
CO_NUECES	3834		
GULF_COAST__NUECES	650	GULF COAST	244
CHOKE-CORPUS_w_bay	3184	CHOKE-CORPUS w bay	105
		GULF COAST	175
		CHOKE-CORPUS w bay	3310
CA_NUECES	54448		
GULF_COAST__NUECES	455	GULF COAST	152
CHOKE-CORPUS_w_bay	53993	CHOKE-CORP	942
		CHOKE-CORP	3000
		GULF COAST	152
		CHOKE-CORP	24385
		TEXANA	25317
		TEXANA	500
CB_NUECES	4354		
GULF_COAST__NUECES	1301	GULF COAST	3
LOCAL__NUECES	950	GULF COAST	1108
SHORTAGE__NUECES	2103	GULF COAST	26
		LOCAL SUP2	912
		LOCAL SUP2	12
		GULF COAST	88
		GULF COAST	19
		GULF COAST	797
		GULF COAST	26
		LOCAL SUP2	170
		TEXANA	542
		LOCAL SUP2	288
REFUGIO	439		
GULF_COAST__REFUGIO	439	GULF COAST	439
WOODSBORO	278		
GULF_COAST__REFUGIO	278	GULF COAST	278

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB	1992 WATER PLAN (AF/Y)
CO_REFUGIO	406		
GULF_COAST__REFUGIO	406	GULF COAST	11
		GULF COAST	395
CB_REFUGIO	940		
GULF_COAST__REFUGIO	940	GULF COAST	165
		GULF COAST	25
		GULF COAST	102
		GULF COAST	271
		LOCAL SUP6	377
MATHIS	1262		
CHOKE-CORPUS_w_bay	1262	CHOKE-CORPUS w bay	1262
ARANSAS_PASS	2010		
CHOKE-CORPUS_w_bay	2010	CHOKE-CORPUS w bay	279
		CHOKE-CORPUS w bay	1
		CHOKE-CORPUS w bay	1730
GREGORY	725		
CHOKE-CORPUS_w_bay	725	CHOKE-CORPUS w bay	725
INGLESIDE	1789		
CHOKE-CORPUS_w_bay	1789	CHOKE-CORPUS w bay	1789
ODEM	636		
CHOKE-CORPUS_w_bay	636	CHOKE-CORPUS w bay	636
PORTLAND	2980		
CHOKE-CORPUS_w_bay	2980		
SINTON	1416		
CHOKE-CORPUS_w_bay	1416	GULF COAST	1416
TAFT	827		
CHOKE-CORPUS_w_bay	827	CHOKE-CORPUS w bay	827
		CHOKE-CORPUS w bay	407
TAFT_SOUTHWEST	407		
GULF_COAST__SAN_PATR	1	CHOKE-CORPUS w bay	407
CHOKE-CORPUS_w_bay	406		
CO_SAN_PATRICIO	3540		
GULF_COAST__SAN_PATR	1045	GULF COAST	690
CHOKE-CORPUS_w_bay	2495	CHOKE-CORPUS w bay	1987
		GULF COAST	522
		CHOKE-CORPUS w bay	341
CA_SAN_PATRICIO	28008		
GULF_COAST__SAN_PATR	2091	GULF COAST	5
CHOKE-CORPUS_w_bay	25917	CHOKE-CORP	27478
		GULF COAST	5
		CHOKE-CORP	520

Table 5.7 Comparison of Allocations (Continued)

AUTOMATED ALLOCATION SYSTEM	(AF/Y)	TWDB	1992 WATER PLAN (AF/Y)
CB_SAN_PATRICIO	4672		
GULF_COAST__SAN_PATR	2091	GULF COAST	43
SHORTAGE__SAN_PATR	2581	GULF COAST	2357
		GULF COAST	177
		LOCAL SUP2	50
		LOCAL SUP2	352
		GULF COAST	13
		LOCAL SUP2	69
		LOCAL SUP2	265
BLOOMINGTON	515		
GULF_COAST__VICTORIA	515	GULF COAST	515
VICTORIA	16243		
GULF_COAST__DE_WITT	2352	GULF COAST	3343
GULF_COAST__GOLIAD	3220	GULF COAST	12908
CARRIZO-WILCOX__GONZ	5618		
GULF_COAST__VICTORIA	5053		
CO_VICTORIA	4125		
GULF_COAST__VICTORIA	4125	GULF COAST	41
		GULF COAST	1859
		GULF COAST	2160
		GULF COAST	65
CA_VICTORIA	79827		
GULF_COAST__VICTORIA	16452	GULF COAST	17
CUERO_I&II	15855	GULF COAST	904
GUADALUPE	47520	GULF COAST	4729
		GUADALUPE	32000
		GUADALUPE	948
		RETURN FL	5000
		RETURN FL	20323
		CUERO I&II	15906
CB_VICTORIA	14985		
GULF_COAST__VICTORIA	14985	GULF COAST	780
		GULF COAST	7
		GULF COAST	29
		GULF COAST	1045
		GULF COAST	9541
		GULF COAST	766
		GULF COAST	1
		GULF COAST	562
		GULF COAST	1104
		GULF COAST	740
		LOCAL SUP2	300
		LOCAL SUP2	19
		GULF COAST	14
		LOCAL SUP6	77



1. *High Plains*
2. *South Plains*
3. *West Central Texas*
4. *North Central Texas*
5. *Northeast Texas*
6. *Deep East Texas*
7. *Gulf Coast*
8. *Heart of Texas*
9. *Central Texas*
10. *Coastal Bend*
11. *Lower Rio Grande*
12. *Edwards/Winter Garden*
13. *Concho Valley*
14. *Permian Basin*
15. *Upper Rio Grande*

Figure 5.1 Prospective water planning regions for the 1994 Texas Water Plan.

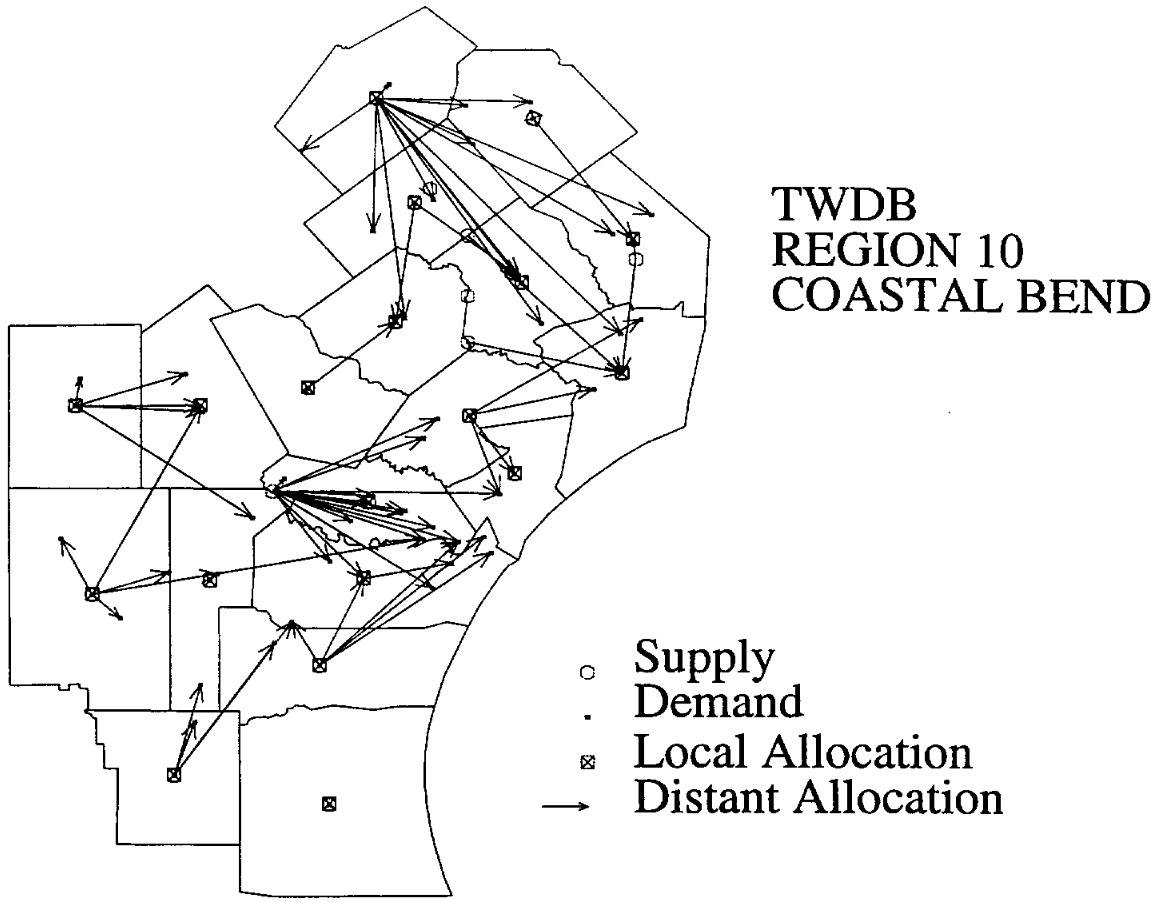


Figure 5.2 PLAN 0 - No Rules.

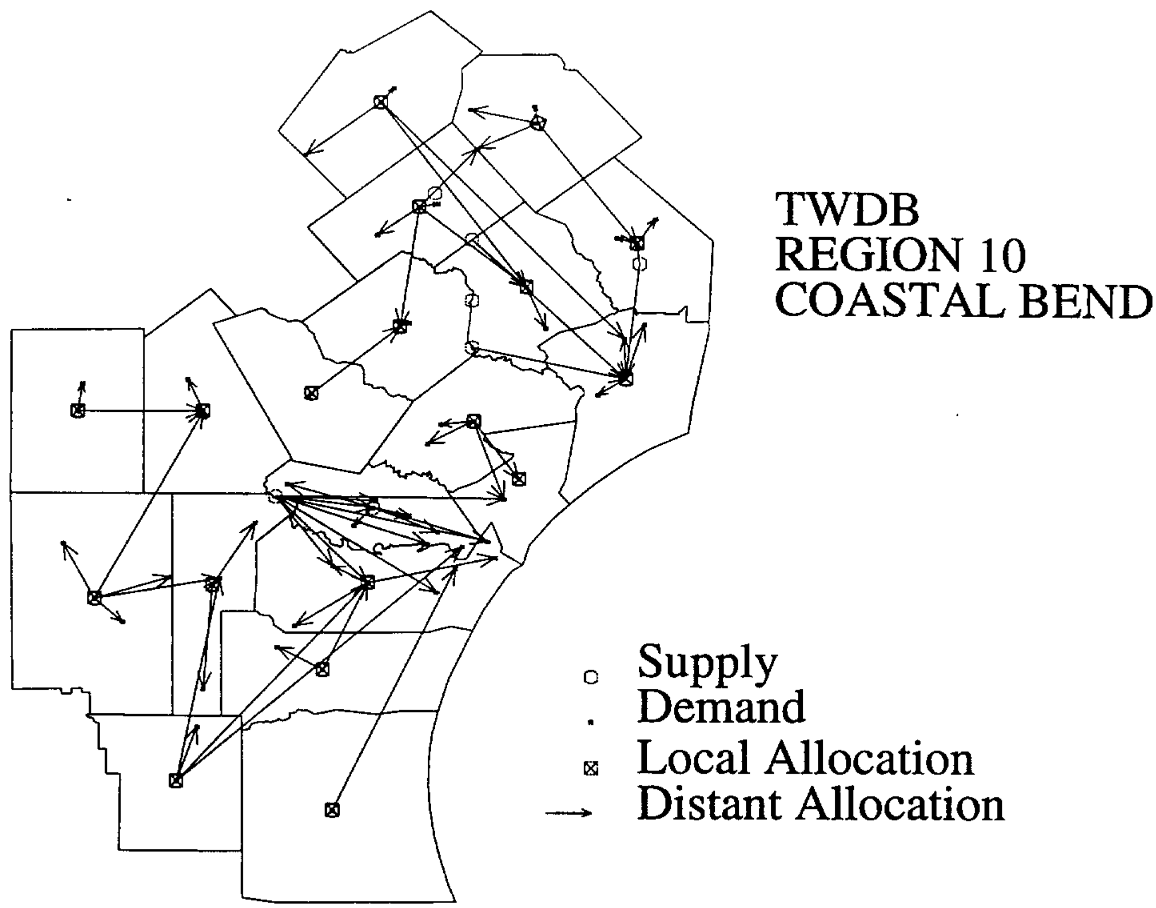


Figure 5.3 PLAN 4 - Statewide Rules.

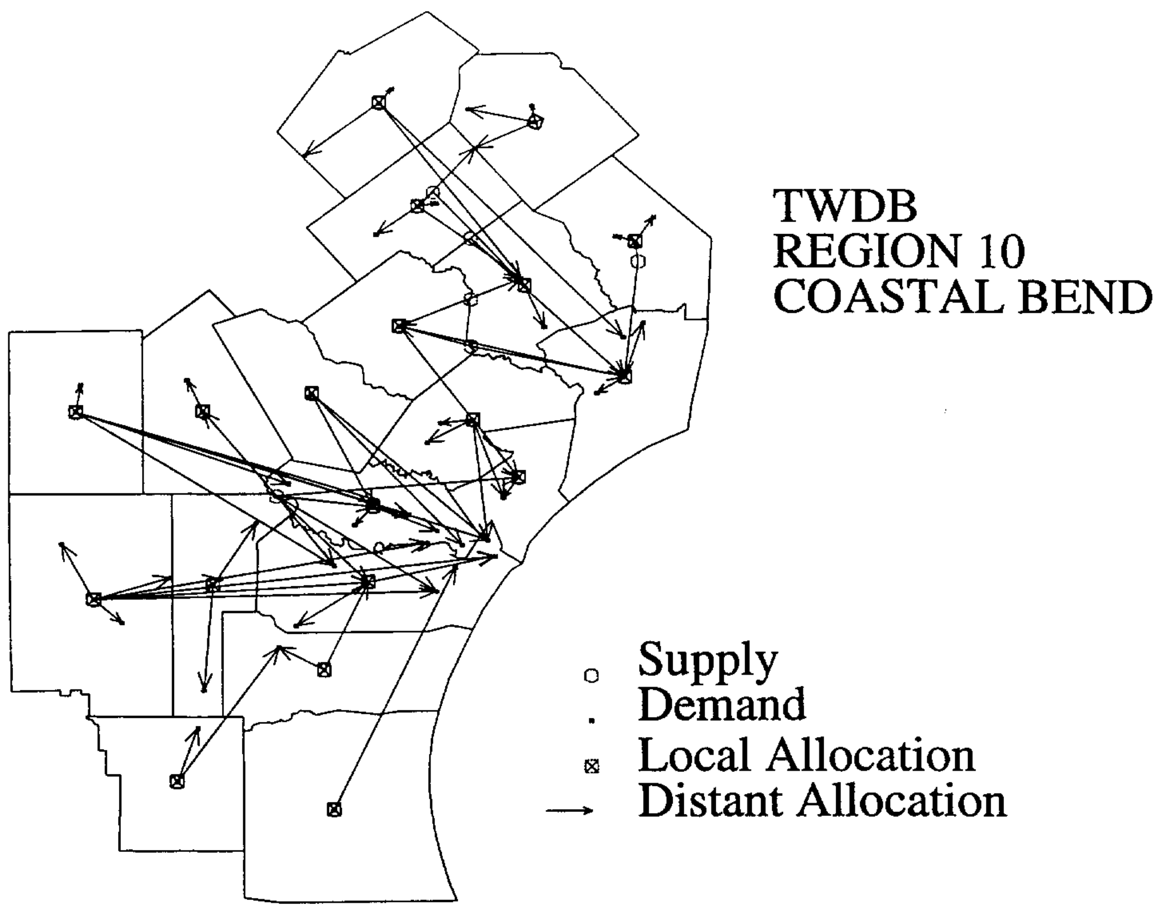


Figure 5.4 PLAN 9 - Statewide & Regional Rules.

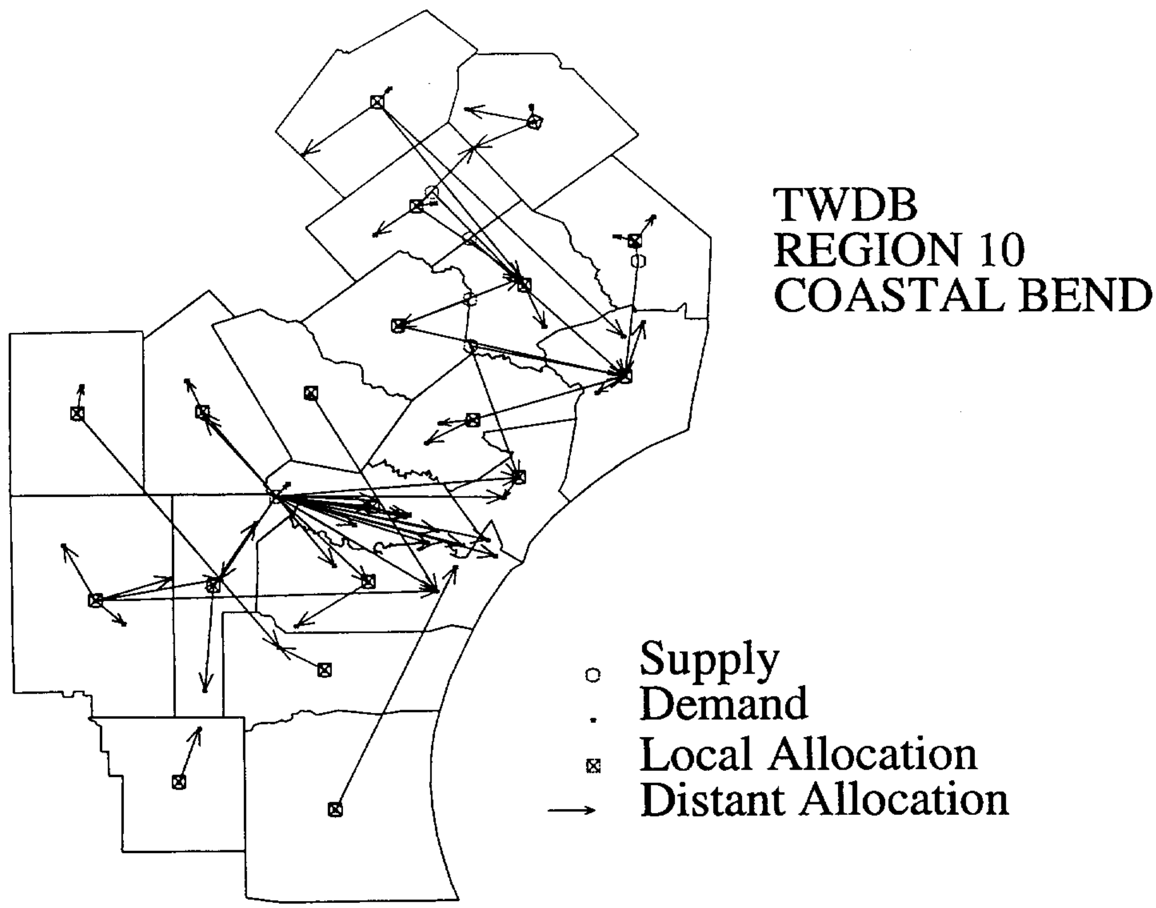


Figure 5.5 PLAN 10 - Statewide, Regional, & Parc Rules.

6.0 CONCLUSIONS

An expert geographic information system (expert GIS) for long-term regional water supply planning was developed in this research. The Automated Allocations System (AAS) has been evaluated through a small example problem developed to illustrate several features of the system, and a case study examining a 19-county study region in South Texas with several water supply sources and demand centers. The AAS is comprised of an expert system, which contains the logical rules and expertise of water resources planning experts; a GIS, which stores and analyzes spatially distributed water supply and demand data; and a network flow solver, to balance flows in the networks developed by the AAS with input from a water resource analyst. Commonly available water demand forecasts and water supply data are used in order to follow the logic of current planning methods and permit the updating and comparison of alternatives. The AAS system has been developed so that it can be expanded to include additional constraints and handle large water resources planning regions.

The system was successfully applied to the TWDB Coastal Bend planning region. The existence of generic categories of rules for regional water planning is evident from this case study. The categories include rules applicable on a statewide basis, a regional basis, or a local basis. The local scale rules are specific to individual arcs in the network model representation and need to be entered individually. However, the application of the small sets of statewide and regional rules is sufficient to generate relatively realistic solutions. A detailed comparison of the allocations made by the AAS and the 1990 Texas Water Plan was made. Many individual discrepancies were found, yet overall the plans are quite similar.

One of the original goals of this research project was to develop an expert GIS which would have the capability of aiding TWDB analysts in their work of preparing the Texas Water Plan. This objective has been met. This research has demonstrated that an automated system to allocate regional water resources can be made to produce results comparable to those of current methods by employing a GIS, an expert system, and an network flow solver. This system affords planners a process that is faster, less tedious, better documented and more rigorous and defensible than current methods. The current system is undergoing testing by TWDB personnel in an effort to fine-tune the Texas Coastal Bend region model. The task of finding a final set of detailed rules for this region is beyond the scope of this investigation and is better left to the professionals at the agency.

The research also demonstrates that there is a hierarchy of rules related to water resources allocations that can be exploited by focusing on rules that pertain to (1) statewide considerations, (2) regional considerations, (3) individual suppliers and demanders, and (4) individual arcs. This hierarchical rules structure was not anticipated at the outset of the research, and only became apparent once the data base and modeling system had been assembled and analysis of the regional planning problem undertaken. It is anticipated that other classes of rules will be identified through the continued application of the system to other regions and the further development of the system to consider additional constraints.

The modeling system makes extensive use of GIS data base management capabilities. This data model is perhaps the most important aspect of the system, as it allows the efficient and convenient construction of models representing a large number of possible water allocation scenarios. The expert system shell provides a convenient rule editing and execution facility. The hierarchical rule structure—state, regional, and local rules—is a unique feature discovered during the construction of the case study and will allow easy application of the system to other planning regions within the state.

The application of the system to other planning regions within Texas, or to the allocation of water statewide is straightforward and could be undertaken at this time. In addition, this system can be linked to more detailed hydrologic modeling systems which could provide input on the expected temporal and spatial variability of reservoir, aquifer, and river yields. Further advances and refinements in the model are needed and should be considered in future research. More work is needed to include other types of supply sources, water supply contracts, and other information (e.g., political constraints, environmental considerations). In addition, cases where certain stakeholders may be presented with a perceived "sub-optimal" solution due to the regional scale of the solution algorithm need to be investigated.

REFERENCES

Arnold, U., Datta, B., Haenscheid, P., Intelligent Geographic Information Systems (IGIS) and Surface Water Modeling, New Directions for Surface Water Modeling, Proceedings of a Symposium held in Baltimore, Maryland, May 1989, IAHS Publication No. 181, International Association of Hydrological Sciences, Washington, DC, p 407-416, 1989.

Bradley, S.P., Hax, A.C., Magnati, T. L., Applied Mathematical Programming, Addison-Wesley Pub. Co., Reading, 1977.

Brendecke, C.M., DeOreo, W.B., Payton, E.A., Rozaklis, L.T., Network Models of Water Rights and System Operations, J. Water Resources Planning and Management (ASCE) Vol. 115, No. 5, p. 684-696, 1989.

Brown, C., Monks, J.G., Park, J.R., Decision Making In Water Resource Allocation, Oregon Water Resources Research Institute Completion Report, 143 p., OWRR-B-016-ORE(2), 1972.

Brown, T.C., Harding, B.L., Lord, W.B., Consumptive Use of Streamflow Increases in the Colorado River Basin, Water Resources Bulletin, Vol. 24, No. 4, p. 801-814, August 1988.

Buras, N., Scientific Allocation of Water Resources, American Elsevier, Inc., New York, 1972.

Chen, W-K., Theory of Nets - Flows in Networks, John Wiley & Sons Inc., New York, 1990.

Fordham, J.W., Simulation Theory Applied To Water Resources Management - Phase III, Development Of Optimal Operating Rules, NTIS PB-220 352, Technical Report Series H-W, Publication No 13, 45 p., OWRR C-2153(3372)(2), 1972.

Foster, E.T. Jr, Chen, T.C., Newton, J.P., and Isu, E.O., Improved River Basin Utilization Through Systems Analysis, Water Resources Bulletin, Vol 8, No 5, p. 863-870, October 1972.

Greathouse, D., Clements, J., and Morris, K., The use of Expert Systems To Assist in Decisions Concerning Environmental Control, Critical Reviews in Environmental Control, Vol 19 Issue 4, 1989.

Hazeghi, K., Schmid, W., and Petalas, P., Application of Operations Research Techniques for a Problem in Water Resources Management: Economic Appraisal of Changes in Water Use Induced by Investments into Navigable Rivers and Canals, Modeling, Identification and Control in Environmental Systems, p 977-988, North-Holland, Publishing Company, 1978.

Hillier, F.S. and Lieberman, G.J., Operations Research 2nd Ed., Holden-Day, Inc., San Francisco, 1974.

Jackson, P., Introduction to Expert Systems, Addison-Wesley, 1986.

Jensen, P.A. and Barnes, J. W., Network Flow Programming, John Wiley & Sons, Inc. 1980.

Larkin, T.J., and Bomar, G.W., Climatic Atlas of Texas, Texas Department of Water Resources, LP-192, December 1983.

Lord, W.B., McGuire, T.R., and Wallace, M.G., Efficient and Equitable Solution of Indian Reserved Rights, NTIS PB89-214498/AS, Final Report, 22 p., June 1989

Loucks, Daniel P., Stedinger, Jery R., and Haith, Douglas A., Water Resources Systems Planning and Analysis, Prentice Hall Inc, Englewood Cliffs, 1981.

Maass, A., Hufschmidt, M., Dorfman, R., Thomas Jr., H., Marglin, S., and Fair, G., Design of Water Resource Systems, Harvard Univ. Press, Cambridge, 1962.

Maddaus, W.O. and McGill, J.M., Development And Application Of a Water Resource Allocation Model, Water Resources Research, Vol. 12, No. 4, p. 767-774, August 1976.

Nieuwkamer, R.L.J. and Winkelbauer, L., Linking of an Expert System with Numerical Models, Computer Techniques and Applications, Hydraulic Engineering Software IV, Computational Mechanics Publications, Southampton, England, and Elsevier Applied Science, London, England, p. 549-559, 1992.

Palmer, R.N. and Holmes, K.J., Operational Guidance During Droughts: Expert System Approach, J. Water Resources Planning and Management, Vol. 114, No. 6, November 1988.

Strzepek, K.M. and Chapra, S.C., Do the Right Thing, Civil Engineering, Vol. 60, No. 11, p. 55-56, November 1990.

Texas Department of Water Resources, Water Conveyance Pipeline Design Model PIPEX-I, Austin, Texas, September 1977.

Texas Water Development Board, Cost of Transporting Water By Pipeline, Report 42, Austin, Texas, March 1967.

Texas Water Development Board, Water For Texas Today And Tomorrow, 1992 Update Of The Texas Water Plan, Austin, Texas, 1992.

Texas Water Development Board, Water For Texas Today And Tomorrow, Austin, Texas, 1990.

Wright, J.R. and Buehler, K.A., GIS Proving Grounds for Water Resources Research, NTIS PB91-111526/AS, Technical Report 189, 67 p, June 1990.

APPENDIX A

TEXAS AUTOMATED ALLOCATION SYSTEM FILES, PROGRAMS, MACROS & COVERAGES

TABLE OF CONTENTS

TWDB - AAS FILES, PROGRAMS, MACROS, & COVERAGES

INPUT	PROCESS	OUTPUT
supsrc.dcs demand.dcs	jfb204f	parc.d00 parc.d01
supsrc.dcs demand.dcs	supsrc.aml demand.aml	supsrc coverage demand coverage
parc.d00 parc.d01	parc.aml	parc coverage
parc coverage	export.aml	arcparc.pdb
arcparc.pdb r10.p10.tkb	nexpert	nexparc.pdb
supsrc.dcs demand.dcs nexparc.pdb	jfb205b	jen1.dat
jen1.dat	jen1	jen1.out
jen1.out	jfb237	jen1.pdb
nexparc.pdb jen1.pdb	import.aml	parc coverage
parc coverage	show10.aml plot10.aml jfb206b	graphical display pl1.ai jfb206.ou1 jfb206.ou2
parc coverage	reset.aml	parc coverage

TWDB - AAS EXECUTION SEQUENCE

1. ESTABLISH ARCINFO COVERAGES

> > jfb204f
> [arc] &run supsrc.aml
> [arc] &run demand.aml
> [arc] &run parc.aml

2. EXPORT POTENTIAL ARC INFORMATION

> [arc] &run export.aml

3. RUN EXPERT SYSTEM WITH DESIRED RULES SET

> > nexpert
> Load r10p10.tkb
> Suggest volunteer
> OK & Knowcess
> (jfb205b)
> (jen1)
> (jfb237)

4. IMPORT MODIFIED POTENTIAL ARC INFORMATION

> [arc] &run import.aml

5. DISPLAY RESULTS

> [arc] arcplot show10.aml
> [arc} arcplot plot10.aml
> > jfb206b

6. RESTART SYSTEM

> [arc] &run reset.aml

Supply Attributes

#,slon,slat,selv,strns,sidn,scnty,scap,scip,sname

#	The sequence number of the potential source of supply.
slon	The longitude of the supply (decimal degrees).
slat	The latitude of the supply (decimal degrees).
selv	The elevation of the supply (ft. above msl).
strns	A transfer flag (0 indicates an original source, 'n' indicates the upstream source).
sidn	The TWDB identification number of the supply.
scnty	The identification number of the county in which the supply is located.
scap	The safe yield capacity of the supply (af/y).
scip	The cost in place of the water ($/af).
sname	The name of the supply source.

Demand Attributes

#,dlon,dlat,delv,dtrns,didn,dcnty,dcap,dcip,dname

#	The sequence number of the demand.
dlon	The longitude of the demand (decimal degrees).
dlat	The latitude of the demand (decimal degrees).
delv	The elevation of the demand (ft. above msl).
dtrns	A transfer flag (0 indicates a simple demander, 1 indicates a demander/wholesaler).
didn	The TWDB identification number for the demander.
dcnty	The identification number of the county in which the demander is located.
dcap	The demand amount (af/y).
dcip	The demand cost ($).
dname	The name of the demand.

supsrc.dcs

#,slon,slat,selv,strns,sidn,scnty,scap,scip,sname

```
1,-96.9973,28.1115,10.0,0,15,4,400,31,'GULF_COAST__ARANSAS'
2,-97.7400,28.4156,203.4,0,10,13,394,24,'CARRIZO-WILCOX__BEE'
3,-97.7400,28.4156,203.4,0,15,13,14577,31,'GULF_COAST__BEE'
4,-98.2185,27.0306,170.6,0,15,24,14577,31,'GULF_COAST__BROOKS'
5,-96.6162,28.4693,10.0,0,15,29,2940,31,'GULF_COAST__CALHOUN'
6,-96.6162,28.4693,10.0,0,99999,29,12600,10,'LOCAL__CALHOUN'
7,-96.6162,28.4693,10.0,0,99901,29,999999,999,'SHORTAGE__CALHOUN'
8,-97.3557,29.0818,210.0,0,15,62,15866,31,'GULF_COAST__DE_WITT'
9,-98.5079,27.6807,547.9,0,15,66,23970,31,'GULF_COAST__DUVAL'
10,-97.4244,28.6544,141.1,0,15,88,12809,31,'GULF_COAST__GOLIAD'
11,-97.4928,29.4563,298.6,0,10,89,19840,16,'CARRIZO-WILCOX__GONZ'
12,-97.4928,29.4563,298.6,0,15,89,2083,39,'GULF_COAST__GONZALES'
13,-97.4928,29.4563,298.6,0,24,89,6104,44,'QUEEN_CITY__GONZALES'
14,-97.4928,29.4563,298.6,0,27,89,16340,49,'SPARTA_SAND__GONZALE'
15,-97.4928,29.4563,298.6,0,99999,89,4200,10,'LOCAL__GONZALES'
16,-96.5766,28.9532,52.5,0,15,120,28343,39,'GULF_COAST__JACKSON'
17,-96.5766,28.9532,52.5,0,99901,120,999999,999,'SHORTAGE__JACKSON'
18,-98.0908,27.7300,213.3,0,15,125,11370,31,'GULF_COAST__JIM_WELL'
19,-98.0908,27.7300,213.3,0,99901,125,999999,999,'SHORTAGE__JIM_WELL'
20,-97.6654,26.9263,23.0,0,15,131,9550,31,'GULF_COAST__KENEDY'
21,-97.6991,27.4258,23.0,0,15,137,17088,31,'GULF_COAST__KLEBERG'
22,-96.9301,29.3840,249.3,0,15,143,38123,39,'GULF_COAST__LAVACA'
23,-98.1241,28.3524,150.9,0,10,149,2399,24,'CARRIZO-WILCOX__LIVE'
24,-98.1241,28.3524,150.9,0,15,149,5242,31,'GULF_COAST__LIVE_OAK'
25,-98.1241,28.3524,150.9,0,99999,149,760,10,'LOCAL__LIVE_OAK'
26,-98.1241,28.3524,150.9,0,99901,149,999999,999,'SHORTAGE__LIVE_OAK'
27,-98.5675,28.3512,295.3,0,10,156,7909,24,'CARRIZO-WILCOX__MCMU'
28,-98.5675,28.3512,295.3,0,15,156,1838,31,'GULF_COAST__MCMULLEN'
29,-98.5675,28.3512,295.3,0,24,156,1105,44,'QUEEN_CITY__MCMULLEN'
30,-98.5675,28.3512,295.3,0,27,156,600,49,'SPARTA_SAND__MCMULLE'
31,-98.5675,28.3512,295.3,0,99901,156,999999,999,'SHORTAGE__MCMULLEN'
32,-97.5391,27.7388,19.7,0,15,178,3254,31,'GULF_COAST__NUECES'
33,-97.5391,27.7388,19.7,0,99999,178,950,10,'LOCAL__NUECES'
34,-97.5391,27.7388,19.7,0,99901,178,999999,999,'SHORTAGE__NUECES'
35,-97.1611,28.3177,45.9,0,15,196,7768,31,'GULF_COAST__REFUGIO'
36,-97.5206,28.0083,55.8,0,15,205,5228,31,'GULF_COAST__SAN_PATR'
37,-97.5206,28.0083,55.8,0,99901,205,999999,999,'SHORTAGE__SAN_PATR'
38,-96.9715,28.7962,88.6,0,15,235,41130,39,'GULF_COAST__VICTORIA'
39,-96.5667,28.8833,55.0,0,16010,120,75000,43,'TEXANA'
40,-97.3000,29.1333,100.0,0,18092,62,52000,49,'CUERO_I&II'
41,-97.1667,28.9667,142.7,0,18501,62,79000,40,'GUADALUPE'
42,-97.1666,28.7500,35.0,0,18100,235,12500,59,'COLETO_CR'
43,-97.1667,28.5833,110.0,0,19500,235,25000,40,'SAN_ANTONIO'
44,-97.8667,28.0500,94.0,0,21101,210,230549,43,'CHOKE-CORPUS_w_bay'
```

demand.dcs

#,dlon,dlat,delv,dtrns,didn,dcnty,dcap,dcip,dname

```
1,-97.0515,28.0407,9.8,0,511,4,2324,0,'ROCKPORT'
2,-96.9973,28.1115,0,0,757,4,4426,0,'CO_ARANSAS'
3,-96.9973,28.1115,0,0,992,4,554,0,'CA_ARANSAS'
4,-96.9973,28.1115,0,0,993,4,107,0,'CB_ARANSAS'
5,-97.7491,28.4056,226.4,0,45,13,3730,0,'BEEVILLE'
6,-97.74,28.4156,203.4,0,757,13,2921,0,'CO_BEE'
7,-97.74,28.4156,203.4,0,992,13,5,0,'CA_BEE'
8,-97.74,28.4156,203.4,0,993,13,2590,0,'CB_BEE'
9,-98.145,27.2231,114.8,0,197,24,1372,0,'FALFURRIAS'
10,-98.2185,27.0306,170.6,0,757,24,769,0,'CO_BROOKS'
11,-98.2185,27.0306,170.6,0,992,24,18,0,'CA_BROOKS'
12,-98.2185,27.0306,170.6,0,993,24,1690,0,'CB_BROOKS'
13,-96.5500,28.6667,6.0,0,474,29,237,0,'POINT_COMFORT'
14,-96.6212,28.6147,0,0,479,29,3213,0,'PORT_LAVACA'
15,-96.7152,28.4139,6.6,0,546,29,398,0,'SEADRIFT'
16,-96.6162,28.4693,0,0,757,29,2219,0,'CO_CALHOUN'
17,-96.6162,28.4693,0,0,992,29,94914,0,'CA_CALHOUN'
18,-96.6162,28.4693,0,0,993,29,23235,0,'CB_CALHOUN'
19,-97.2874,29.0941,187,0,147,62,1831,0,'CUERO'
20,-97.5041,28.9829,288.7,0,671,62,535,0,'YORKTOWN'
21,-97.1465,29.2928,324.8,0,670,62,550,0,'YOAKUM'
22,-97.3557,29.0818,210,0,757,62,1182,0,'CO_DE_WITT'
23,-97.3557,29.0818,210,0,992,62,7212,0,'CA_DE_WITT'
24,-97.3557,29.0818,210,0,993,62,3070,0,'CB_DE_WITT'
25,-98.4092,27.5977,380.6,0,50,66,747,0,'BENAVIDES'
26,-98.6182,27.8814,547.9,0,218,66,1227,0,'FREER'
27,-98.2383,27.7592,298.6,0,534,66,1294,0,'SAN_DIEGO'
28,-98.5079,27.6807,547.9,0,757,66,448,0,'CO_DUVAL'
29,-98.5079,27.6807,547.9,0,993,66,5506,0,'CB_DUVAL'
30,-97.3916,28.6697,173.9,0,240,88,671,0,'GOLIAD'
31,-97.4244,28.6544,141.1,0,757,88,921,0,'CO_GOLIAD'
32,-97.4244,28.6544,141.1,0,992,88,16000,0,'CA_GOLIAD'
33,-97.4244,28.6544,141.1,0,993,88,1934,0,'CB_GOLIAD'
34,-97.4475,29.5086,298.6,0,241,89,2932,0,'GONZALES'
35,-97.7619,29.2693,397,0,432,89,653,0,'NIXON'
36,-97.4928,29.4563,298.6,0,757,89,2593,0,'CO_GONZALES'
37,-97.4928,29.4563,298.6,0,992,89,2672,0,'CA_GONZALES'
38,-97.4928,29.4563,298.6,0,993,89,6775,0,'CB_GONZALES'
39,-96.6473,28.974,62.3,0,183,120,1573,0,'EDNA'
40,-96.5114,29.0421,59.1,0,228,120,418,0,'GANADO'
41,-96.5766,28.9532,52.5,0,757,120,1338,0,'CO_JACKSON'
42,-96.5766,28.9532,52.5,0,992,120,66,0,'CA_JACKSON'
43,-96.5766,28.9532,52.5,0,993,120,61120,0,'CB_JACKSON'
44,-98.0655,27.7552,203.4,0,6,125,9410,0,'ALICE'
45,-97.9389,27.9558,193.6,0,444,125,403,0,'ORANGE_GROVE'
46,-98.1241,27.3578,147.6,0,486,125,1351,0,'PREMONT'
47,-98.0908,27.73,213.3,0,757,125,2560,0,'CO_JIM_WELLS'
48,-98.0908,27.73,213.3,0,992,125,347,0,'CA_JIM_WELLS'
49,-98.0908,27.73,213.3,0,993,125,4652,0,'CB_JIM_WELLS'
50,-97.225,27.7917,0,0,542,131,14,0,'SARITA'
51,-97.6654,26.9263,23,0,757,131,49,0,'CO_KENEDY'
52,-97.6654,26.9263,23,0,993,131,1821,0,'CB_KENEDY'
53,-97.8607,27.5089,59.1,0,323,137,9179,0,'KINGSVILLE'
```

demand.dcs (continued)

```
54,-97.6991,27.4258,23,0,757,137,2028,0,'CO_KLEBERG'
55,-97.6991,27.4258,23,0,992,137,2574,0,'CA_KLEBERG'
56,-97.6991,27.4258,23,0,993,137,2612,0,'CB_KLEBERG'
57,-96.9417,29.445,229.7,0,259,143,831,0,'HALLETTSVILLE'
58,-97.1718,29.432,347.8,0,557,143,746,0,'SHINER'
59,-97.1465,29.2928,324.8,0,670,143,915,0,'YOAKUM'
60,-96.9302,29.3841,249.3,0,757,143,1865,0,'CO_LAVACA'
61,-96.9302,29.3841,249.3,0,992,143,6933,0,'CA_LAVACA'
62,-96.9302,29.3841,249.3,0,993,143,15216,0,'CB_LAVACA'
63,-98.1177,28.3303,167.3,0,234,149,592,0,'GEORGE_WEST'
64,-98.1779,28.4656,137.8,0,604,149,485,0,'THREE_RIVERS'
65,-98.1241,28.3524,150.9,0,757,149,795,0,'CO_LIVE_OAK'
66,-98.1241,28.3524,150.9,0,992,149,16113,0,'CA_LIVE_OAK'
67,-98.1241,28.3524,150.9,0,993,149,4960,0,'CB_LIVE_OAK'
68,-98.55,28.45,272.3,0,606,156,76,0,'TILDEN'
69,-98.5675,28.3512,295.3,0,757,156,155,0,'CO_MCMULLEN'
70,-98.5675,28.3512,295.3,0,993,156,4626,0,'CB_MCMULLEN'
71,-97.7976,27.5848,55.8,0,59,178,848,0,'BISHOP'
72,-97.2928,27.7057,9.8,0,135,178,119046,0,'CORPUS_CHRISTI'
73,-97.0826,27.8307,0,0,475,178,2161,0,'PORT_ARANSAS'
74,-97.6607,27.7992,68.9,0,508,178,2820,0,'ROBSTOWN'
75,-97.5391,27.7388,19.7,0,757,178,3834,0,'CO_NUECES'
76,-97.5391,27.7388,19.7,0,992,178,54448,0,'CA_NUECES'
77,-97.5391,27.7388,19.7,0,993,178,4354,0,'CB_NUECES'
78,-97.275,28.3071,59.1,0,497,196,439,0,'REFUGIO'
79,-97.3248,28.2375,49.2,0,665,196,278,0,'WOODSBORO'
80,-97.1611,28.3178,45.9,0,757,196,406,0,'CO_REFUGIO'
81,-97.1611,28.3178,45.9,0,993,196,940,0,'CB_REFUGIO'
82,-97.8245,28.0934,150.9,0,392,205,1262,0,'MATHIS'
83,-97.1087,27.8881,0,0,23,205,2010,0,'ARANSAS_PASS'
84,-97.2908,27.9221,26.3,0,251,205,725,0,'GREGORY'
85,-97.2001,27.87,19.7,0,296,205,1789,0,'INGLESIDE'
86,-97.5868,27.9455,72.2,0,437,205,636,0,'ODEM'
87,-97.3269,27.8789,29.5,0,478,205,2980,0,'PORTLAND'
88,-97.5096,28.0335,45.9,0,562,205,1416,0,'SINTON'
89,-97.3908,27.9811,49.2,0,592,205,827,0,'TAFT'
90,-97.4053,27.9723,49.2,0,593,205,407,0,'TAFT_SOUTHWEST'
91,-97.5206,28.0083,55.8,0,757,205,3540,0,'CO_SAN_PATRICIO'
92,-97.5206,28.0083,55.8,0,992,205,28008,0,'CA_SAN_PATRICIO'
93,-97.5206,28.0083,55.8,0,993,205,4672,0,'CB_SAN_PATRICIO'
94,-96.902,28.6505,55.8,0,61,235,515,0,'BLOOMINGTON'
95,-96.9829,28.8242,98.4,0,624,235,16243,0,'VICTORIA'
96,-96.9715,28.7962,88.6,0,757,235,4125,0,'CO_VICTORIA'
97,-96.9715,28.7962,88.6,0,992,235,79827,0,'CA_VICTORIA'
98,-96.9715,28.7962,88.6,0,993,235,14985,0,'CB_VICTORIA'
```

supsrc.aml

```
/*  supsrc.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/**********************************************************************
/*
/* USAGE: This aml creates the SUPSRC coverage and loads it with
/*        data from the ASCII file SUPSRC.DCS
/*
/*
/*
/**********************************************************************
/*
/*  INVOKED BY: [ARC] &run supsrc.aml
/*  RELATED COVERAGES:  supsrc
/*  RELATED FILES:  supsrc.dcs
/*
/**********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> SUPSRC Coverage creation in progress'
&type '<> Processing'
&MESSAGES &OFF &ALL
&SEVERITY &WARNING &IGNORE
&SEVERITY &ERROR &IGNORE
KILL SUPSRC ALL
TABLES
KILL SUPSRC
Q STOP
GENERATE SUPSRC
INPUT SUPSRC.DCS
POINT
QUIT
BUILD SUPSRC POINT
TABLES
DEFINE SUPSRC
SUPSRC_ID
10
10
I
SLON
10
10
N
4
SLAT
10
10
N
4
SELV
10
10
```

supsrc.aml (continued)

```
N
1
STRNS
10
10
I
SIDN
10
10
I
SCNTY
10
10
I
SCAP
10
10
I
SCIP
10
10
I
SNAME
20
20
C
~
ADD FROM SUPSRC.DCS
Q STOP
&type '<> Complete.     Created coverage:  SUPSRC'
&type ' '
QUIT
```

demand.aml

```
/*  demand.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/**********************************************************************
/*
/* USAGE: This aml creates the DEMAND coverage and loads it with
/*        data from the ASCII file DEMAND.DCS
/*
/*
/*
/**********************************************************************
/*
/*  INVOKED BY: [ARC] &run demand.aml
/*  RELATED COVERAGES:  demand
/*  RELATED FILES:  demand.dcs
/*
/**********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> DEMAND Coverage creation in progress'
&type '<> Processing'
&MESSAGES &OFF &ALL
&SEVERITY &WARNING &IGNORE
&SEVERITY &ERROR &IGNORE
KILL DEMAND ALL
TABLES
KILL DEMAND
Q STOP
GENERATE DEMAND
INPUT DEMAND.DCS
POINT
QUIT
BUILD DEMAND POINT
TABLES
DEFINE DEMAND
DEMAND_ID
10
10
I
DLON
10
10
N
4
DLAT
10
10
N
4
DELV
10
10
N
```

demand.aml (continued)

```
1
DTRNS
10
10
I
DIDN
10
10
I
DCNTY
10
10
I
DCAP
10
10
I
DCIP
10
10
I
DNAME
20
20
C
~
ADD FROM DEMAND.DCS
Q STOP
&type '<> Complete.     Created coverage:  DEMAND'
&type ' '
QUIT
```

jfb204f.for

```
C     PROGRAM JFB204F.FOR  SUPSRC.DCS + DEMAND.DCS ---> PARC.D00
C                                                       PARC.D01
C                                                       (JEN1.DAT)
C     USING JB224   COST=ANNUALIZED PROJ + PWR + O&M
      CHARACTER*20 SNAME
      CHARACTER*20 DNAME
C     CHARACTER*20 TNAME
      DIMENSION ISCNTY(199),SLAT(199),SLON(199),SELV(199),ISCAP(199),
     .          ISCIP(199),SNAME(199),ISTRNS(199),ISIDN(199)
      DIMENSION IDCNTY(199),DLAT(199),DLON(199),DELV(199),IDCAP(199),
     .          IDCIP(199),DNAME(199),IDTRNS(199),IDIDN(199)
      DIMENSION ITCNTY(199),TLAT(199),TLON(199),TELV(199),ITCAP(199),
     .ITSU(199),ITCIP(199),          ITTRNS(199),ITIDN(199)
      OPEN(1,FILE='supsrc.dcs')
      OPEN(2,FILE='demand.dcs')
      OPEN(4,FILE='jen1.dat')
      OPEN(9,FILE='parc.d00')
      OPEN(10,FILE='parc.d01')
      WRITE(*,10)
10    FORMAT(1X,   '******************************',/,
     .       1X,   '* PROGRAM JFB204F.FOR        *',/,
     .       1X,   '* PARC    GENERATOR          *',/,
     .       1X,   '* VERSION 06/01/94           *',/,
     .       1X,   '******************************',///)
C
      NL=0
      ISUM=0
      CALL GETDAT(IYR,IMON,IDAY)
      CALL GETTIM(IHR,IMIN,ISEC,I100TH)
      IYR=IYR-1900
      WRITE(4,20) IHR,IMIN,ISEC,IMON,IDAY,IYR
20    FORMAT(1X,'JFB204F.FOR',2X,
     . I2.2,':',I2.2,':',I2.2,1X,I2.2,'/',I2.2,'/',I2.2)
C
C
      NS=0
      ISTOT=0
100   CONTINUE
      NS=NS+1
      IF(NS.GT.199)GO TO 190
      READ(1,*,ERR=190,END=190)K,SLON(NS),SLAT(NS),SELV(NS),
     .  ISTRNS(NS),ISIDN(NS),ISCNTY(NS),ISCAP(NS),ISCIP(NS),SNAME(NS)
      IF(ISTRNS(NS).EQ.0)ISTOT=ISTOT+ISCAP(NS)
      GO TO 100
190   CONTINUE
      NS=NS-1
C
      ND=0
      IDTOT=0
200   CONTINUE
      ND=ND+1
      IF(ND.GT.199)GO TO 290
      READ(2,*,ERR=290,END=290)K,DLON(ND),DLAT(ND),DELV(ND),
     .IDTRNS(ND),IDIDN(ND),IDCNTY(ND),IDCAP(ND),IDCIP(ND),DNAME(ND)
      IDTOT=IDTOT+IDCAP(ND)
      GO TO 200
```

jfb204f.for (continued)

```
290    CONTINUE
       ND=ND-1
C
300    CONTINUE
       NT=0
       DO 390 ID=1,ND
       IF(IDTRNS(ID).EQ.0)GO TO 390
C
       NT=NT+1
       IF(NT.GT.199)GO TO 390
       ITSU(NT)=ID
CJFB   ITIDN(NT)=IDIDN(ID)
       ITIDN(NT)=99999
CJFB   ITTRNS(NT)=IDTRNS(ID)
       ITTRNS(NT)=0
       ITCNTY(NT)=IDCNTY(ID)
       TLAT(NT)=DLAT(ID)
       TLON(NT)=DLON(ID)
       TELV(NT)=DELV(ID)
CJFB   ITCAP(NT)=IDCAP(ID)
       ITCAP(NT)=999999
       ITCIP(NT)=IDCIP(ID)
C      TNAME(NT)=DNAME(ID)
C
390    CONTINUE
C
400    CONTINUE
       IF(IDTOT.GT.ISTOT)GO TO 850
       N=NS+ND+1
       WRITE(4,410)N,NS,ND,NT
410    FORMAT(4I10,5X,'(N NS ND NT   #NODES)')
C
C      SET UP SUPPLY NODES
       DO 450 I=1,NS
       IX1=I
       IX2=0
       IX3=0
       IX4=0
       WRITE(4,420)IX1,IX2,IX3,IX4,SNAME(I)
420    FORMAT(4I10,A20)
450    CONTINUE
C      SET UP DEMAND NODES
       DO 480 I=1,ND
       IX1=NS+I
       IX2=-(IDCAP(I))
       IX3=0
       IX4=0
       WRITE(4,420)IX1,IX2,IX3,IX4,DNAME(I)
       ISUM=ISUM-IX2
480    CONTINUE
C
C      SET UP MASTER SUPPLY NODE
       IX1=ND+NS+1
       IX2=ISUM
       IX3=0
       IX4=0
```

jfb204f.for (continued)

```
      WRITE(4,495)IX1,IX2,IX3,IX4
495   FORMAT(4I10,'MASTER SUPPLY NODE  ')
      WRITE(4,496)
496   FORMAT(50X,'(BLANK)')
C
500   CONTINUE
C     SET UP SUPPLY NODE CAPACITIES & COSTS
      DO 550 I=1,NS
      IF(ISTRNS(I).EQ.0)GO TO 504
      J=ISTRNS(I)
      IX1=ISTRNS(I)
      IX4=ISCAP(I)
      IX5=ISCIP(I)-ISCIP(J)
      IF(IX5.LT.0)IX5=0
      GO TO 506
504   CONTINUE
      IX1=NS+ND+1
      IX4=ISCAP(I)
      IX5=ISCIP(I)
506   CONTINUE
      IX2=I
      IX3=0
      NL=NL+1
      WRITE(4,510)IX1,IX2,IX3,IX4,IX5
510   FORMAT(5I10)
550   CONTINUE
C
600   CONTINUE
C     SET UP SUPPLY ARCS & COSTS
C     NSPJ=DEMAND NODE ID  (JEN1.FOR SCHEME)
      ILOWB=0
      IUPPB=999999
      IFLOW=0
C                    NOTE:  IFEAS=0  FEASIBLE
C                           IFEAS>0  NOT FEASIBLE
      IFEAS=0
C
C
      DO 690 I=1,NS
      DO 680 J=1,ND
      NSPJ=NS+J
C
      SLATI=SLAT(I)
      SLONI=SLON(I)
      DLATJ=DLAT(J)
      DLONJ=DLON(J)
      IF(ABS(DLATJ-SLATI).LT.0.0001.AND.ABS(DLONJ-SLONI).LT.0.0001)THEN
         CPTG=0.
         PLEN=0.
         SLONI=SLONI+0.0001
         GO TO 670
         END IF
      CALL JB6(SLATI,SLONI,DLATJ,DLONJ,D)
      PLEN=D
C     QMGD=IDCAP(J)/1120.162
      QMGD=AMIN0(ISCAP(I),IDCAP(J))/1120.162
```

```
      ELEV1=SELV(I)
      ELEV2=DELV(J)
      XLMI=D+0.1
      CALL JB224(QMGD,ELEV1,ELEV2,XLMI,CPTG,IERR)
670   CONTINUE
      ICOST=INT(CPTG*1000./3.069+0.5)
C
C
C
C
C
      NL=NL+1
      NLMNS=NL-NS
      WRITE(4,510)I,NSPJ,ILOWB,IUPPB,ICOST
      WRITE(9,674)NLMNS,SLONI,SLATI,DLONJ,DLATJ
674   FORMAT(I10,/,F10.4,',',F10.4,/,F10.4,',',F10.4,/,'END')
      WRITE(10,675)NLMNS,SLON(I),SLAT(I),DLON(J),DLAT(J),SELV(I),DELV(J)
     .,ISTRNS(I),IDTRNS(J),ISIDN(I),IDIDN(J),ISCNTY(I),IDCNTY(J),
     .ISCAP(I),IDCAP(J),ISCIP(I),IDCIP(J),
     .PLEN,ILOWB,ILOWB,IUPPB,IUPPB,ICOST,ICOST,IFLOW,IFLOW,IFEAS,I,J
675   FORMAT(I10,4F10.4,2F10.1,10I10,F10.1,9I10,2X,'S',I3.3,'D',I3.3)
680   CONTINUE
690   CONTINUE
C
700   CONTINUE
C     SET UP TRANSFER  SUPPLY ARCS
C
      ILOWB=0
      IUPPB=999999
      IFLOW=0
      IFEAS=0
C
      DO 790 J=1,NT
      NSPITSU=NS+ITSU(J)
      TLATJ=TLAT(J)
      TLONJ=TLON(J)
      DO 780 K=1,ND
      NSPK=NS+K
C     TRANSFER TO ONESELF UNNECESSARY
      IF(NSPITSU.EQ.NSPK)GO TO 780
C
      DLATK=DLAT(K)
      DLONK=DLON(K)
      IF(ABS(DLATK-TLATJ).LT.0.0001.AND.ABS(DLONK-TLONJ).LT.0.0001)THEN
         CPTG=0.
         PLEN=0.
         TLONJ=TLONJ+0.0001
         GO TO 770
         END IF
      CALL JB6(TLATJ,TLONJ,DLATK,DLONK,D)
      PLEN=D
      QMGD=IDCAP(K)/1120.162
      ELEV1=TELV(J)
      ELEV2=DELV(K)
      XLMI=D+0.1
      CALL JB224(QMGD,ELEV1,ELEV2,XLMI,CPTG,IERR)
```

```
770    CONTINUE
       ICOST=INT(CPTG*1000./3.069+0.5)
C
C
C
C
C
       NL=NL+1
       NLMNS=NL-NS
       WRITE(4,510)NSPITSU,NSPK,ILOWB,IUPPB,ICOST
       WRITE(9,674)NLMNS,TLONJ,TLATJ,DLONK,DLATK
       WRITE(10,775)NLMNS,TLON(J),TLAT(J),DLON(K),DLAT(K),TELV(J),DELV(K)
      .,ITTRNS(J),IDTRNS(K),ITIDN(J),IDIDN(K),ITCNTY(J),IDCNTY(K),
      .ITCAP(J),IDCAP(K),ITCIP(J),IDCIP(K),
      .PLEN,ILOWB,ILOWB,IUPPB,IUPPB,ICOST,ICOST,IFLOW,IFLOW,IFEAS,
      .ITSU(J),K
775    FORMAT(I10,4F10.4,2F10.1,10I10,F10.1,9I10,2X,'D',I3.3,'D',I3.3)
780    CONTINUE
790    CONTINUE
C
       WRITE(4,496)
C
       WRITE(10,795)
795    FORMAT('END')
800    CONTINUE
       WRITE(*,899)NS,N,ND,NL,NT
899    FORMAT(5X,5X,'FROM INPUT',  5X,5X,'CREATED OUTPUT',/,
      .        5X,5X,'----------',  5X,5X,'--------------',/,
      .        5X,5X,'SUPSRC.DCS',  5X,5X,'     PARC.D00',/,
      .        5X,5X,'DEMAND.DCS',  5X,5X,'     PARC.D01',/,
      .        5X,5X,'          ',  5X,5X,'     JEN1.DAT',//,
      .        5X,I5,'    SUPSRC',  5X,I5,'        NODES',/,
      .        5X,I5,'    DEMAND',  5X,I5,'         ARCS',/,
      .        5X,I5,'    TRANSF')
       GO TO 900
C
850    CONTINUE
       WRITE(*,851) IDTOT,ISTOT
851    FORMAT(1X,'850 ERROR...DEMAND GT SUPPLY',
      .        1X,I10,5X,I10)
C
900    CONTINUE
       CLOSE(1)
       CLOSE(2)
       CLOSE(3)
       CLOSE(9)
       CLOSE(10)
       END
       SUBROUTINE JB6(AL1,AM1,AL2,AM2,D)
       AL1R=AL1/57.29578
       AL2R=AL2/57.29578
       AM1R=AM1/57.29578
       AM2R=AM2/57.29578
       X=SIN(AL1R)*SIN(AL2R)+COS(AL1R)*COS(AL2R)*COS(AM2R-AM1R)
       D=ARCOS(X)
       D=D*57.29578
```

jfb204f.for (continued)

```
      D=D*60.*1.1507803
      RETURN
      END
      FUNCTION ARCOS(X)
      XXX = X
      IF(XXX.EQ.-1.) XXX = -.99999
      ARG = (1.-X)/(1.+XXX)
      TARG = SQRT(ARG)
      Y = 2.0*ATAN(TARG)
      ARCOS = Y
      RETURN
      END
      SUBROUTINE JB224(QMGD,ELEV1,ELEV2,XLMI,CPTG,IERR)
      IPR=0
      QAF=1120.162*QMGD
      QCFS=QMGD*1.547228
      C  =18700.*(QAF**0.600)
C
      AI=0.04
      N =40
      PAF= AI/(1.-(1.+AI)**(-N))
C
      ACC=PAF*C
      IF(IPR.GT.0)WRITE(*,10)C,ACC
10    FORMAT(1X,'TOTAL PROJECT COST             ',F15.2,/,
     .       1X,'ANNUALIZED (40YR 4%)           ',F15.2)
C
      CALL PIPED(QMGD,ELEV1,ELEV2,XLMI,HLTPS,DIN,IERR)
      DFT=DIN/12.
      VFPS=QCFS/((3.14159/4.)*DFT*DFT)
      IF(IPR.GT.0)WRITE(*,20)DIN,VFPS,HLTPS
20    FORMAT(1X,'PIPE DIAMETER (IN)             ',F15.2,/,
     .       1X,'WATER VELOCITY (FPS)           ',F15.2,/,
     .       1X,'HEAD LOSS (FT)                 ',F15.2)
C
      SHFT=ELEV2-ELEV1
C     IF(SHFT.LT.0)SHFT=0.
      HFT=SHFT+HLTPS
C
      PSOMR=0.534*QCFS*HFT
      IF(PSOMR.LT.0.)PSOMR=0.
      PIOMR=4.*DIN*XLMI
C
      UKWHR=0.10
      POWRC=0.084*UKWHR*QCFS*HFT
      POWRC=POWRC*24.*365.
      IF(POWRC.LT.0.)POWRC=0.
C
      TAC=ACC+PSOMR+PIOMR+POWRC
      IF(IPR.GT.0)WRITE(*,30)PSOMR,PIOMR,POWRC
30    FORMAT(1X,'PUMP STATION O&M               ',F15.2,/,
     .       1X,'PIPELINE O&M                   ',F15.2,/,
     .       1X,'POWER COST                     ',F15.2)
C
      QTG=QMGD*365.*1000.
      CPTG=TAC/QTG
```

jfb204f.for (continued)

```
      IF(IPR.GT.0)WRITE(*,40)TAC,QTG,CPTG
40    FORMAT(1X,'TOTAL ANNUAL COSTS          ',F15.2,/,
     .        1X,'QUANTITY IN THOUSAND GAL.  ',F15.2,/,
     .        1X,'COST PER THOUSAND GAL.     ',F15.2)
C
CJB   WRITE(*,50)QMGD,ELEV1,ELEV2,XLMI,CPTG,IERR
50    FORMAT(1X,'JB224:      ',4F10.2,F10.5,I5)
C     IF(IPR.EQ.1)READ(*,51)IANS
51    FORMAT(I1)
      RETURN
      END
      SUBROUTINE PIPED(QMGD,ELEV1,ELEV2,XLMI,HLTPS,DIN,IERR)
C
C
      INTEGER ITYPE, MTYPE, NTYPE, NUNIT, IPMAX, IDMAX, ITG1, N, ND
      INTEGER I,IANS,IFLAG(20), ICMAX ,IERR,IPR
C
      REAL XLEN, Q, TG1, TG2, PMAX(4), EL1, EL2
C     REAL CSC(28)
C     REAL DIP(13)
C     REAL PVC(6)
      REAL SP(38)
      REAL DIA(38), PI, AHL, C(20), D, DIAM(20), ERH
      REAL HGL1, HGL2, HL(20), HLPT(20), RH(20), V, XLENT, XL, VEL(20)
      REAL ELEV1,ELEV2,XLMI,QMGD,HLTPS,DIN,AAHL
C
      CHARACTER PTYPE*28, FUNIT*3, STYPE*12, DTYPE*20
      CHARACTER VUNIT*3, LUNIT*11
C
      PARAMETER ( PI = 3.14159)
C
      LOGICAL SMALL
C
C
C**** ASSIGN NOMINAL INSIDE DIAMETERS FOR EACH PIPE MATERIAL ****
C
C
C           DATA CSC /16,18,20,24,27,30,33,36,39,42,45,
C    $          48,54,60,66,72,78,84,90,96,102,
C    $          108,114,120,126,132,138,144/
C           DATA DIP /12,14,16,18,20,24,30,36,42,48,54,
C    $          60,64/
C           DATA PVC /12,14,16,18,20,24/
         DATA SP /12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,45,
     $          48,51,54,57,60,63,66,69,72,75,78,81,84,87,90,93,96,102,
     $          108,114,120/

C
C**** SUGGESTED UPPER LIMITS TO PIPE INTERNAL WORKING PRESSURE, FT ***
C****  PMAX(1) - CONC. STEEL CYL
C****      (2) - DUCTILE IRON
C****      (3) - PVC
C****      (4) - STEEL PIPE
C
         DATA PMAX /576,576,288,461/
```

jfb204f.for (continued)

```
C
C**** ASSIGNMENT STATEMENTS ****
C
C****  INITIALIZE ARRAY OF OUTPUT MESSAGE FLAGS
C
      IPR=0
      IERR=0
      DIN=0.
      HLTPS=0.
   10 DO 15 I = 1,4
      IFLAG(I) = 0
   15 CONTINUE
C
C
C     MTYPE=1
C     PTYPE = 'CONCRETE STEEL CYLINDER PIPE'
C     IDMAX = 28
C     DO 45 N = 1,IDMAX
C     DIA(N) = CSC(N)
C  45 CONTINUE
C
C
C     MTYPE=2
C     PTYPE = 'DUCTILE IRON PIPE'
C     IDMAX = 13
C     DO 46 N = 1,IDMAX
C     DIA(N) = DIP(N)
C  46 CONTINUE
C
C
C     MTYPE=3
C     PTYPE = 'POLYVINYL CHLORIDE PIPE'
C     IDMAX = 6
C     DO 47 N = 1,IDMAX
C     DIA(N) = PVC(N)
C  47 CONTINUE
C
      MTYPE=4
      PTYPE = 'STEEL PIPE'
      IDMAX = 38
      DO 48 N = 1,IDMAX
      DIA(N) = SP(N)
   48 CONTINUE
C
C
C
C**** RETRIEVE C-FACTORS
C
C
      ICMAX=1
      C(1)=100.
C
C
```

jfb204f.for (continued)

```
C**** RETRIEVE PIPE LENGTH ****
C
      LUNIT = 'MILES'
C
      XLEN=XLMI
C
C**** CONVERT LENGTH TO LINEAR FT.
C
      XL =XLEN * 5280.
C
C**** PUT LENGTH IN THOUSAND FT UNITS FOR OUTPUT
C
      XLENT = XL/1000.
C
C**** RETRIEVE FLOW UNITS AND FLOW RATE ****
C
C
C**** READ UNIT TYPE, AND ASSIGN OUTPUT LABELS ****
C
      NUNIT=3
C
      FUNIT = 'MGD'
      VUNIT = 'FPS'
C
C
C
C
      Q=QMGD
C
C
C**** RETRIEVE SUPPLY CONDITIONS ****
C
C
C
C**** READ TYPE OF SUPPLY FACILITY, AND ASSIGN OUTPUT LABEL ***
C
      ITYPE=1
      STYPE = 'PUMP STATION'
C
C**** GET WATER SUPPLY HEAD ****
C
CJFB   ASSUMPTIONS
C
      AAHL=1.1
      TG2=10.
      TG1=TG2+ELEV2+AAHL*XLENT-ELEV1
      IF(TG1.LT.0.)TG1=0.
C
CJFB  ASSUMPTIONS
C
      IF(PMAX(MTYPE) .LT. TG1) THEN
          IPMAX = NINT( PMAX(MTYPE)/ 2.307)
      ITG1  = NINT( TG1/ 2.307)
      IF(IPR.GT.0)WRITE(*,84) TG1,ITG1,IPMAX,PTYPE
```

jfb204f.for (continued)

```
84     FORMAT(//2X,'WARNING:'//
     $  2x,'THE INPUT WATER SUPPLY HEAD IS ',F5.1,' FT (',I3,' psi)'/
     $  2x,'MANUFACTURER DATA SUGGESTS THAT TYPICAL DESIGN PRESSURE'/
     $  2X,'DOES NOT EXCEED ',I3,' PSI, WITH ',A28,'.'/
     $  2X,'CONSULT MANUFACTURER TO VERIFY ALLOWABLE INTERNAL'/
     $  2X,'PIPELINE PRESSURE.')
      IERR=1
      ENDIF
C
C**** GET PIPE ELEVATION ****
C
C  90 FORMAT(//2X,'ENTER PIPE ELEVATION, FT: --->'\)
      EL1=ELEV1
C
C
C**** RETRIEVE WATER DISCHARGE CONDITIONS ****
C
C


C**** READ DISCHARGE FACILITY TYPE, AND ASSIGN OUTPUT LABELS ***
C
      NTYPE=2
      DTYPE = 'IN-LINE BOOSTER PUMP'
C
C
C**** GET MINIMUM REQUIRED DISCHARGE HEAD ****
C

C 120 FORMAT(//2X,'ENTER REQUIRED DISCHARGE HEAD, FT: --->'\)
CJFB  TG2=10.
C
C**** GET PIPE ELEVATION AT DISCHARGE ***
C
      EL2=ELEV2
C
C
C     *** CALCULATE THE TOTAL HYDRAULIC GRADE AT SUPPLY AND DISCHARGE,
C     *** AND DETERMINE THE ALLOWABLE HEADLOSS IN THE PIPELINE.
C
      HGL1 = TG1 + EL1
      HGL2 = TG2 + EL2
       AHL = HGL1 - HGL2
C
C     **** TEST FOR ADEQUATE HEADLOSS ALLOWANCE
C
      IF (AHL .LE. (XL * 0.001) ) THEN
          IF(IPR.GT.0)WRITE (*,135) TG2
  135     FORMAT (//10X,'********** WARNING:'//
     $           10X,'INPUT DATA INDICATES THAT ONLY LOW HEADLOSSES'
     $           ' ARE ALLOWED.'//
     $           10X,'HEADLOSSES INCURRED AT A RATE GREATER THAN'
     $           ' 1 FT PER 1000 FT'//
     $           10X,'WILL RESULT IN DISCHARGE HEAD BELOW',F9.1,' FT'
```

```
     $           ', AS SPECIFIED.'//
     $           10X,'DESIGN UNDER THESE CONDITIONS MAY RESULT IN'
     $           ' LARGE DIAMETER PIPE'//
     $           10X,' AND LOW WATER VELOCITY.  CHECK INPUT DATA.')
C
      IF(IPR.GT.0)READ(*,141)IANS
      IERR=IERR+2
      ENDIF
C
C*** PRINT DATA SUMMARY TO SCREEN AND TO HARD COPY ***
C
C
          IF(IPR.GT.0)WRITE(*,140) PTYPE, XLEN, LUNIT, STYPE, Q,
     $ FUNIT,TG1,EL1,HGL1, DTYPE, TG2, EL2, HGL2
  140 FORMAT(1H1//25X,'INPUT DATA SUMMARY'//
     $        10X,'PIPE MATERIAL: ',A23,//
     $        10X,'PIPE LENGTH = ',F9.2,1X,A11,//
     $        20X,'**** SUPPLY CONDITIONS ****'//
     $        10X,'SUPPLY FROM:',T53,A12,/
     $        10X,'DESIGN FLOW',T50,'= ',F10.1,1X,A3,/
     $        10X,'SUPPLY PRESSURE HEAD',T50,'= ',F10.1,' FT'/
     $        10X,'PIPE ELEVATION',T50,'= ',F10.1,' FT'/
     $        10X,'TOTAL HYDRAULIC GRADE',T50,'= 'F10.1,' FT'//
     $        20X,'**** DISCHARGE CONDITIONS ****'//
     $        10X,'DISCHARGE TO:',T53,A20,/
     $        10X,'REQUIRED DISCHG. PRESSURE HEAD',T50,'= ',F10.1,' FT'/
     $        10X,'PIPE ELEVATION',T50,'= ',F10.1,' FT'/
     $        10X,'TOTAL HYDRAULIC GRADE',T50,'= ',F10.1,' FT')
      IF(IPR.GT.0)READ(*,141)IANS
141   FORMAT(I1)
C
C
C
C**** CONVERT INPUT FLOW RATE UNITS TO CUBIC FEET PER SECOND  ****
C**** FOR VELOCITY AND HEADLOSS CALCULATIONS                  ****
C
C
      IF (NUNIT .EQ. 2) THEN
C                                             *** CONVERT FROM GPM ***
          Q = Q/448.8
C
      ELSEIF (NUNIT .EQ. 3) THEN
C                                             *** CONVERT FROM MGD ***
      Q = Q * 1.547
      ELSEIF (NUNIT .EQ. 4) THEN
C                                             *** CONVERT FROM CMS ***
      Q = Q * 35.32
C
      ENDIF
C
C
C****  BEGIN DIAMETER ARRAY SEARCH LOOP TO FIND WATER VELOCITY ****
C****  LESS THAT 5 FT/SEC.  START WITH SMALLEST DIA. IN ARRAY. ****
C****  USE COUNT INDEX, ND, TO INDEX DIA. ARRAY OUTSIDE SEARCH ****
C****  LOOP.
```

```
C
C
      DO 200 N = 1,IDMAX
      D = DIA(N)/12.
          V = Q/ ((PI/4.) * D**2)
          ND = N
          IF ( V .LE. 5.)  GO TO 240
  200 CONTINUE
C
C
      IF(IPR.GT.0)WRITE (*,205) PTYPE, DIA(IDMAX), V, VUNIT
  205 FORMAT (1H1//5X,'***** PLEASE NOTE:'//
     $       2X,'EXCESSIVE WATER VELOCITY WAS ENCOUNTERED.'//
     $       2X,'THE LARGEST AVAILABLE INSIDE DIAMETER FOR ',A23,//
     $       2X,'IS',F7.0,' INCHES,'//
     $       2X,'THE DESIGN FLOW VELOCITY IN THIS PIPELINE IS ',F7.1,
     $       1X,A3,//
     $       2X,'EITHER PARALLEL PIPES OR A NON-STANDARD',
     $       ' PIPE SIZE MAY BE REQUIRED,'//
     $       2X,'TO CONVEY THE SPECIFIED DESIGN FLOW.')
C
      IERR=IERR+4
CJFB  GO TO 900
C
C
C****  BEGIN EVALUATION OF HEADLOSSES IN THE PIPELINE.  AT THIS
C****  POINT A DIAMETER HAS BEEN SELECTED WITH A WATER VELOCITY
C****  LESS OR EQUAL TO 5 FT/SEC.
C
C
C****  START LOOP TO CALC. HEADLOSS FOR EACH C-FACTOR (20 VALUES MAX.)
C****  DEFAULT C-FACTORS: NEW PIPE, C=130, OLD PIPE, C=100.
C
C
  240  DO 250, I = 1,ICMAX
C
C      *** BEGIN EACH PROBLEM ITERATION WITH SMALL .EQ. .FALSE.
C      *** SMALL WILL BECOME .TRUE. ONLY WHEN HEADLOSSES RESULT IN A
C      *** PIPE DISCHARGE PRESSURE BELOW SPECIFIED AMOUNT, AND A LARGER
C      *** PIPE DIAMETER IS REQUIRED.  THE LOGICAL VARIABLE SMALL
C      *** PREVENTS OSCILLATION BETWEEN THE ROUTINES WHICH SELECT
C      *** LARGER OR SMALLER PIPE I.D., AS REQUIRED TO SATISFY THE
C      *** SPECIFIED DISCHARGE CONDITIONS.
C
C
        SMALL = .FALSE.
  260      HL(I) = ANINT ((4.73 * XL/ D**4.87) * ( Q/C(I) )**1.852)
        ERH = AHL - HL(I)
C
C
C     ****  TEST ADEQUACY OF DISCHARGE HEAD.
C
C
       IF ( ERH .GE. 0.  .AND.  ERH .LT. 35.) THEN
```

```
C
C              *** STORE OUTPUT VALUES
C
             RH(I) = ANINT ( ERH + TG2)
           HLPT(I) = HL(I)/XLENT
           DIAM(I) = DIA(ND)
            VEL(I) = V
C
C
C
      ELSEIF ( ERH .GE. 35. .AND. ND .GT. 1 .AND. .NOT. SMALL) THEN
C
C                     *** THIS TEST WILL ROUTE THE PROG. BACK THROUGH
C                     *** THE DIA. ARRAY TO FIND A SMALLER PIPE.  THE
C                     *** OBJECTIVE IS TO REDUCE RESULTANT DISCHARGE
C                     *** HEAD AND SATISFY THE ACCEPTABLE DISCHG. COND.
C                     *** WHILE MAINT. VEL. BELOW MAX. ALLOW., 6 FPS.
C
C                     *** LOGICAL, SMALL, IS FALSE UNTIL ERH BECOMES
C                     *** NEG. SEE DEFINITION, ABOVE.
C
            N = ND - 1
            D = DIA(N)/12.
            V = Q/ ((PI/4.) * D**2)
            IF (V .LE. 6.) THEN
             ND = N
             GO TO 260
            ELSE
C
C                         *** COULD NOT FIND DIA. TO SATISFY
C                         *** THE ACCEPTABLE DISCHG., AND KEEP VEL.
C                         *** BELOW 6 FPS. PRINT RESULTS FOR V< 6 FPS.
C
C                 *** STORE OUTPUT VALUES
C
             IFLAG(I) = 1
             D = DIA(ND)/12.
             VEL(I) = Q/ ((PI/4.) * D**2)
             RH(I) = ANINT (ERH + TG2)
             HLPT(I) = HL(I)/XLENT
             DIAM(I) = DIA(ND)
C
C
C
             GO TO 250
            ENDIF
C
C
      ELSEIF ( ERH .GE. 35. .AND. ND .GT. 1 .AND. SMALL) THEN
C
C                         ***  THIS TEST HANDLES THE CASE WHERE THE
C                         ***  DISCHARGE HEAD OSCILLATES BETWEEN ERH
C                         ***  VALUES ABOVE AND BELOW THE ACCEPTABLE
C                         ***  DISCHARGE HEAD,( 0 .LE. ERH .LT. 35.).
```

jfb204f.for (continued)

```
C                         ***  DIA(ND) RESULTS IN DISCHARGE HEAD
C                         ***  GREATER THAN 35 FT OVER REQUIRED DISCHG.
C                         ***  PRESSURE, BUT THE PREVIOUS
C                         ***  DIAMETER, DIA(ND-1) RESULTS IN DISCHARGE
C                         ***  HEAD BELOW THE SPECIFIED AMOUNT.  STORE
C                         ***  THE RESULTS FOR DIA(ND), TRANSFERRED
C                         ***  FROM THE BLOCK WHERE SMALL = TRUE. SEE
C                         ***  BELOW.
C
            IFLAG(I) = 2
            RH(I) = ANINT ( ERH + TG2)
            HLPT(I) = HL(I)/XLENT
            DIAM(I) = DIA(ND)
             VEL(I) = V
C
C
            GO TO 250
C
      ELSEIF ( ERH .GE. 35. .AND. ND .EQ. 1) THEN
C
C                          *** THIS TEST HANDLES THE CASE WHERE
C                          *** THE SMALLEST AVAIL. DIAMETER RESULTS
C                          *** IN DISCHARGE HEAD GREATER THAN 35 FT
C                          *** OVER THE SPECIFIED AMOUNT.
C
C
            IFLAG(I) = 3
            RH(I) = ANINT ( ERH + TG2)
            HLPT(I) = HL(I)/XLENT
            DIAM(I) = DIA(ND)
           VEL(I) = V
C
C
C
            GO TO 250
C
      ELSE
C
C                          *** THE RESULTANT DISCHARGE HEAD IS LESS
C                          *** THAN THE AMOUNT SPECIFIED.  AT THIS
C                          *** POINT, THE PROG. IS ROUTED FORWARD
C                          *** THROUGH THE DIAMETER ARRAY TO FIND A
C                          *** LARGER PIPE SIZE. THE OBJECTIVE IS TO
C                          *** FIND THE SMALLEST PIPE SIZE THAT
C                          *** SATISFIES THE SPECIFIED DISCHARGE HEAD
C                          *** CONDITION.
C
            SMALL = .TRUE.
              N = ND + 1
            IF (N .LE. IDMAX) THEN
             D = DIA(N)/12.
             V = Q/ ((PI/4.) * D**2)
            ND = N
            GO TO 260
            ELSE
```

jfb204f.for (continued)

```
C
C                      *** THE LARGEST DIAMETER WAS CHECKED AND THE
C                      *** DISCHARGE HEAD IS STILL BELOW THE
C                      *** SPECIFIED AMOUNT.  EITHER THE SUPPLY HEAD
C                      *** IS TOO LOW (TG1 < PMAX), OR THE PIPELINE
C                      *** REQUIRES BOOSTER STATIONS ALONG THE ROUTE.
C
C
C
C                     **** STORE OUTPUT VALUES
C
                 IFLAG(I) = 4
                 RH(I) = ANINT ( ERH + TG2)
                 HLPT(I) = HL(I)/XLENT
                  VEL(I) = Q/ ((PI/4.) * D**2)
                 DIAM(I) = DIA(ND)
C
C
              ENDIF
C
        ENDIF
C
  250 CONTINUE
C
C
C      ****  IF(IPR.GT.0)WRITE TO OUTPUT
C
           IF(IPR.GT.0)WRITE (*,350) ICMAX
C
           DO 355 I = 1, ICMAX
C
C            *** IF FLOW RATE UNITS ARE CUBIC METERS PER SEC, CONVERT
C            *** VELOCITY FROM FT PER SEC TO METERS PER SEC FOR OUTPUT
C
           IF (NUNIT .EQ. 4) THEN
             VEL(I) = VEL(I) * 0.3048
           ENDIF
C
           IF(IPR.GT.0)WRITE (*,360) C(I), DIAM(I), RH(I), TG2, VEL(I),
     $           VUNIT, HL(I), AHL, HLPT(I)
C
C            *** PRINT WARNING MESSAGES, IF ANY
C
           IF (IFLAG(I) .EQ. 1) THEN
              IF(IPR.GT.0)WRITE (*,364)
           ELSEIF (IFLAG(I) .EQ. 2) THEN
              IF(IPR.GT.0)WRITE (*,366)
           ELSEIF (IFLAG(I) .EQ. 3) THEN
              IF(IPR.GT.0)WRITE (*,368)
           ELSEIF (IFLAG(I) .EQ. 4) THEN
              IF(IPR.GT.0)WRITE (*,370) C(I)
           ENDIF
C
      IF(IPR.GT.0)READ(*,141)IANS
```

jfb204f.for (continued)

```
355   CONTINUE
C
C
       DO 400 I = 1,ICMAX
          IF (NUNIT .EQ. 4) THEN
            VEL(I) = VEL(I) * 0.3048
          ENDIF
C
             IF (IFLAG(I) .EQ. 1) THEN
             ELSEIF (IFLAG(I) .EQ. 2) THEN
             ELSEIF (IFLAG(I) .EQ. 3) THEN
             ELSEIF (IFLAG(I) .EQ. 4) THEN
             ENDIF
  400    CONTINUE
C
C
C
C     **** OUTPUT FORMATS
C
C
  350       FORMAT (1H1//15X,'PROGRAM PIPE-D RESULTS FOR',I2,
     $             ' PIPE ROUGHNESS CONDITIONS')
C
  360       FORMAT (///10X,'*** RESULTS BASED ON C-FACTOR, C = '
     $         ,F5.1,///
     $         10X,'MINIMUM INSIDE DIAMETER ',T50,'= ',F7.1,' INCHES'//
     $         10X,'CALCULATED DISCHARGE PRESSURE HEAD',T50,'= ',F7.1,
     $         ' FT'//
     $         10X,'REQUIRED DISCHG. PRESSURE HEAD',T50,'= ',F7.1,
     $         ' FT'//
     $         10X,'WATER VELOCITY AT DESIGN FLOW ',T50,'= ',F7.1,
     $         1X,A3,//
     $         10X,'TOTAL HEADLOSS IN PIPE SEGMENT ',T50,'= ',F7.1,
     $         ' FT'//
     $         10X,'ALLOWABLE HEADLOSS IN PIPE SEGMENT',T50,'= ',F7.1,
     $         ' FT'//
     $         10X,'HEADLOSS PER 1,000 LF ',T50,'= ',F7.1,' FT'/)
C
C              **** IFLAG = 1
C
  364          FORMAT (//10X,'*******  PLEASE NOTE:'//
     $                10X,'DISCHARGE HEAD IS GREATER THAN REQ''D.'
     $                ' BUT SMALLER PIPE '/10X,'CAUSES WATER VELOCITY'
     $                ' GREATER THAN 6 FPS.')
C
C              **** IFLAG = 2
C
  366          FORMAT (//10X,'*******  PLEASE NOTE:'//
     $                10X,'DISCHARGE HEAD IS GREATER THAN REQ''D.'
     $                ' BUT SMALLER PIPE '/10X,'CAUSES DISCHARGE HEAD'
     $                ' BELOW THE SPECIFIED AMOUNT')
C
C              **** IFLAG = 3
```

jfb204f.for (continued)

```
C
  368          FORMAT (//10X,'*******  PLEASE NOTE:'//
     $              10X,'THE SMALLEST AVAILABLE DIAMETER WAS CHECKED'
     $              ' AND THE DISCHARGE'/
     $              10X,'HEAD IS STILL GREATER THAN THE AMOUNT'
     $              ' SPECIFIED.  A SMALLER PIPE'/
     $              10X,'SIZE MAY SATISFY THE SPECIFIED DISCHARGE'
     $              ' CONDITIONS.  CHECK OTHER'/
     $              10X,'PIPE MATERIALS FOR SMALLER AVAILABLE PIPE'
     $              ' SIZES.  CHECK DESIGN FLOW.')
C
C               **** IFLAG = 4
C
  370           FORMAT (//10X,'****** WARNING ******'//
     $              10X,'THE LARGEST AVAIL. INSIDE PIPE DIAMETER'
     $              ' WAS CHECKED,'//
     $              10X,'AND IT COULD NOT SATISFY THE DISCHARGE REQUIR'
     $              'EMENT FOR A PIPE'//
     $              10X,'WITH A C-FACTOR = ',F5.1)
C
C
C
      DIN=DIAM(1)
      HLTPS=HL(1)
900   CONTINUE
      RETURN
      END
      SUBROUTINE GETDAT(IYR,IMO,IDAY)
      CALL IDATE(I,J,K)
      IDAY=I
      IMO=J
      IYR=K
      RETURN
      END
      CALL ITIME(I,J,K)
      IHR=I
      IMIN=J
      ISEC=K
      I100TH=0
      RETURN
      END
```

parc.aml

```
/*  parc.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/**********************************************************************
/*
/* USAGE: This aml creates the PARC   coverage and loads it with
/*        data from the ASCII files  PARC.D00 and PARC.D01
/*
/*
/*
/**********************************************************************
/*
/*  INVOKED BY: [ARC] &run parc.aml
/*  RELATED COVERAGES:   parc
/*  RELATED FILES:   parc.d00   parc.d01
/*
/**********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> PARC   Coverage creation in progress'
&type '<> Processing'
&MESSAGES &OFF &ALL
&SEVERITY &WARNING &IGNORE
&SEVERITY &ERROR &IGNORE
KILL PARC ALL
TABLES
KILL PARC.AAT
KILL PARC.DAT
Q STOP
GENERATE PARC
INPUT PARC.D00
LINES
QUIT
BUILD PARC   LINE
TABLES
DEFINE PARC.DAT
PARC-ID
10
10
I
PSLON
10
10
N
4
PSLAT
10
10
N
4
```

parc.aml (continued)

```
PDLON
10
10
N
4
PDLAT
10
10
N
4
PSELV
10
10
N
1
PDELV
10
10
N
1
PSTRNS
10
10
I
PDTRNS
10
10
I
PSIDN
10
10
I
PDIDN
10
10
I
PSCNTY
10
10
I
PDCNTY
10
10
I
PSCAP
10
10
I
PDCAP
10
10
I
```

parc.aml (continued)

```
PSCIP
10
10
I
PDCIP
10
10
I
PLENGTH
10
10
N
1
PLOWB
10
10
I
PLOWBO
10
10
I
PUPPB
10
10
I
PUPPBO
10
10
I
PCOST
10
10
I
PCOSTO
10
10
I
PFLOWP
10
10
I
PFLOW
10
10
I
PFEAS
10
10
I
PARC_ID
10
10
C
~
```

parc.aml (continued)

```
ADD FROM PARC.D01
Q STOP
JOINITEM PARC.AAT PARC.DAT PARC.AAT PARC-ID PARC-ID LINEAR
&type '<> Complete.     Created coverage:    PARC'
&type ' '
QUIT
&return
```

export.aml

```
/*  export.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/***********************************************************************
/*
/* USAGE: This aml exports the PARC   coverage to an ASCII file
/*        named  arcparc.pdb.
/*
/*
/*
/***********************************************************************
/*
/*  INVOKED BY: [ARC] &run export.aml
/*  RELATED COVERAGES:  parc
/*  RELATED FILES:    arcparc.pdb
/*
/***********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> PARC   Coverage export in progress'
&type '<> Processing'
&MESSAGES &OFF &ALL
&SEVERITY &WARNING &IGNORE
&SEVERITY &ERROR &IGNORE
&SYS rm nexparc.pdb
&SYS rm arcparc.pdb
&SYS rm jen1.pdb
TABLES
SELECT PARC.AAT
CALC PFEAS = 0
CALC PFLOW = 0
CALC PLOWB = PLOWBO
CALC PUPPB = PUPPBO
CALC PCOST = PCOSTO
&r unload2nxp arcparc.pdb PARC-ID PSLON PSLAT PDLON PDLAT PSELV PDELV
PSTRNS ~
PDTRNS PSIDN PDIDN PSCNTY PDCNTY PSCAP PDCAP PSCIP PDCIP PLENGTH PLOWB
PUPPB PCOST PFLOWP PFLOW PFEAS PARC_ID
Q STOP
&type '<> Complete      Created file: arcparc.pdb'
&type ' '
QUIT
```

unload2nxp.aml

```
/*
/*   unload2nxp.aml
/*   V 1.0
/*   Bastille Day 1994
/*   Tom Evans
/*   Center for Research in Water Resources, The University of Texas at Austin
/*
/*****************************************************************************
/*
/*   unload2nxp -- Creates an nxpdb file containing the items listed in the
/*   command line.
/*
/*   USAGE:  unload2nxp <nxpdb_file> {item1 item2 ...}
/*
/*   The selected items are written to a fixed-item-width ASCII text file by
/*   the UNLOAD command in TABLES.  That text file is re-written into nxpdb
/*   format by the system program expnxp.
/*
/*   Only text is transfered between ARC/INFO and NEXPERT.  It is the
/*   responsibility of the user to see to it that the proper matching of
/*   data types occurs.
/*
/*   NOTE:  THE TABLES UNLOAD COMMAND WRITES DATES OUT IN THE ONE OF THE
/*   FOLLOWING FORMATS:
/*     MM/DD/YYYY   (OUTPUT WIDTH = 10)
/*     MM/DD/YY     (OUTPUT WIDTH = 8)
/*     MM/DD        (OUTPUT WIDTH = 5)
/*   THE FORMAT FOR DATE PROPERTIES IN NEXPERT TO BE READ FROM AN NXPDB
/*   DATABASE CREATED BY THIS COMMAND SHOULD BE SET TO ACCOMODATE THE
/*   FORMAT THAT THE DATA COMES IN.  FOR THE FIRST OF THE FORMATS LISTED
/*   ABOVE THE FOLLOWING DATE FORMAT WORKS:
/*
/*     yyyymmdd;m/d/yyyy;m/ d/yyyy
/*
/*   THE SECOND INPUT FORMAT IS REQUIRED TO ACCOMODATE THE INTERNAL BLANK THAT
/*   THE UNLOAD TEXT FORMAT PRODUCES.  WITH THIS FORMAT, DATES CAN BE READ AS
/*   (FOR EXAMPLE) 5/ 7/1993 OR 11/30/1993, AND NEXPERT WILL DISPLAY THEM AS
/*   19930507 AND 19931130.  (See Nexpert documentation on formats in
/*   general and date formats in particular.)
/*
/*****************************************************************************
/*
/*   CALLED BY: user
/*
/*   SYSTEM CALLS:  expnxp
/*
/*   AML CALLS: none
/*
/*   RELATED FILES:
/*     zzdata.tmp -- text file containing INFO data, created by TABLES unload
/*      command.
/*     zzformat.tmp -- text file containing INFO formats (item widths), created
/*       by TABLES unload command.
/*
/*****************************************************************************
/*
```

unload2nxp.aml (continued)

```
/*  VARIABLES
/*
/* LOCAL
/*
/*  delstat(integer):  Status variable set by file delete.
/*
/*  itemlist (string):  A string containing the names of the items to
/*    be written to the nxpdb file.
/*
/*  msg (string):  Message to user describing error conditions.
/*
/*  namelist (string):  User's list of items to transfer. (argument)
/*
/*  nxpfile (string):  The name of the nxpdb file to be created by this
/*    program.  (argument)
/*
/*  nextitem (string):  An item read from namelist.
/*
/*  j1, j2, j3, sl (integer):  String index counters.  Necessary to outsmart
/*    idiocy of &do &list behavior with string variable.
/*
/* GLOBAL
/*    no global variables.
/*
/*****************************************************************************
/*
/*  The program actually starts here.
/*

/*  Read the nxpdb file name and the list of item names from the command line.
&args nxpfile namelist:REST
/*  Make sure the program was launched from TABLES with a selected table.
&if [SHOW PROGRAM] ne TABLES &then ~
  &return &error infonxp must run under TABLES.
&if [NULL %nxpfile%] &then ~
  &return &error Usage:  unload2nxp <nxpdb_file> {item...item}\
&if [INDEX [QUOTE [SHOW SELECT]] 'No file selected.'] = 1 &then ~
  &return &error No selected data file.\

/*  If the nxpdb file (or another file with the same name) already exists,
/*  give the user a chance to delete it or quit.
&if [EXISTS %nxpfile% -FILE] &then &do /* nxpdb file conflict block
  &sv delstat := [DELETE %nxpfile% -FILE]
  &if %delstat% ne 0 &then &do  /* delete error block
    &ty \Error deleting %nxpfile%
    &return &error infonxp aborted.
  &end  /* end delete error block
&end /* end nxpdb file conflict block

/*  If no items appear on command line, transfer all items in table.
&if [NULL %namelist%] &then ~
  &sv itemlist := [LISTITEM [SHOW SELECT] -INFO]

/*  test item names on command line and construct item list.
&else &do /* null namelist block
  &sv itemlist := ''
  &sv sl := [LENGTH %namelist%]
  &do &while %sl% > 0 /* loop for item names

    /* extract an item name from namelist (the hard way)
```

unload2nxp.aml (continued)

```
    &sv j1 := [INDEX %namelist% ' ']
    &if %j1% = 0 &then &do
       &sv nextitem := %namelist%
       &sv namelist := ''
    &end
    &else &do
      &sv j2 := %j1% - 1
      &sv j3 := %j1% + 1
      &sv nextitem := [SUBSTR %namelist% 1 %j2%]
      &sv namelist := [SUBSTR %namelist% %j3% %sl%]
    &end

    /* check to see if the item is defined in the INFO file
    &if [ITEMINFO [SHOW SELECT] -INFO %nextitem% -EXISTS] ne .TRUE. &then ~
      &do /* no item block
      &ty \Item %nextitem% not present in selected INFO file.
      &ty  %nextitem% deleted from item list.\
      &sv nextitem :=  /* set null
    &end  /* end no item block
    &sv itemlist := [QUOTE [UNQUOTE %itemlist%] %nextitem%]
    &sv sl := [LENGTH %namelist%]
  &end /* end item name loop
&end /* end null namelist block

&sv itemlist := [UNQUOTE %itemlist%]

/*  if the list of items has no members, give up.
&if [NULL %itemlist%] &then ~
  &return &error No valid item names listed.

/*  silently create the data and format text files
&messages &off
unload zzdata.tmp %itemlist% columnar zzformat.tmp INIT
&messages &on

/*  make zzdata.tmp and zzformat.tmp into an nxpdb file.
&sys expnxp %nxpfile% %itemlist%

/*  remove the data files
&sv delstat := [DELETE zzdata.tmp -FILE]
&if %delstat% ne 0 &then ~
  &ty \Error deleting data file
&sv delstat := [DELETE zzformat.tmp -FILE]
&if %delstat% ne 0 &then ~
  &ty \Error deleting format file

/*  upon completion, write a message to the user
&if [EXISTS %nxpfile% -FILE] &then &do
  &ty wrote items:  %itemlist%
  &ty to nxpdb file:  %nxpfile%.\
&end

&else &ty \\*** PROCEDURE FAILED: nxpdb file not written. ***

/*
/*  done
/*

&return
```

r10p0.tkb

```
(@VERSION=  020)
(@PROPERTY= NPARC_ID     @TYPE=String;)
(@PROPERTY= NPCOST       @TYPE=Integer;)
(@PROPERTY= NPDCAP       @TYPE=Integer;)
(@PROPERTY= NPDCIP       @TYPE=Integer;)
(@PROPERTY= NPDCNTY      @TYPE=Integer;)
(@PROPERTY= NPDELV       @TYPE=Float;)
(@PROPERTY= NPDIDN       @TYPE=Integer;)
(@PROPERTY= NPDLAT       @TYPE=Float;)
(@PROPERTY= NPDLON       @TYPE=Float;)
(@PROPERTY= NPDTRNS      @TYPE=Integer;)
(@PROPERTY= NPFEAS       @TYPE=Integer;)
(@PROPERTY= NPFLOW       @TYPE=Integer;)
(@PROPERTY= NPFLOWP      @TYPE=Integer;)
(@PROPERTY= NPLENGTH     @TYPE=Float;)
(@PROPERTY= NPLOWB       @TYPE=Integer;)
(@PROPERTY= NPSCAP       @TYPE=Integer;)
(@PROPERTY= NPSCIP       @TYPE=Integer;)
(@PROPERTY= NPSCNTY      @TYPE=Integer;)
(@PROPERTY= NPSELV       @TYPE=Float;)
(@PROPERTY= NPSIDN       @TYPE=Integer;)
(@PROPERTY= NPSLAT       @TYPE=Float;)
(@PROPERTY= NPSLON       @TYPE=Float;)
(@PROPERTY= NPSTRNS      @TYPE=Integer;)
(@PROPERTY= NPUPPB       @TYPE=Integer;)


(@CLASS=    NPARC
      (@PROPERTIES=
            NPARC_ID
            NPCOST
            NPDCAP
            NPDCIP
            NPDCNTY
            NPDELV
            NPDIDN
            NPDLAT
            NPDLON
            NPDTRNS
            NPFEAS
            NPFLOW
            NPFLOWP
            NPLENGTH
            NPLOWB
            NPSCAP
            NPSCIP
            NPSCNTY
            NPSELV
            NPSIDN
            NPSLAT
            NPSLON
            NPSTRNS
            NPUPPB
      )
)
```

r10p0.tkb (continued)

```
(@OBJECT=   AIDATA_ACQUIRED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   AIDATA_EXPORTED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   LPSOLVED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   NETGEND
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@RULE=     ARCINFO_DATA_RETRIEVAL
      (@LHS=
            (Retrieve   ("arcparc.pdb")
      (@TYPE=NXPDB;@FILL=ADD;@NAME="'P'!PARC_ID!";\
@CREATE=|NPARC|;@PROPS=NPARC_ID,NPSLON,NPSLAT,\
NPDLON,NPDLAT,NPSELV,NPDELV,NPSTRNS,NPDTRNS,\
NPSIDN,NPDIDN,NPSCNTY,NPDCNTY,NPSCAP,NPDCAP,\
NPSCIP,NPDCIP,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOWP,NPFLOW,NPFEAS;@FIELDS="PARC_ID","PSLON",\
"PSLAT","PDLON","PDLAT","PSELV","PDELV","PSTRNS",\
"PDTRNS","PSIDN","PDIDN","PSCNTY","PDCNTY",\
"PSCAP","PDCAP","PSCIP","PDCIP","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOWP","PFLOW",\
"PFEAS";))
      )
      (@HYPO=     AIDATA_ACQUIRED)
      (@RHS=
            (Execute    ("Message") (@STRING="@TEXT=ARCINFO  COVERAGES
RETRIEVED,\
@OK";))
      )
)

(@RULE=     EXPORT
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
      )
      (@HYPO=     AIDATA_EXPORTED)
      (@RHS=
            (Write      ("nexparc.pdb")
```

jen1.for (continued)

```
      GO TO 280
232   CONTINUE
      IF(D.GT.DEL)GO TO 234
      GO TO 236
234   CONTINUE
      GO TO 280
236   CONTINUE
      DEL=D
      KE=K
      GO TO 280
240   CONTINUE
      IF(F(K).EQ.0.)GO TO 241
      GO TO 250
241   CONTINUE
      GO TO 280
250   CONTINUE
      IF(-D.GT.DEL)GO TO 251
      GO TO 260
251   CONTINUE
      GO TO 280
260   CONTINUE
      DEL=-D
      KE=-K
C
280   CONTINUE
C
C
      IF(KE.EQ.0)GO TO 281
      GO TO 285
281   CONTINUE
      GO TO 290
285   CONTINUE
      IFIN=0
      IST=0
      SN=I+1
      FN=N
      IF(SN.GT.N)GO TO 286
      GO TO 287
286   CONTINUE
      SN=1
      GO TO 289
287   CONTINUE
C
289   CONTINUE
      GO TO 900
C
C
290   CONTINUE
C
C     COMPLETE
C
300   CONTINUE
      IF(SN.EQ.1)GO TO 310
      GO TO 320
310   CONTINUE
```

r10p4.tkb

```
(@VERSION=   020)
(@PROPERTY= NPARC_ID     @TYPE=String;)
(@PROPERTY= NPCOST       @TYPE=Integer;)
(@PROPERTY= NPDCAP       @TYPE=Integer;)
(@PROPERTY= NPDCIP       @TYPE=Integer;)
(@PROPERTY= NPDCNTY      @TYPE=Integer;)
(@PROPERTY= NPDELV       @TYPE=Float;)
(@PROPERTY= NPDIDN       @TYPE=Integer;)
(@PROPERTY= NPDLAT       @TYPE=Float;)
(@PROPERTY= NPDLON       @TYPE=Float;)
(@PROPERTY= NPDTRNS      @TYPE=Integer;)
(@PROPERTY= NPFEAS       @TYPE=Integer;)
(@PROPERTY= NPFLOW       @TYPE=Integer;)
(@PROPERTY= NPFLOWP      @TYPE=Integer;)
(@PROPERTY= NPLENGTH     @TYPE=Float;)
(@PROPERTY= NPLOWB       @TYPE=Integer;)
(@PROPERTY= NPSCAP       @TYPE=Integer;)
(@PROPERTY= NPSCIP       @TYPE=Integer;)
(@PROPERTY= NPSCNTY      @TYPE=Integer;)
(@PROPERTY= NPSELV       @TYPE=Float;)
(@PROPERTY= NPSIDN       @TYPE=Integer;)
(@PROPERTY= NPSLAT       @TYPE=Float;)
(@PROPERTY= NPSLON       @TYPE=Float;)
(@PROPERTY= NPSTRNS      @TYPE=Integer;)
(@PROPERTY= NPUPPB       @TYPE=Integer;)


(@CLASS=     NPARC
      (@PROPERTIES=
             NPARC_ID
             NPCOST
             NPDCAP
             NPDCIP
             NPDCNTY
             NPDELV
             NPDIDN
             NPDLAT
             NPDLON
             NPDTRNS
             NPFEAS
             NPFLOW
             NPFLOWP
             NPLENGTH
             NPLOWB
             NPSCAP
             NPSCIP
             NPSCNTY
             NPSELV
             NPSIDN
             NPSLAT
             NPSLON
             NPSTRNS
             NPUPPB
      )
)
```

```
(@OBJECT=   AIDATA_ACQUIRED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   AIDATA_EXPORTED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   CNTY_GW_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   CNTY_LS_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   CNTY_LSU_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   LPSOLVED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   NETGEND
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   TEXAS_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   XCNTY_LS_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@RULE=     ARCINFO_DATA_RETRIEVAL
```

r10p4.tkb (continued)

```
	(@LHS=
		(Retrieve	("arcparc.pdb")
	(@TYPE=NXPDB;@FILL=ADD;@NAME="'P'!PARC_ID!";\
@CREATE=|NPARC|;@PROPS=NPARC_ID,NPSLON,NPSLAT,\
NPDLON,NPDLAT,NPSELV,NPDELV,NPSTRNS,NPDTRNS,\
NPSIDN,NPDIDN,NPSCNTY,NPDCNTY,NPSCAP,NPDCAP,\
NPSCIP,NPDCIP,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOWP,NPFLOW,NPFEAS;@FIELDS="PARC_ID","PSLON",\
"PSLAT","PDLON","PDLAT","PSELV","PDELV","PSTRNS",\
"PDTRNS","PSIDN","PDIDN","PSCNTY","PDCNTY",\
"PSCAP","PDCAP","PSCIP","PDCIP","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOWP","PFLOW",\
"PFEAS";))
	)
	(@HYPO=		AIDATA_ACQUIRED)
	(@RHS=
		(Execute	("Message")	(@STRING="@TEXT=ARCINFO	COVERAGES
RETRIEVED,\
@OK";))
	)
)

(@RULE=		EXPORT
	(@LHS=
		(Yes	(TEXAS_RULES_APPLIED))
	)
	(@HYPO=		AIDATA_EXPORTED)
	(@RHS=
		(Write		("nexparc.pdb")
	(@TYPE=NXPDB;@FILL=NEW;@NAME="'P'!PARC_ID!";\
@PROPS=NPARC_ID,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOW,NPFEAS;@FIELDS="PARC_ID","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOW","PFEAS";@ATOMS=<<|NPARC|>>;\
))
		(Execute	("Message")	(@STRING="@TEXT=PARED NETWORK
EXPORTED,\
@OK";))
	)
)

(@RULE=		TEXAS_RULE001
	(@LHS=
		(Yes	(AIDATA_ACQUIRED))
		(>	(<|NPARC|>.NPSIDN)		(0))
		(<=	(<|NPARC|>.NPSIDN)		(99))
		(=	(<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)		(0))
	)
	(@HYPO=		CNTY_GW_RULE_APPLIED)
	(@RHS=
		(Execute	("SetValue")
	(@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
		(Execute	("Message")	(@STRING="@TEXT=IN-COUNTY
GROUNDWATER TRANSPORT COST RESET TO ZERO	(TEXAS_RULE\
001),@OK";))
```

r10p4.tkb (continued)

```
        )
)

(@RULE=     TEXAS_RULE002
        (@LHS=
                (Yes    (AIDATA_ACQUIRED))
                (=      (<|NPARC|>.NPSIDN)          (99999))
                (=      (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
        )
        (@HYPO=     CNTY_LS_RULE_APPLIED)
        (@RHS=
                (Execute    ("SetValue")
        (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
                (Execute    ("Message") (@STRING="@TEXT=IN-COUNTY LS
TRANSPORT COST RESET TO ZERO  (TEXAS_RULE002),\
@OK";))
        )
)

(@RULE=     TEXAS_RULE003
        (@LHS=
                (Yes    (AIDATA_ACQUIRED))
                (=      (<|NPARC|>.NPSIDN)          (99999))
                (=      (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
                (<>     (<|NPARC|>.NPDIDN)          (993))
        )
        (@HYPO=     CNTY_LSU_RULE_APPLIED)
        (@RHS=
                (Execute    ("SetValue")
        (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=3,\
@STRAT=SETFWRD";))
                (Execute    ("Message") (@STRING="@TEXT=IN-COUNTY LS MUN,
CLASS O,\
 AND CLASS A USERS  NOT FEASIBLE  (TEXAS_RULE003),\
@OK";))
        )
)

(@RULE=     LPSOLV
        (@LHS=
                (Yes    (NETGEND))
        )
        (@HYPO=     LPSOLVED)
        (@RHS=
                (Execute    ("Message") (@STRING="@TEXT=EXECUTE LP
SOLVER,@OK";))
                (Execute    ("jen1")    (@TYPE=EXE;@WAIT=TRUE;))
                (Execute    ("jfb237")  (@TYPE=EXE;@WAIT=TRUE;))
        )
)

(@RULE=     NETGEN
        (@LHS=
                (Yes    (AIDATA_EXPORTED))
```

r10p4.tkb (continued)

```
        )
        (@HYPO=       NETGEND)
        (@RHS=
              (Execute      ("jfb205b") (@TYPE=EXE;@WAIT=TRUE;))
        )
)

(@RULE=      TEX
        (@LHS=
              (Yes    (AIDATA_ACQUIRED))
              (<>     (COMPARE(COMPARE(CNTY_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTY_GW_RULE_APPLIED,FALSE)))          (0))
              (<>     (COMPARE(COMPARE(XCNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(XCNTY_LS_RULE_APPLIED,FALSE)))          (0))
              (<>     (COMPARE(COMPARE(CNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LS_RULE_APPLIED,FALSE)))          (0))
              (<>     (COMPARE(COMPARE(CNTY_LSU_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LSU_RULE_APPLIED,FALSE)))          (0))
        )
        (@HYPO=       TEXAS_RULES_APPLIED)
)
(@RULE=      TEXAS_RULE004
        (@LHS=
              (Yes    (AIDATA_ACQUIRED))
              (>      (<|NPARC|>.NPSIDN)         (99900))
              (<>     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)        (0))
        )
        (@HYPO=       XCNTY_LS_RULE_APPLIED)
        (@RHS=
              (Execute      ("SetValue")
        (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=4,\
@STRAT=SETFWRD";))
              (Execute      ("Message") (@STRING="@TEXT=LOCAL SUPPLY AND
SHORTAGE EXPORT  NOT FEASIBLE  (TEXAS_RULE004)\
,@OK";))
        )
)
(@GLOBALS=
        @INHVALUP=FALSE;
        @INHVALDOWN=TRUE;
        @INHOBJUP=FALSE;
        @INHOBJDOWN=FALSE;
        @INHCLASSUP=FALSE;
        @INHCLASSDOWN=TRUE;
        @INHBREADTH=TRUE;
        @INHPARENT=FALSE;
        @PWTRUE=TRUE;
        @PWFALSE=TRUE;
        @PWNOTKNOWN=TRUE;
        @EXHBWRD=TRUE;
        @PTGATES=TRUE;
        @PFACTIONS=TRUE;
        @SOURCESON=TRUE;
        @CACTIONSON=TRUE;
        @SUGLIST=LPSOLVED;
)
```

r10p9.tkb

```
(@VERSION=  020)
(@PROPERTY= NPARC_ID     @TYPE=String;)
(@PROPERTY= NPCOST       @TYPE=Integer;)
(@PROPERTY= NPDCAP       @TYPE=Integer;)
(@PROPERTY= NPDCIP       @TYPE=Integer;)
(@PROPERTY= NPDCNTY      @TYPE=Integer;)
(@PROPERTY= NPDELV       @TYPE=Float;)
(@PROPERTY= NPDIDN       @TYPE=Integer;)
(@PROPERTY= NPDLAT       @TYPE=Float;)
(@PROPERTY= NPDLON       @TYPE=Float;)
(@PROPERTY= NPDTRNS      @TYPE=Integer;)
(@PROPERTY= NPFEAS       @TYPE=Integer;)
(@PROPERTY= NPFLOW       @TYPE=Integer;)
(@PROPERTY= NPFLOWP      @TYPE=Integer;)
(@PROPERTY= NPLENGTH     @TYPE=Float;)
(@PROPERTY= NPLOWB       @TYPE=Integer;)
(@PROPERTY= NPSCAP       @TYPE=Integer;)
(@PROPERTY= NPSCIP       @TYPE=Integer;)
(@PROPERTY= NPSCNTY      @TYPE=Integer;)
(@PROPERTY= NPSELV       @TYPE=Float;)
(@PROPERTY= NPSIDN       @TYPE=Integer;)
(@PROPERTY= NPSLAT       @TYPE=Float;)
(@PROPERTY= NPSLON       @TYPE=Float;)
(@PROPERTY= NPSTRNS      @TYPE=Integer;)
(@PROPERTY= NPUPPB       @TYPE=Integer;)


(@CLASS=     NPARC
      (@PROPERTIES=
            NPARC_ID
            NPCOST
            NPDCAP
            NPDCIP
            NPDCNTY
            NPDELV
            NPDIDN
            NPDLAT
            NPDLON
            NPDTRNS
            NPFEAS
            NPFLOW
            NPFLOWP
            NPLENGTH
            NPLOWB
            NPSCAP
            NPSCIP
            NPSCNTY
            NPSELV
            NPSIDN
            NPSLAT
            NPSLON
            NPSTRNS
            NPUPPB
      )
)
```

r10p9.tkb (continued)

```
(@OBJECT=   AIDATA_ACQUIRED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   AIDATA_EXPORTED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CB_JACKSON_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTY_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTY_LS_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTY_LSU_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTYA_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTYB_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   CNTYB_SW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)
```

r10p9.tkb (continued)

```
(@OBJECT=   CO20_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   COLETTO_CR_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   DEM10_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   GW40_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   GWEX_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   LPSOLVED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   NETGEND
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   REG10_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   SUP10_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)
```

r10p9.tkb (continued)

```
(@OBJECT=   TEXAS_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   TILDEN_GW_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   XCNTY_LS_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@RULE=     ARCINFO_DATA_RETRIEVAL
      (@LHS=
            (Retrieve   ("arcparc.pdb")
      (@TYPE=NXPDB;@FILL=ADD;@NAME="'P'!PARC_ID!";\
@CREATE=|NPARC|;@PROPS=NPARC_ID,NPSLON,NPSLAT,\
NPDLON,NPDLAT,NPSELV,NPDELV,NPSTRNS,NPDTRNS,\
NPSIDN,NPDIDN,NPSCNTY,NPDCNTY,NPSCAP,NPDCAP,\
NPSCIP,NPDCIP,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOWP,NPFLOW,NPFEAS;@FIELDS="PARC_ID","PSLON",\
"PSLAT","PDLON","PDLAT","PSELV","PDELV","PSTRNS",\
"PDTRNS","PSIDN","PDIDN","PSCNTY","PDCNTY",\
"PSCAP","PDCAP","PSCIP","PDCIP","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOWP","PFLOW",\
"PFEAS";))
      )
      (@HYPO=     AIDATA_ACQUIRED)
      (@RHS=
            (Execute    ("Message") (@STRING="@TEXT=ARCINFO  COVERAGES
RETRIEVED,\
@OK";))
      )
)

(@RULE=     EXPORT
      (@LHS=
            (Yes  (TEXAS_RULES_APPLIED))
            (Yes  (REG10_RULES_APPLIED))
            (Yes  (SUP10_RULES_APPLIED))
            (Yes  (DEM10_RULES_APPLIED))
      )
      (@HYPO=     AIDATA_EXPORTED)
      (@RHS=
            (Write      ("nexparc.pdb")
      (@TYPE=NXPDB;@FILL=NEW;@NAME="'P'!PARC_ID!";\
@PROPS=NPARC_ID,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOW,NPFEAS;@FIELDS="PARC_ID","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOW","PFEAS";@ATOMS=<<|NPARC|>>;\
))
```

r10p9.tkb (continued)

```
              (Execute    ("Message") (@STRING="@TEXT=PARED NETWORK
EXPORTED,\
@OK";))
       )
)

(@RULE=       DEM10_RULE301
       (@LHS=
              (Yes   (AIDATA_ACQUIRED))
              (=     (<|NPARC|>.NPDCNTY)      (120))
              (=     (<|NPARC|>.NPDIDN)       (993))
              (>     (<|NPARC|>.NPSIDN)       (0))
              (<=    (<|NPARC|>.NPSIDN)       (99))
       )
       (@HYPO=      CB_JACKSON_RULE_APPLIED)
       (@RHS=
              (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPUPPB;@STRING="@VALUE=999999,\
@STRAT=SETFWRD";))
              (Execute    ("Message") (@STRING="@TEXT=JACKSON COUNTY CLASS
B USERS  NOT RESTRICTED BY RULE 104  (DEM1\
0_RULE301),@OK";))
       )
)

(@RULE=       TEXAS_RULE001
       (@LHS=
              (Yes   (AIDATA_ACQUIRED))
              (>     (<|NPARC|>.NPSIDN)       (0))
              (<=    (<|NPARC|>.NPSIDN)       (99))
              (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
       )
       (@HYPO=      CNTY_GW_RULE_APPLIED)
       (@RHS=
              (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
              (Execute    ("Message") (@STRING="@TEXT=IN-COUNTY
GROUNDWATER TRANSPORT COST RESET TO ZERO  (TEXAS_RULE\
001),@OK";))
       )
)

(@RULE=       TEXAS_RULE002
       (@LHS=
              (Yes   (AIDATA_ACQUIRED))
              (=     (<|NPARC|>.NPSIDN)       (99999))
              (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
       )
       (@HYPO=      CNTY_LS_RULE_APPLIED)
       (@RHS=
              (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
```

r10p9.tkb (continued)

```
            (Execute     ("Message") (@STRING="@TEXT=IN-COUNTY LS
TRANSPORT COST RESET TO ZERO  (TEXAS_RULE002),\
@OK";))
      )
)

(@RULE=     TEXAS_RULE003
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (=     (<|NPARC|>.NPSIDN)        (99999))
            (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
            (<>    (<|NPARC|>.NPDIDN)        (993))
      )
      (@HYPO=     CNTY_LSU_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=3,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=IN-COUNTY LS MUN,
CLASS O,\
 AND CLASS A USERS  NOT FEASIBLE  (TEXAS_RULE003),\
@OK";))
      )
)

(@RULE=     REG10_RULE101
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)        (0))
            (<=    (<|NPARC|>.NPSIDN)        (99))
            (=     (<|NPARC|>.NPDIDN)        (992))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CNTYA_GW_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=101,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=GW IMPORT BY CLASS A
USERS  NOT FEASIBLE  (REG10_RULE101),\
@OK";))
      )
)

(@RULE=     REG10_RULE102
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)        (0))
            (<=    (<|NPARC|>.NPSIDN)        (99))
            (=     (<|NPARC|>.NPDIDN)        (993))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CNTYB_GW_RULE_APPLIED)
```

r10p9.tkb (continued)

```
        (@RHS=
                (Execute     ("SetValue")
        (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=102,\
@STRAT=SETFWRD";))
                (Execute     ("Message") (@STRING="@TEXT=GW IMPORT BY CLASS B
USERS  NOT FEASIBLE  (REG10_RULE102),\
@OK";))
        )
)

(@RULE=      REG10_RULE106
        (@LHS=
                (Yes   (AIDATA_ACQUIRED))
                (<     (<|NPARC|>.NPSIDN)        (99900))
                (=     (<|NPARC|>.NPDIDN)        (993))
                (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
        )
        (@HYPO=      CNTYB_SW_RULE_APPLIED)
        (@RHS=
                (Execute     ("SetValue")
        (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=106,\
@STRAT=SETFWRD";))
                (Execute     ("Message") (@STRING="@TEXT=SW IMPORT BY CLASS B
USERS  NOT FEASIBLE  (REG10_RULE106),\
@OK";))
        )
)

(@RULE=      REG10_RULE103
        (@LHS=
                (Yes   (AIDATA_ACQUIRED))
                (=     (<|NPARC|>.NPDIDN)        (757))
                (=     (<|NPARC|>.NPSIDN)        (15))
                (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
        )
        (@HYPO=      CO20_RULE_APPLIED)
        (@RHS=
                (Do    (MIN(<|NPARC|>.NPDCAP,<|NPARC|>.NPSCAP*.2))
        (<|NPARC|>.NPLOWB))
                (Execute     ("Message") (@STRING="@TEXT=CLASS O USERS
DEDICATED UP TO 20PCT OF LOCAL GULF COAST AQUIFER\
  (REG10_RULE103),@OK";))
        )
)

(@RULE=      SUP10_RULE201
        (@LHS=
                (Yes   (AIDATA_ACQUIRED))
                (=     (<|NPARC|>.NPSIDN)        (18100))
                (<>    (<|NPARC|>.NPDIDN)        (992))
        )
        (@HYPO=      COLETTO_CR_RULE_APPLIED)
        (@RHS=
                (Execute     ("SetValue")
```

r10p9.tkb (continued)

```
	(@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=201,\
@STRAT=SETFWRD";))
		(Execute	("Message")	(@STRING="@TEXT=COLETTO CREEK  MUN,
CLASS O,\
 AND CLASS B USERS  NOT FEASIBLE  (SUP10_RULE201),\
@OK";))
	)
)

(@RULE=	DEM10
	(@LHS=
		(Yes	(AIDATA_ACQUIRED))
		(<>	(COMPARE(COMPARE(CB_JACKSON_RULE_APPLIED,TRUE),\
COMPARE(CB_JACKSON_RULE_APPLIED,FALSE)))	(0))
		(<>	(COMPARE(COMPARE(TILDEN_GW_RULE_APPLIED,TRUE),\
COMPARE(TILDEN_GW_RULE_APPLIED,FALSE)))	(0))
	)
	(@HYPO=	DEM10_RULES_APPLIED)
)

(@RULE=	REG10_RULE104
	(@LHS=
		(Yes	(AIDATA_ACQUIRED))
		(>	(<|NPARC|>.NPSIDN)	(0))
		(<=	(<|NPARC|>.NPSIDN)	(99))
	)
	(@HYPO=	GW40_RULE_APPLIED)
	(@RHS=
		(Do	(<|NPARC|>.NPSCAP*.4)	(<|NPARC|>.NPUPPB))
		(Execute	("Message")	(@STRING="@TEXT=INDIVIDUAL GW USERS
LIMITED TO 40PCT OF AVAILABLE SUPPLY  (REG1\
0_RULE104),@OK";))
	)
)

(@RULE=	REG10_RULE105
	(@LHS=
		(Yes	(AIDATA_ACQUIRED))
		(>	(<|NPARC|>.NPSIDN)	(0))
		(<=	(<|NPARC|>.NPSIDN)	(99))
		(<>	(<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)	(0))
	)
	(@HYPO=	GWEX_RULE_APPLIED)
	(@RHS=
		(Do	(<|NPARC|>.NPCOST+<|NPARC|>.NPSCIP*.2)
	(<|NPARC|>.NPCOST))
		(Execute	("Message")	(@STRING="@TEXT=20PCT SURCHARGE ON
COST IN PLACE  APPLIED TO GW EXPORTS  (REG10\
_RULE105),@OK";))
	)
)

(@RULE=	LPSOLV
	(@LHS=
		(Yes	(NETGEND))
```

r10p9.tkb (continued)

```
        )
        (@HYPO=      LPSOLVED)
        (@RHS=
               (Execute     ("Message")  (@STRING="@TEXT=EXECUTE LP
SOLVER,@OK";))
               (Execute     ("jen1")     (@TYPE=EXE;@WAIT=TRUE;))
               (Execute     ("jfb237")   (@TYPE=EXE;@WAIT=TRUE;))
        )
)

(@RULE=      NETGEN
        (@LHS=
               (Yes   (AIDATA_EXPORTED))
        )
        (@HYPO=      NETGEND)
        (@RHS=
               (Execute     ("jfb205b")  (@TYPE=EXE;@WAIT=TRUE;))
        )
)

(@RULE=      REG10
        (@LHS=
               (Yes   (AIDATA_ACQUIRED))
               (<>    (COMPARE(COMPARE(CNTYB_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTYB_GW_RULE_APPLIED,FALSE)))      (0))
               (<>    (COMPARE(COMPARE(CNTYA_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTYA_GW_RULE_APPLIED,FALSE)))      (0))
               (<>
        (COMPARE(COMPARE(GW40_RULE_APPLIED,TRUE),COMPARE(GW40_RULE_APPLIED
,\
FALSE)))       (0))
               (<>
        (COMPARE(COMPARE(CO20_RULE_APPLIED,TRUE),COMPARE(CO20_RULE_APPLIED
,\
FALSE)))       (0))
               (<>
        (COMPARE(COMPARE(GWEX_RULE_APPLIED,TRUE),COMPARE(GWEX_RULE_APPLIED
,\
FALSE)))       (0))
               (<>    (COMPARE(COMPARE(CNTYB_SW_RULE_APPLIED,TRUE),\
COMPARE(CNTYB_SW_RULE_APPLIED,FALSE)))      (0))
        )
        (@HYPO=      REG10_RULES_APPLIED)
)

(@RULE=      SUP10
        (@LHS=
               (Yes   (AIDATA_ACQUIRED))
               (<>    (COMPARE(COMPARE(COLETTO_CR_RULE_APPLIED,TRUE),\
COMPARE(COLETTO_CR_RULE_APPLIED,FALSE)))   (0))
        )
        (@HYPO=      SUP10_RULES_APPLIED)
)
```

r10p9.tkb (continued)

```
(@RULE=     TEX
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (<>    (COMPARE(COMPARE(CNTY_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTY_GW_RULE_APPLIED,FALSE)))     (0))
            (<>    (COMPARE(COMPARE(XCNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(XCNTY_LS_RULE_APPLIED,FALSE)))     (0))
            (<>    (COMPARE(COMPARE(CNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LS_RULE_APPLIED,FALSE)))     (0))
            (<>    (COMPARE(COMPARE(CNTY_LSU_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LSU_RULE_APPLIED,FALSE)))     (0))
      )
      (@HYPO=     TEXAS_RULES_APPLIED)
)

(@RULE=     DEM10_RULE302
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (=     (<|NPARC|>.NPDIDN)      (606))
            (>     (<|NPARC|>.NPSIDN)      (10))
            (<=    (<|NPARC|>.NPSIDN)      (99))
      )
      (@HYPO=     TILDEN_GW_RULE_APPLIED)
      (@RHS=
            (Execute    ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=302,\
@STRAT=SETFWRD";))
            (Execute    ("Message") (@STRING="@TEXT=TILDEN TO GULF
COAST, QUEEN CITY,\
 SPARTA SAND  NOT FEASIBLE  (DEM10_RULE302),\
@OK";))
      )
)

(@RULE=     TEXAS_RULE004
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)      (99900))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
      )
      (@HYPO=     XCNTY_LS_RULE_APPLIED)
      (@RHS=
            (Execute    ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=4,\
@STRAT=SETFWRD";))
            (Execute    ("Message") (@STRING="@TEXT=LOCAL SUPPLY AND
SHORTAGE EXPORT  NOT FEASIBLE  (TEXAS_RULE004)\
,@OK";))
      )
)
```

r10p9.tkb (continued)

```
(@GLOBALS=
      @INHVALUP=FALSE;
      @INHVALDOWN=TRUE;
      @INHOBJUP=FALSE;
      @INHOBJDOWN=FALSE;
      @INHCLASSUP=FALSE;
      @INHCLASSDOWN=TRUE;
      @INHBREADTH=TRUE;
      @INHPARENT=FALSE;
      @PWTRUE=TRUE;
      @PWFALSE=TRUE;
      @PWNOTKNOWN=TRUE;
      @EXHBWRD=TRUE;
      @PTGATES=TRUE;
      @PFACTIONS=TRUE;
      @SOURCESON=TRUE;
      @CACTIONSON=TRUE;
      @SUGLIST=LPSOLVED;
)
```

r10p10.tkb

```
(@VERSION=  020)
(@PROPERTY= NPARC_ID     @TYPE=String;)
(@PROPERTY= NPCOST       @TYPE=Integer;)
(@PROPERTY= NPDCAP       @TYPE=Integer;)
(@PROPERTY= NPDCIP       @TYPE=Integer;)
(@PROPERTY= NPDCNTY      @TYPE=Integer;)
(@PROPERTY= NPDELV       @TYPE=Float;)
(@PROPERTY= NPDIDN       @TYPE=Integer;)
(@PROPERTY= NPDLAT       @TYPE=Float;)
(@PROPERTY= NPDLON       @TYPE=Float;)
(@PROPERTY= NPDTRNS      @TYPE=Integer;)
(@PROPERTY= NPFEAS       @TYPE=Integer;)
(@PROPERTY= NPFLOW       @TYPE=Integer;)
(@PROPERTY= NPFLOWP      @TYPE=Integer;)
(@PROPERTY= NPLENGTH     @TYPE=Float;)
(@PROPERTY= NPLOWB       @TYPE=Integer;)
(@PROPERTY= NPSCAP       @TYPE=Integer;)
(@PROPERTY= NPSCIP       @TYPE=Integer;)
(@PROPERTY= NPSCNTY      @TYPE=Integer;)
(@PROPERTY= NPSELV       @TYPE=Float;)
(@PROPERTY= NPSIDN       @TYPE=Integer;)
(@PROPERTY= NPSLAT       @TYPE=Float;)
(@PROPERTY= NPSLON       @TYPE=Float;)
(@PROPERTY= NPSTRNS      @TYPE=Integer;)
(@PROPERTY= NPUPPB       @TYPE=Integer;)


(@CLASS=     NPARC
      (@PROPERTIES=
            NPARC_ID
            NPCOST
            NPDCAP
            NPDCIP
            NPDCNTY
            NPDELV
            NPDIDN
            NPDLAT
            NPDLON
            NPDTRNS
            NPFEAS
            NPFLOW
            NPFLOWP
            NPLENGTH
            NPLOWB
            NPSCAP
            NPSCIP
            NPSCNTY
            NPSELV
            NPSIDN
            NPSLAT
            NPSLON
            NPSTRNS
            NPUPPB
      )
)
```

r10p10.tkb (continued)

```
(@OBJECT=    AIDATA_ACQUIRED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    AIDATA_EXPORTED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CB_JACKSON_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CC_CA_GOLIAD_TRANSPORT_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CHC_ARANSAS_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CHC_BEE_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CHC_JIM_WELLS_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CHC_KLEBERG_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)

(@OBJECT=    CHC_LIVE_OAK_RULE_APPLIED
        (@PROPERTIES=
                Value @TYPE=Boolean;
        )
)
```

r10p10.tkb (continued)

```
(@OBJECT=    CHC_NUECES_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CHC_SAN_PATRICIO_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTY_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTY_LS_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTY_LSU_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTYA_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTYB_GW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CNTYB_SW_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=    CO20_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)
```

r10p10.tkb (continued)

```
(@OBJECT=   COLETTO_CR_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   DEM10_RULES_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   GW40_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   GWEX_RULE_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   LPSOLVED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   NETGEND
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   PAR10_RULES_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   REG10_RULES_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)

(@OBJECT=   SUP10_RULES_APPLIED
       (@PROPERTIES=
              Value @TYPE=Boolean;
       )
)
```

r10p10.tkb (continued)

```
(@OBJECT=   TEXAS_RULES_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   TILDEN_GW_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@OBJECT=   XCNTY_LS_RULE_APPLIED
      (@PROPERTIES=
            Value @TYPE=Boolean;
      )
)

(@RULE=     ARCINFO_DATA_RETRIEVAL
      (@LHS=
            (Retrieve   ("arcparc.pdb")
      (@TYPE=NXPDB;@FILL=ADD;@NAME="'P'!PARC_ID!";\
@CREATE=|NPARC|;@PROPS=NPARC_ID,NPSLON,NPSLAT,\
NPDLON,NPDLAT,NPSELV,NPDELV,NPSTRNS,NPDTRNS,\
NPSIDN,NPDIDN,NPSCNTY,NPDCNTY,NPSCAP,NPDCAP,\
NPSCIP,NPDCIP,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOWP,NPFLOW,NPFEAS;@FIELDS="PARC_ID","PSLON",\
"PSLAT","PDLON","PDLAT","PSELV","PDELV","PSTRNS",\
"PDTRNS","PSIDN","PDIDN","PSCNTY","PDCNTY",\
"PSCAP","PDCAP","PSCIP","PDCIP","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOWP","PFLOW",\
"PFEAS";))
      )
      (@HYPO=     AIDATA_ACQUIRED)
      (@RHS=
            (Execute    ("Message") (@STRING="@TEXT=ARCINFO  COVERAGES
RETRIEVED,\
@OK";))
      )
)

(@RULE=     EXPORT
      (@LHS=
            (Yes  (TEXAS_RULES_APPLIED))
            (Yes  (REG10_RULES_APPLIED))
            (Yes  (SUP10_RULES_APPLIED))
            (Yes  (DEM10_RULES_APPLIED))
            (Yes  (PAR10_RULES_APPLIED))
      )
      (@HYPO=     AIDATA_EXPORTED)
      (@RHS=
```

```
            (Write       ("nexparc.pdb")
      (@TYPE=NXPDB;@FILL=NEW;@NAME="'P'!PARC_ID!";\
@PROPS=NPARC_ID,NPLENGTH,NPLOWB,NPUPPB,NPCOST,\
NPFLOW,NPFEAS;@FIELDS="PARC_ID","PLENGTH",\
"PLOWB","PUPPB","PCOST","PFLOW","PFEAS";@ATOMS=<<|NPARC|>>;\
))
            (Execute     ("Message") (@STRING="@TEXT=PARED NETWORK
EXPORTED,\
@OK";))
      )
)

(@RULE=     DEM10_RULE301
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPDCNTY)      (120))
            (=    (<|NPARC|>.NPDIDN)       (993))
            (>    (<|NPARC|>.NPSIDN)       (0))
            (<=   (<|NPARC|>.NPSIDN)       (99))
      )
      (@HYPO=     CB_JACKSON_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPUPPB;@STRING="@VALUE=999999,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=JACKSON COUNTY CLASS
B USERS  NOT RESTRICTED BY RULE 104  (DEM1\
0_RULE301),@OK";))
      )
)

(@RULE=     PAR10_RULE401
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)       (18100))
            (=    (<|NPARC|>.NPDIDN)       (992))
            (=    (<|NPARC|>.NPDCNTY)      (88))
      )
      (@HYPO=     CC_CA_GOLIAD_TRANSPORT_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=COLETTO CREEK -
CA_GOLIAD  TRANSPORT COST SET TO ZERO  (PAR10_R\
ULE401),@OK";))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=      PAR10_RULE402
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)       (21101))
            (<=   (<|NPARC|>.NPDIDN)       (757))
            (=    (<|NPARC|>.NPDCNTY)      (4))
      )
      (@HYPO=      CHC_ARANSAS_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)   (<|NPARC|>.NPCOST))
            (Execute     ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO
ARANSAS COUNTY MUNICIPALITIES TRANSPORTATION DI\
SCOUNT APPLIED  (PAR10_RULE402),@OK";))
      )
)

(@RULE=      PAR10_RULE403
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)       (21101))
            (<=   (<|NPARC|>.NPDIDN)       (757))
            (=    (<|NPARC|>.NPDCNTY)      (13))
      )
      (@HYPO=      CHC_BEE_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)   (<|NPARC|>.NPCOST))
            (Execute     ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO BEE
COUNTY MUNICIPALITIES TRANSPORTATION DISCOU\
NT APPLIED  (PAR10_RULE403),@OK";))
      )
)

(@RULE=      PAR10_RULE404
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)       (21101))
            (<=   (<|NPARC|>.NPDIDN)       (757))
            (=    (<|NPARC|>.NPDCNTY)      (125))
      )
      (@HYPO=      CHC_JIM_WELLS_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)   (<|NPARC|>.NPCOST))
            (Execute     ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO JIM
WELLS COUNTY MUNICIPALITIES TRANSPORTATION \
DISCOUNT APPLIED  (PAR10_RULE404),@OK";))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=     PAR10_RULE405
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)      (21101))
            (<=   (<|NPARC|>.NPDIDN)      (757))
            (=    (<|NPARC|>.NPDCNTY)     (137))
      )
      (@HYPO=     CHC_KLEBERG_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)  (<|NPARC|>.NPCOST))
            (Execute    ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO
KLEBERG COUNTY MUNICIPALITIES TRANSPORTATION DI\
SCOUNT APPLIED  (PAR10_RULE405),@OK";))
      )
)

(@RULE=     PAR10_RULE406
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)      (21101))
            (<=   (<|NPARC|>.NPDIDN)      (757))
            (=    (<|NPARC|>.NPDCNTY)     (149))
      )
      (@HYPO=     CHC_LIVE_OAK_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)  (<|NPARC|>.NPCOST))
            (Execute    ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO LIVE
OAK COUNTY MUNICIPALITIES TRANSPORTATION D\
ISCOUNT APPLIED  (PAR10_RULE406),@OK";))
      )
)

(@RULE=     PAR10_RULE407
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)      (21101))
            (<=   (<|NPARC|>.NPDIDN)      (757))
            (=    (<|NPARC|>.NPDCNTY)     (178))
      )
      (@HYPO=     CHC_NUECES_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST*.25)  (<|NPARC|>.NPCOST))
            (Execute    ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO
NUECES COUNTY MUNICIPALITIES TRANSPORTATION DIS\
COUNT APPLIED  (PAR10_RULE407),@OK";))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=      PAR10_RULE408
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (=     (<|NPARC|>.NPSIDN)        (21101))
             (<=    (<|NPARC|>.NPDIDN)        (757))
             (=     (<|NPARC|>.NPDCNTY)       (205))
       )
       (@HYPO=      CHC_SAN_PATRICIO_RULE_APPLIED)
       (@RHS=
             (Do    (<|NPARC|>.NPCOST*.25)   (<|NPARC|>.NPCOST))
             (Execute    ("Message") (@STRING="@TEXT=CHOKE-CORPUS TO SAN
PATRICIO COUNTY MUNICIPALITIES TRANSPORTATI\
ON DISCOUNT APPLIED  (PAR10_RULE408),@OK";\
))
       )
)

(@RULE=      TEXAS_RULE001
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (>     (<|NPARC|>.NPSIDN)        (0))
             (<=    (<|NPARC|>.NPSIDN)        (99))
             (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
       )
       (@HYPO=      CNTY_GW_RULE_APPLIED)
       (@RHS=
             (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
             (Execute    ("Message") (@STRING="@TEXT=IN-COUNTY
GROUNDWATER TRANSPORT COST RESET TO ZERO  (TEXAS_RULE\
001),@OK";))
       )
)

(@RULE=      TEXAS_RULE002
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (=     (<|NPARC|>.NPSIDN)        (99999))
             (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
       )
       (@HYPO=      CNTY_LS_RULE_APPLIED)
       (@RHS=
             (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPCOST;@STRING="@VALUE=0,\
@STRAT=SETFWRD";))
             (Execute    ("Message") (@STRING="@TEXT=IN-COUNTY LS
TRANSPORT COST RESET TO ZERO  (TEXAS_RULE002),\
@OK";))
       )
)
```

r10p10.tkb (continued)

```
(@RULE=     TEXAS_RULE003
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (=     (<|NPARC|>.NPSIDN)        (99999))
            (=     (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
            (<>    (<|NPARC|>.NPDIDN)        (993))
      )
      (@HYPO=     CNTY_LSU_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=3,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=IN-COUNTY LS MUN,
CLASS O,\
 AND CLASS A USERS  NOT FEASIBLE  (TEXAS_RULE003),\
@OK";))
      )
)

(@RULE=     REG10_RULE101
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)        (0))
            (<=    (<|NPARC|>.NPSIDN)        (99))
            (=     (<|NPARC|>.NPDIDN)        (992))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CNTYA_GW_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=101,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=GW IMPORT BY CLASS A
USERS  NOT FEASIBLE  (REG10_RULE101),\
@OK";))
      )
)

(@RULE=     REG10_RULE102
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)        (0))
            (<=    (<|NPARC|>.NPSIDN)        (99))
            (=     (<|NPARC|>.NPDIDN)        (993))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CNTYB_GW_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=102,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=GW IMPORT BY CLASS B
USERS  NOT FEASIBLE  (REG10_RULE102),\
@OK";))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=     REG10_RULE106
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (<    (<|NPARC|>.NPSIDN)        (99900))
            (=    (<|NPARC|>.NPDIDN)        (993))
            (<>   (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CNTYB_SW_RULE_APPLIED)
      (@RHS=
            (Execute    ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=106,\
@STRAT=SETFWRD";))
            (Execute    ("Message") (@STRING="@TEXT=SW IMPORT BY CLASS B
USERS  NOT FEASIBLE  (REG10_RULE106),\
@OK";))
      )
)

(@RULE=     REG10_RULE103
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPDIDN)        (757))
            (=    (<|NPARC|>.NPSIDN)        (15))
            (=    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)       (0))
      )
      (@HYPO=     CO20_RULE_APPLIED)
      (@RHS=
            (Do   (MIN(<|NPARC|>.NPDCAP,<|NPARC|>.NPSCAP*.2))
      (<|NPARC|>.NPLOWB))
            (Execute    ("Message") (@STRING="@TEXT=CLASS O USERS
DEDICATED UP TO 20PCT OF LOCAL GULF COAST AQUIFER\
  (REG10_RULE103),@OK";))
      )
)

(@RULE=     SUP10_RULE201
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (=    (<|NPARC|>.NPSIDN)        (18100))
            (<>   (<|NPARC|>.NPDIDN)        (992))
      )
      (@HYPO=     COLETTO_CR_RULE_APPLIED)
      (@RHS=
            (Execute    ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=201,\
@STRAT=SETFWRD";))
            (Execute    ("Message") (@STRING="@TEXT=COLETTO CREEK  MUN,
CLASS O,\
 AND CLASS B USERS  NOT FEASIBLE  (SUP10_RULE201),\
@OK";))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=     DEM10
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (<>    (COMPARE(COMPARE(CB_JACKSON_RULE_APPLIED,TRUE),\
COMPARE(CB_JACKSON_RULE_APPLIED,FALSE)))   (0))
            (<>    (COMPARE(COMPARE(TILDEN_GW_RULE_APPLIED,TRUE),\
COMPARE(TILDEN_GW_RULE_APPLIED,FALSE)))    (0))
      )
      (@HYPO=      DEM10_RULES_APPLIED)
)

(@RULE=     REG10_RULE104
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (>    (<|NPARC|>.NPSIDN)       (0))
            (<=   (<|NPARC|>.NPSIDN)       (99))
      )
      (@HYPO=      GW40_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPSCAP*.4)    (<|NPARC|>.NPUPPB))
            (Execute     ("Message") (@STRING="@TEXT=INDIVIDUAL GW USERS
LIMITED TO 40PCT OF AVAILABLE SUPPLY  (REG1\
0_RULE104),@OK";))
      )
)

(@RULE=     REG10_RULE105
      (@LHS=
            (Yes  (AIDATA_ACQUIRED))
            (>    (<|NPARC|>.NPSIDN)       (0))
            (<=   (<|NPARC|>.NPSIDN)       (99))
            (<>   (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
      )
      (@HYPO=      GWEX_RULE_APPLIED)
      (@RHS=
            (Do   (<|NPARC|>.NPCOST+<|NPARC|>.NPSCIP*.2)
      (<|NPARC|>.NPCOST))
            (Execute     ("Message") (@STRING="@TEXT=20PCT SURCHARGE ON
COST IN PLACE  APPLIED TO GW EXPORTS  (REG10\
_RULE105),@OK";))
      )
)

(@RULE=     LPSOLV
      (@LHS=
            (Yes  (NETGEND))
      )
      (@HYPO=      LPSOLVED)
      (@RHS=
            (Execute     ("Message") (@STRING="@TEXT=EXECUTE LP
SOLVER,@OK";))
            (Execute     ("jenl")     (@TYPE=EXE;@WAIT=TRUE;))
            (Execute     ("jfb237")   (@TYPE=EXE;@WAIT=TRUE;))
      )
)
```

r10p10.tkb (continued)

```
(@RULE=      NETGEN
       (@LHS=
              (Yes   (AIDATA_EXPORTED))
       )
       (@HYPO=       NETGEND)
       (@RHS=
              (Execute      ("jfb205b") (@TYPE=EXE;@WAIT=TRUE;))
       )
)

(@RULE=      PAR10
       (@LHS=
              (Yes   (AIDATA_ACQUIRED))
              (<>    (COMPARE(COMPARE(CC_CA_GOLIAD_TRANSPORT_RULE_APPLIED,\
TRUE),COMPARE(CC_CA_GOLIAD_TRANSPORT_RULE_APPLIED,\
FALSE)))     (0))
              (<>    (COMPARE(COMPARE(CHC_ARANSAS_RULE_APPLIED,\
TRUE),COMPARE(CHC_ARANSAS_RULE_APPLIED,FALSE)))  (0))
              (<>    (COMPARE(COMPARE(CHC_BEE_RULE_APPLIED,TRUE),\
COMPARE(CHC_BEE_RULE_APPLIED,FALSE)))      (0))
              (<>    (COMPARE(COMPARE(CHC_JIM_WELLS_RULE_APPLIED,\
TRUE),COMPARE(CHC_JIM_WELLS_RULE_APPLIED,\
FALSE)))     (0))
              (<>    (COMPARE(COMPARE(CHC_KLEBERG_RULE_APPLIED,\
TRUE),COMPARE(CHC_KLEBERG_RULE_APPLIED,FALSE)))  (0))
              (<>    (COMPARE(COMPARE(CHC_LIVE_OAK_RULE_APPLIED,\
TRUE),COMPARE(CHC_LIVE_OAK_RULE_APPLIED,FALSE)))       (0))
              (<>    (COMPARE(COMPARE(CHC_NUECES_RULE_APPLIED,TRUE),\
COMPARE(CHC_NUECES_RULE_APPLIED,FALSE)))   (0))
              (<>    (COMPARE(COMPARE(CHC_SAN_PATRICIO_RULE_APPLIED,\
TRUE),COMPARE(CHC_SAN_PATRICIO_RULE_APPLIED,\
FALSE)))     (0))
       )
       (@HYPO=       PAR10_RULES_APPLIED)
)

(@RULE=      REG10
       (@LHS=
              (Yes   (AIDATA_ACQUIRED))
              (<>    (COMPARE(COMPARE(CNTYB_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTYB_GW_RULE_APPLIED,FALSE)))     (0))
              (<>    (COMPARE(COMPARE(CNTYA_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTYA_GW_RULE_APPLIED,FALSE)))     (0))
              (<>
       (COMPARE(COMPARE(GW40_RULE_APPLIED,TRUE),COMPARE(GW40_RULE_APPLIED
,\
FALSE)))     (0))
              (<>
       (COMPARE(COMPARE(CO20_RULE_APPLIED,TRUE),COMPARE(CO20_RULE_APPLIED
,\
FALSE)))     (0))
              (<>
       (COMPARE(COMPARE(GWEX_RULE_APPLIED,TRUE),COMPARE(GWEX_RULE_APPLIED
,\
```

r10p10.tkb (continued)

```
FALSE)))     (0))
             (<>    (COMPARE(COMPARE(CNTYB_SW_RULE_APPLIED,TRUE),\
COMPARE(CNTYB_SW_RULE_APPLIED,FALSE)))     (0))
       )
       (@HYPO=      REG10_RULES_APPLIED)
)

(@RULE=      SUP10
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (<>    (COMPARE(COMPARE(COLETTO_CR_RULE_APPLIED,TRUE),\
COMPARE(COLETTO_CR_RULE_APPLIED,FALSE)))   (0))
       )
       (@HYPO=      SUP10_RULES_APPLIED)
)

(@RULE=      TEX
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (<>    (COMPARE(COMPARE(CNTY_GW_RULE_APPLIED,TRUE),\
COMPARE(CNTY_GW_RULE_APPLIED,FALSE)))      (0))
             (<>    (COMPARE(COMPARE(XCNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(XCNTY_LS_RULE_APPLIED,FALSE)))     (0))
             (<>    (COMPARE(COMPARE(CNTY_LS_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LS_RULE_APPLIED,FALSE)))      (0))
             (<>    (COMPARE(COMPARE(CNTY_LSU_RULE_APPLIED,TRUE),\
COMPARE(CNTY_LSU_RULE_APPLIED,FALSE)))     (0))
       )
       (@HYPO=      TEXAS_RULES_APPLIED)
)

(@RULE=      DEM10_RULE302
       (@LHS=
             (Yes   (AIDATA_ACQUIRED))
             (=     (<|NPARC|>.NPDIDN)       (606))
             (>     (<|NPARC|>.NPSIDN)       (10))
             (<=    (<|NPARC|>.NPSIDN)       (99))
       )
       (@HYPO=      TILDEN_GW_RULE_APPLIED)
       (@RHS=
             (Execute    ("SetValue")
       (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=302,\
@STRAT=SETFWRD";))
             (Execute    ("Message") (@STRING="@TEXT=TILDEN TO GULF
COAST, QUEEN CITY,\
 SPARTA SAND  NOT FEASIBLE  (DEM10_RULE302),\
@OK";))
       )
)
```

r10p10.tkb (continued)

```
(@RULE=     TEXAS_RULE004
      (@LHS=
            (Yes   (AIDATA_ACQUIRED))
            (>     (<|NPARC|>.NPSIDN)        (99900))
            (<>    (<|NPARC|>.NPSCNTY-<|NPARC|>.NPDCNTY)      (0))
      )
      (@HYPO=     XCNTY_LS_RULE_APPLIED)
      (@RHS=
            (Execute     ("SetValue")
      (@ATOMID=<|NPARC|>.NPFEAS;@STRING="@VALUE=4,\
@STRAT=SETFWRD";))
            (Execute     ("Message") (@STRING="@TEXT=LOCAL SUPPLY AND
SHORTAGE EXPORT  NOT FEASIBLE  (TEXAS_RULE004)\
,@OK";))
      )
)

(@GLOBALS=
      @INHVALUP=FALSE;
      @INHVALDOWN=TRUE;
      @INHOBJUP=FALSE;
      @INHOBJDOWN=FALSE;
      @INHCLASSUP=FALSE;
      @INHCLASSDOWN=TRUE;
      @INHBREADTH=TRUE;
      @INHPARENT=FALSE;
      @PWTRUE=TRUE;
      @PWFALSE=TRUE;
      @PWNOTKNOWN=TRUE;
      @EXHBWRD=TRUE;
      @PTGATES=TRUE;
      @PFACTIONS=TRUE;
      @SOURCESON=TRUE;
      @CACTIONSON=TRUE;
      @SUGLIST=LPSOLVED;
)
```

jfb205b.for

```
C     PROGRAM JFB205B.FOR SUPSRC.DCS+DEMAND.DCS+NEXPARC.PDB--->JEN1.DAT
C     ALL ASCII INPUT TO PRODUCE THE JEN1.DAT FILE
      CHARACTER*1 IANS,JANS
      CHARACTER*20 SNAME
      CHARACTER*20 DNAME
C     CHARACTER*20 TNAME
      DIMENSION ISCNTY(199),SLAT(199),SLON(199),SELV(199),ISCAP(199),
     .          ISCIP(199),SNAME(199),ISTRNS(199),ISIDN(199)
      DIMENSION IDCNTY(199),DLAT(199),DLON(199),DELV(199),IDCAP(199),
     .          IDCIP(199),DNAME(199),IDTRNS(199),IDIDN(199)
      DIMENSION ITCNTY(199),TLAT(199),TLON(199),TELV(199),ITCAP(199),
     .ITSU(199),ITCIP(199),             ITTRNS(199),ITIDN(199)
      OPEN(1,FILE='supsrc.dcs')
      OPEN(2,FILE='demand.dcs')
      OPEN(3,FILE='nexparc.pdb')
      OPEN(4,FILE='jen1.dat')
      WRITE(*,10)
 10   FORMAT(1X,  '********************************',/,
     .       1X,  '* PROGRAM JFB205B.FOR           *',/,
     .       1X,  '* NETWORK GENERATOR             *',/,
     .       1X,  '* VERSION 06/01/94              *',/,
     .       1X,  '********************************',///)
C
C
      NL=0
      ISUM=0
      CALL GETDAT(IYR,IMON,IDAY)
      CALL GETTIM(IHR,IMIN,ISEC,I100TH)
      IYR=IYR-1900
      WRITE(4,20) IHR,IMIN,ISEC,IMON,IDAY,IYR
 20   FORMAT(1X,'JFB205B.FOR',2X,
     . I2.2,':',I2.2,':',I2.2,1X,I2.2,'/',I2.2,'/',I2.2)
C
C
      NF=0
      NS=0
      ISTOT=0
 100  CONTINUE
      NS=NS+1
      IF(NS.GT.199)GO TO 190
      READ(1,*,ERR=190,END=190)K,SLON(NS),SLAT(NS),SELV(NS),
     .  ISTRNS(NS),ISIDN(NS),ISCNTY(NS),ISCAP(NS),ISCIP(NS),SNAME(NS)
      IF(ISTRNS(NS).EQ.0)ISTOT=ISTOT+ISCAP(NS)
      GO TO 100
 190  CONTINUE
      NS=NS-1
C
      ND=0
      IDTOT=0
 200  CONTINUE
      ND=ND+1
      IF(ND.GT.199)GO TO 290
      READ(2,*,ERR=290,END=290)K,DLON(ND),DLAT(ND),DELV(ND),
     .IDTRNS(ND),IDIDN(ND),IDCNTY(ND),IDCAP(ND),IDCIP(ND),DNAME(ND)
      IDTOT=IDTOT+IDCAP(ND)
      GO TO 200
```

jfb205b.for (continued)

```
290   CONTINUE
      ND=ND-1
C
300   CONTINUE
      NT=0
      DO 390 ID=1,ND
      IF(IDTRNS(ID).EQ.0)GO TO 390
C
      NT=NT+1
      IF(NT.GT.199)GO TO 390
      ITSU(NT)=ID
CJFB  ITIDN(NT)=IDIDN(ID)
      ITIDN(NT)=99999
CJFB  ITTRNS(NT)=IDTRNS(ID)
      ITTRNS(NT)=0
      ITCNTY(NT)=IDCNTY(ID)
      TLAT(NT)=DLAT(ID)
      TLON(NT)=DLON(ID)
      TELV(NT)=DELV(ID)
CJFB  ITCAP(NT)=IDCAP(ID)
      ITCAP(NT)=999999
      ITCIP(NT)=IDCIP(ID)
C     TNAME(NT)=DNAME(ID)
C
390   CONTINUE
C
400   CONTINUE
      IF(IDTOT.GT.ISTOT)GO TO 850
      N=NS+ND+1
      WRITE(4,410)N,NS,ND,NT
410   FORMAT(4I10,5X,'(N NS ND NT   #NODES)')
C
C     SET UP SUPPLY NODES
      DO 450 I=1,NS
      IX1=I
      IX2=0
      IX3=0
      IX4=0
      WRITE(4,420)IX1,IX2,IX3,IX4,SNAME(I)
420   FORMAT(4I10,A20)
450   CONTINUE
C     SET UP DEMAND NODES
      DO 480 I=1,ND
      IX1=NS+I
      IX2=-(IDCAP(I))
      IX3=0
      IX4=0
      WRITE(4,420)IX1,IX2,IX3,IX4,DNAME(I)
      ISUM=ISUM-IX2
480   CONTINUE
C
```

jfb205b.for (continued)

```
C      SET UP MASTER SUPPLY NODE
       IX1=ND+NS+1
       IX2=ISUM
       IX3=0
       IX4=0
       WRITE(4,495)IX1,IX2,IX3,IX4
495    FORMAT(4I10,'MASTER SUPPLY NODE  ')
       WRITE(4,496)
496    FORMAT(50X,'(BLANK)')
C
500    CONTINUE
C      SET UP SUPPLY NODE CAPACITIES & COSTS
       DO 550 I=1,NS
       IF(ISTRNS(I).EQ.0)GO TO 504
       J=ISTRNS(I)
       IX1=ISTRNS(I)
       IX4=ISCAP(I)
       IX5=ISCIP(I)-ISCIP(J)
       IF(IX5.LT.0)IX5=0
       GO TO 506
504    CONTINUE
       IX1=NS+ND+1
       IX4=ISCAP(I)
       IX5=ISCIP(I)
506    CONTINUE
       IX2=I
       IX3=0
       NL=NL+1
       WRITE(4,510)IX1,IX2,IX3,IX4,IX5
510    FORMAT(5I10)
550    CONTINUE
600    CONTINUE
C      IX1=SOURCE NODE ID  IX2=DEMAND NODE ID  (JEN1.FOR SCHEME)
C      IX3=LOWB  IX4=UPPB  IX5=COST  IX6=FLOW  IX7=FEAS (0=FEASIBLE)
C      IX3=0
C      IX4=999999
C      IX6=0
C      IX7=0
C
       READ(3,601)IANS
       READ(3,601)IANS
601    FORMAT(A1)
C
605    CONTINUE
       READ(3,610,ERR=700)IANS,I,JANS,J,IPLOWB,IPUPPB,IPCOST,IPFEAS
610    FORMAT(22X,A1,I3,A1,I3,16X,1X,I15,1X,I15,1X,I15,16X,1X,I15)
C
C      IF FEAS FLAG GT 0   THEN THE ARC IS INFEASIBLE
       IF(IPFEAS.GT.0)NF=NF+1
       IF(IPFEAS.GT.0)GO TO 605
C
       IF(IANS.EQ.'S')IX1=I
       IF(IANS.NE.'S')IX1=NS+I
       IF(JANS.EQ.'D')IX2=NS+J
       IF(JANS.NE.'D')IX2=0
```

jfb205b.for (continued)

```
C
      WRITE(4,510)IX1,IX2,IPLOWB,IPUPPB,IPCOST
      NL=NL+1
680   CONTINUE
690   CONTINUE
      GO TO 605
C
C
700   CONTINUE
C
      WRITE(4,496)
      WRITE(*,710)NS,N,ND,NL,NT,NF
710   FORMAT(5X,5X,'FROM INPUT ', 5X,5X,'CREATED OUTPUT',/,
     .       5X,5X,'-----------', 5X,5X,'--------------',/,
     .       5X,5X,'SUPSRC.DCS ', 5X,5X,'      JEN1.DAT',/,
     .       5X,5X,'DEMAND.DCS ', 5X,5X,'              ',/,
     .       5X,5X,'NEXPARC.PDB', 5X,5X,'              ',//,
     .       5X,I5,'SUPSRC     ', 5X,I5,'         NODES',/,
     .       5X,I5,'DEMAND     ', 5X,I5,'          ARCS',/,
     .       5X,I5,'TRANSF     ', 5X,I5,' ARCS NOT FEAS')
      GO TO 900
C
850   CONTINUE
      WRITE(*,810) IDTOT,ISTOT
810   FORMAT(1X,'810 ERROR...DEMAND GT SUPPLY',
     .       1X,I10,5X,I10)
C
900   CONTINUE
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
C
C     READ(*,999)IANS
999   FORMAT(A1)
C
      END
      SUBROUTINE GETDAT(IYR,IMO,IDAY)
      CALL IDATE(I,J,K)
      IDAY=I
      IMO=J
      IYR=K
      RETURN
      END
      SUBROUTINE GETTIM(IHR,IMIN,ISEC,I100TH)
      CALL ITIME(I,J,K)
      IHR=I
      IMIN=J
      ISEC=K
      I100TH=0
      RETURN
      END
```

jen1.for

```
C      PROGRAM JEN1.FOR
       IMPLICIT INTEGER*4 (A-Z)
       CHARACTER*72 ADLR
       REAL*4 BTIC,ETIC
       COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .        H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .        PF(195),PR(195),PD(195),PI(195)
       DIMENSION LISA(195),LISN(195)
C
C
C
       OPEN(1,FILE='jen1.dat')
       OPEN(2,FILE='jen1.out')
C
C
C
       WRITE(*,10)
       WRITE(2,10)
10     FORMAT(1X,  '**********************************',/,
     .        1X,  '* PROGRAM JEN1.FOR                *',/,
     .        1X,  '* NETWORK SIMPLEX                 *',/,
     .        1X,  '* VERSION 09/17/92   195X7900     *',/,
     .        1X,  '**********************************',/)
C
       CALL GETDAT(IYR,IMON,IDAY)
       IYR=IYR-1900
       WRITE(*,11)IMON,IDAY,IYR
       WRITE(2,11)IMON,IDAY,IYR
11     FORMAT(1X,'DATE            :',6X,I2.2,'/',I2.2,'/',I2.2)
       CALL GETTIM(IHR,IMIN,ISEC,I100TH)
       WRITE(*,12)IHR,IMIN,ISEC,I100TH
       WRITE(2,12)IHR,IMIN,ISEC,I100TH
12     FORMAT(1X,'START PROCESS:',3X,
     .        I2.2,1H:,I2.2,1H:,I2.2,1H.,I2.2,
     .        10X,'FROM INPUT FILE      : JEN1.DAT')
C
       READ(1,13)IPR,ADLR
13     FORMAT(I1,A)
C
C      IPR=0   NORMAL OUTPUT
C      IPR=1   TRACE SUBROUTINES
C      IPR=2   STEP + DETAILED OUTPUT
C
C
       CALL READJB
C
C
       CALL ARTIFIC
C
C
C
C
       CALL PRIMAL
```

jen1.for (continued)

```
C
C
C
      CALL GETTIM(JHR,JMIN,JSEC,J100TH)
      WRITE(*,910)JHR,JMIN,JSEC,J100TH
      WRITE(2,910)JHR,JMIN,JSEC,J100TH
910   FORMAT(1X,'END    PROCESS:',3X
     .        I2.2,1H:,I2.2,1H:,I2.2,1H.,I2.2,
     .        10X,'CREATED OUTPUT FILE : JEN1.OUT')
      BTIC=FLOAT(I100TH+ISEC*100+IMIN*60*100+IHR*60*60*100)/100.
      ETIC=FLOAT(J100TH+JSEC*100+JMIN*60*100+JHR*60*60*100)/100.
      JHR=0
      JMIN=0
      JSEC=0
      J100TH=0
      ETIC=ETIC-BTIC
911   CONTINUE
      IF(ETIC.LT.3600.)GO TO 912
      JHR=JHR+1
      ETIC=ETIC-3600.
      GO TO 911
912   CONTINUE
913   CONTINUE
      IF(ETIC.LT.60.)GO TO 914
      JMIN=JMIN+1
      ETIC=ETIC-60.
      GO TO 913
914   CONTINUE
915   CONTINUE
      IF(ETIC.LT.1.)GO TO 916
      JSEC=JSEC+1
      ETIC=ETIC-1.
      GO TO 915
916   CONTINUE
C     J100TH=INT2(ETIC*100.+0.5)
      WRITE(*,920)JHR,JMIN,JSEC,J100TH
      WRITE(2,920)JHR,JMIN,JSEC,J100TH
920   FORMAT(1X,'TOTAL ELAPSED:',3X,
     .        I2.2,1H:,I2.2,1H:,I2.2,1H.,I2.2,/)
C
C
      WRITE(2,925)ADLR
925   FORMAT(1X,A)
C
      IADDR=1
      CALL OUT(IADDR,LISA,LISN)
C
      CLOSE(1)
      CLOSE(2)
C
C     READ(*,999)IPR
999   FORMAT(I1)
C
C
C
      END
```

jen1.for (continued)

```
      SUBROUTINE GETDAT(IYR,IMO,IDAY)
      CALL IDATE(I,J,K)
      IDAY=I
      IMO=J
      IYR=K
      RETURN
      END
      SUBROUTINE GETTIM(IHR,IMIN,ISEC,I100TH)
      CALL ITIME(I,J,K)
      IHR=I
      IMIN=J
      ISEC=K
      I100TH=0
      RETURN
      END
      SUBROUTINE PRIMAL
C     FROM FLOWCHART PG. 187  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      IST=1
C
C     SELECT
C
200   CONTINUE
C
      IADDR=46
      IF(IPR.GE.2)CALL SCOST(IADDR)
C
C
      CALL SELECT(IST,KE,DEL,IFIN)
C
C
      IF(IFIN.EQ.1) GO TO 900
C
C     FLOW
C
300   CONTINUE
      IF (KE.GT.0) GO TO 310
      GO TO 350
310   CONTINUE
      IS=T(KE)
      IT=O(KE)
      GO TO 390
```

jen1.for (continued)

```
350   CONTINUE
      IS=O(-KE)
      IT=T(-KE)
390   CONTINUE
      CALL TPATH(IS,IT,LISA,LISN,IC,JUNC,NP)
      IC=IC+1
      LISA(IC)=KE
      CALL MFLO(LISA,IC,MF,KL,ILC)
C
C
      CALL FLOCHG(LISA,IC,MF)
      IF(KL.EQ.KE)GO TO 200
C
C     TREE
C
400   CONTINUE
      IF(ILC.LE.JUNC)GO TO 410
      GO TO 450
410   CONTINUE
      KE=-KE
      DEL=-DEL
      GO TO 490
450   CONTINUE
      KL=-KL
490   CONTINUE
CJFB  WRITE(*,491)KL,KE
491   FORMAT(1X,'CALL TRECHG(KL,KE)',2I5)
      CALL TRECHG(KL,KE)
C
CJFB  CALL JB200
C
C     POTENTIAL
C
500   CONTINUE
      IF(KE.LT.0)GO TO 510
      GO TO 550
510   CONTINUE
      IT=O(-KE)
      GO TO 590
550   CONTINUE
      IT=T(KE)
590   CONTINUE
      CALL ROOT(IT,LISA,LISN,IC,CYC)
CJFB  WRITE(*,591)(LISN(I),I=1,IC+1)
591   FORMAT(1X,10I5)
      DO 595 L=1,IC+1
      I=LISN(L)
      PI(I)=PI(I)+DEL
595   CONTINUE
      GO TO 200
900   CONTINUE
      IF(IPR.GE.2)WRITE(*,995)N,M
995   FORMAT(1X,'PRIMAL(N,M)                           ',2I10)
      IF(IPR.EQ.2)READ(*,999)IPR
999   FORMAT(I1)
```

jen1.for (continued)

```
C
      IADDR=100
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE SELECT(IST,KE,DEL,IFIN)
C     FROM FLOWCHART PG. 189  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      IF(IST.EQ.1)GO TO 110
      GO TO 120
110   CONTINUE
      SN=1
      FN=N
120   CONTINUE
C
C     FIND
C
200   CONTINUE
      DO 290 I=SN,FN
      DEL=0
      KE=0
      CALL ORIG(I,LISA,LISN,IC)
      IF(IC.GT.0)GO TO 210
      GO TO 290
210   CONTINUE
      DO 280 L=1,IC
      K=LISA(L)
      J=LISN(L)
      D=PI(I)+H(K)-PI(J)
C
      IF(D.EQ.0)GO TO 220
      GO TO 221
220   CONTINUE
      GO TO 280
221   CONTINUE
      IF(D.LT.0)GO TO 230
      GO TO 240
230   CONTINUE
      IF(C(K).EQ.F(K))GO TO 231
      GO TO 232
231   CONTINUE
```

jen1.for (continued)

```
      GO TO 280
232   CONTINUE
      IF(D.GT.DEL)GO TO 234
      GO TO 236
234   CONTINUE
      GO TO 280
236   CONTINUE
      DEL=D
      KE=K
      GO TO 280
240   CONTINUE
      IF(F(K).EQ.0.)GO TO 241
      GO TO 250
241   CONTINUE
      GO TO 280
250   CONTINUE
      IF(-D.GT.DEL)GO TO 251
      GO TO 260
251   CONTINUE
      GO TO 280
260   CONTINUE
      DEL=-D
      KE=-K
C
280   CONTINUE
C
C
      IF(KE.EQ.0)GO TO 281
      GO TO 285
281   CONTINUE
      GO TO 290
285   CONTINUE
      IFIN=0
      IST=0
      SN=I+1
      FN=N
      IF(SN.GT.N)GO TO 286
      GO TO 287
286   CONTINUE
      SN=1
      GO TO 289
287   CONTINUE
C
289   CONTINUE
      GO TO 900
C
C
290   CONTINUE
C
C     COMPLETE
C
300   CONTINUE
      IF(SN.EQ.1)GO TO 310
      GO TO 320
310   CONTINUE
```

jen1.for (continued)

```
      IFIN=1
      GO TO 900
320   CONTINUE
      FN=SN-1
      SN=1
      GO TO 200
C
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)IST,KE,DEL,IFIN
995   FORMAT(1X,'SELECT(IST,KE,DEL,IFIN)     ',4I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
C
      IADDR=200
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE TPATH(IS,IT,LISA,LISN,IC,JUNC,NP)
C     FROM FLOWCHART PG. 107  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      II=IS
      IJ=IT
      ICO=1
      ICN=N-1
      JUNC=0
      NP=0
      DDIF=PD(IS)-PD(IT)
C
C     DECIDE
C
200   CONTINUE
      IF(DDIF.EQ.0)GO TO 201
      GO TO 210
201   CONTINUE
      GO TO 500
210   CONTINUE
      IF(DDIF.GT.0)GO TO 211
      GO TO 220
211   CONTINUE
      GO TO 400
220   CONTINUE
      GO TO 300
```

jen1.for (continued)

```
C
C       ITBACK
C
300     CONTINUE
        K=PB(IJ)
        IF(K.EQ.0)GO TO 310
        GO TO 320
310     CONTINUE
        NP=1
        GO TO 900
320     CONTINUE
        IF(K.GT.0)GO TO 321
        GO TO 330
321     CONTINUE
        IJ=O(K)
        GO TO 340
330     CONTINUE
        IJ=T(-K)
340     CONTINUE
        LISA(ICO)=K
        LISN(ICO)=IJ
        ICO=ICO+1
        DDIF=DDIF+1
        GO TO 200
C
C       ISBACK
C
400     CONTINUE
        K=PB(II)
        LISA(ICN)=-K
        LISN(ICN)=II
        ICN=ICN-1
C
        IF(K.EQ.0)GO TO 410
        GO TO 420
410     CONTINUE
        NP=1
        GO TO 900
420     CONTINUE
        IF(K.GT.0)GO TO 421
        GO TO 430
421     CONTINUE
        II=O(K)
        GO TO 440
430     CONTINUE
        II=T(-K)
440     CONTINUE
        DDIF=DDIF-1
        GO TO 200
```

jen1.for (continued)

```
C
C     COMPARE
C
500   CONTINUE
      IF(II.EQ.IJ)GO TO 510
      GO TO 520
510   CONTINUE
      GO TO 530
520   CONTINUE
      GO TO 300
530   CONTINUE
      JUNC=ICO-1
      IC=ICO-1
C
C     COMBINE
C
600   CONTINUE
      IF(ICN.EQ.N-1)GO TO 610
      GO TO 620
610   CONTINUE
      GO TO 900
620   CONTINUE
      IC=IC+1
      ICN=ICN+1
      LISA(IC)=LISA(ICN)
      LISN(IC)=LISN(ICN)
      GO TO 600
C
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)IS,IT,IC,JUNC,NP
995   FORMAT(1X,'TPATH(IS,IT,IC,JUNC,NP)     ',5I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=300
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE MFLO(LISA,IC,MF,KL,ILC)
C     FROM FLOWCHART PG. 122  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
```

jen1.for (continued)

```
C
C       INITIAL
C
100     CONTINUE
        MF=999999
        KL=0
        ILC=0
C
C       FIND
C
200     CONTINUE
        DO 290 L=1,IC
        K=LISA(L)
        IF(K.GT.0)GO TO 210
        GO TO 250
210     CONTINUE
        IF(MF.GT.C(K)-F(K))GO TO 211
        GO TO 220
211     CONTINUE
        MF=C(K)-F(K)
        KL=K
        ILC=L
        GO TO 290
220     CONTINUE
        GO TO 290
250     CONTINUE
        IF(MF.GT.F(-K))GO TO 251
        GO TO 260
251     CONTINUE
        MF=F(-K)
        KL=K
        ILC=L
        GO TO 290
260     CONTINUE
        GO TO 290
290     CONTINUE
900     CONTINUE
        IF(IPR.EQ.1)WRITE(*,995)IC,MF,KL,ILC
995     FORMAT(1X,'MFLO(IC,MF,KL,ILC)          ',4I10)
        IF(IPR.EQ.2)READ(*,998)IPR
998     FORMAT(I1)
        IF(IPR.LT.0)STOP
        IADDR=400
        IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
        RETURN
        END
        SUBROUTINE FLOCHG(LISA,IC,MF)
C       FROM FLOWCHART PG. 122  NETWORK FLOW PROGRAMMING
        IMPLICIT INTEGER*4 (A-Z)
C
C
```

```
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C
C
      DO 100 L=1,IC
      K=LISA(L)
C
      IF(K.GT.0)GO TO 10
      GO TO 20
10    CONTINUE
      F(K)=F(K)+MF
      GO TO 100
20    CONTINUE
      F(-K)=F(-K)-MF
      GO TO 100
100   CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)IC,MF
995   FORMAT(1X,'FLOCHG(IC,MF)                    ',2I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=500
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE TRECHG(KL,KE)
C     FROM FLOWCHART PG. 116  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
      IC=0
C
C
C     DELETE
C
100   CONTINUE
      CALL DELTRE(KL)
```

jen1.for (continued)

```
C
C       FIND
C
200     CONTINUE
        IF(KL.GT.0)GO TO 210
        GO TO 220
210     CONTINUE
        JL=T(KL)
        GO TO 230
220     CONTINUE
        JL=O(-KL)
        GO TO 230
230     CONTINUE
C
        IF(KE.GT.0)GO TO 240
        GO TO 250
240     CONTINUE
        JE=T(KE)
        GO TO 260
250     CONTINUE
        JE=O(-KE)
        GO TO 260
260     CONTINUE
C
C       CHECK
C
300     CONTINUE
        IC=1
        LISA(1)=-KE
        LISN(1)=JE
C
        IF(JE.EQ.JL)GO TO 310
        GO TO 320
310     CONTINUE
C
C
        GO TO 600
320     CONTINUE
        I=JE
        GO TO 400
C
C       OBTAIN
C
400     CONTINUE
        K=PB(I)
        IC=IC+1
        LISA(IC)=K
C
        IF(K.GT.0)GO TO 410
        GO TO 420
410     CONTINUE
        I=O(K)
        GO TO 430
```

jen1.for (continued)

```
420   CONTINUE
      I=T(-K)
      GO TO 430
430   CONTINUE
      LISN(IC)=I
C
      IF(I.EQ.JL)GO TO 500
      GO TO 400
C
C     REVERSE
C
500   CONTINUE
CJB   WRITE(*,991)KL,KE,IC,JL,JE
991   FORMAT(1X,'TR500   KL,KE,IC,JL,JE',5I5)
CJB   WRITE(*,992)(LISA(I),I=1,IC)
992   FORMAT(10I5)
      IF(IPR.EQ.2)READ(*,998)IPR
      IF(IPR.LT.0)STOP
C
C
C
      DO 590 I=2,IC
      LISAJ1=LISA(I)
      LISAJ2=-LISA(I-1)
      CALL DELTRE(LISAJ1)
      CALL ADDTRE(LISAJ2)
590   CONTINUE
C
C     FINISH
C
600   CONTINUE
      LISAJ0=-LISA(IC)
      CALL ADDTRE(LISAJ0)
C
900   CONTINUE
910   CONTINUE
      IF(IPR.GE.2)GO TO 920
      GO TO 999
920   CONTINUE
      WRITE(*,995)KL,KE
      WRITE(2,995)KL,KE
995   FORMAT(1X,'TRECHG(KL,KE)                    ',2I10)
      READ(*,998)IPR
998   FORMAT(I1)
C
999   CONTINUE
      IADDR=600
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
```

jen1.for (continued)

```
      SUBROUTINE ROOT(IROOT,LISA,LISN,IC,CYC)
C     FROM FLOWCHART PG. 108  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      II=IROOT
      IC=0
      LISN(1)=IROOT
      CYC=0
C
C     FORWARD
C
200   CONTINUE
      JJ=PF(II)
C
      IF(JJ.EQ.0)GO TO 210
      GO TO 220
210   CONTINUE
      IF(II.EQ.IROOT)GO TO 211
      GO TO 215
211   CONTINUE
      GO TO 900
215   CONTINUE
      GO TO 300
220   CONTINUE
      GO TO 400
C
C     RIGHT
C
300   CONTINUE
      JJ=PR(II)
C
      IF(JJ.EQ.0)GO TO 310
      GO TO 320
310   CONTINUE
      GO TO 500
320   CONTINUE
      GO TO 400
```

jen1.for (continued)

```
C
C      ADDLST
C
400    CONTINUE
       IC=IC+1
       LISA(IC)=PB(JJ)
       LISN(IC+1)=JJ
       II=JJ
C
       IF(II.EQ.IROOT)GO TO 410
       GO TO 420
410    CONTINUE
       CYC=1
       GO TO 900
420    CONTINUE
       GO TO 200
C
C      BACK
C
500    CONTINUE
       K=PB(II)
C
       IF(K.GT.0)GO TO 510
       GO TO 520
510    CONTINUE
       II=O(K)
       GO TO 530
520    CONTINUE
       II=T(-K)
       GO TO 530
530    CONTINUE
C
       IF(II.EQ.IROOT)GO TO 540
       GO TO 550
540    CONTINUE
       GO TO 900
550    CONTINUE
       GO TO 300
C
900    CONTINUE
CJFB   WRITE(*,995)IROOT,IC,CYC,(LISN(I),I=1,IC+1)
995    FORMAT(1X,'ROOT(IROOT,IC,CYC)     LISN ',3I5,5X,10I5)
       IF(IPR.EQ.2)READ(*,998)IPR
998    FORMAT(I1)
C
       IADDR=700
       IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
       RETURN
       END
```

jen1.for (continued)

```
      SUBROUTINE ORIG(I,LISA,LISN,L)
C     FROM FLOWCHART PG. 103  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C
C
      ISTA=PO(I)
      ISTO=PO(I+1)-1
      L=0
C
      IF(ISTO.GT.ISTA)GO TO 10
      GO TO 20
10    CONTINUE
      DO 19 K=ISTA,ISTO
      L=L+1
      LISA(L)=K
      LISN(L)=T(K)
19    CONTINUE
      GO TO 30
20    CONTINUE
      GO TO 30
30    CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)I,L
995   FORMAT(1X,'ORIG(I,L)                    ',2I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=800
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE DELTRE(KL)
C     FROM FLOWCHART PG. 111  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
C
C
```

jen1.for (continued)

```
C       FORWARD
C
100     CONTINUE
C
        IF(KL.GT.0)GO TO 110
        GO TO 120
110     CONTINUE
        IL=O(KL)
        JL=T(KL)
        GO TO 130
120     CONTINUE
        IL=T(-KL)
        JL=O(-KL)
        GO TO 130
130     CONTINUE
C
        IF(PF(IL).EQ.JL)GO TO 140
        GO TO 150
140     CONTINUE
        PF(IL)=PR(JL)
        GO TO 300
150     CONTINUE
        L=PF(IL)
        GO TO 200
C
C       RIGHT
C
200     CONTINUE
C
        IF(PR(L).EQ.JL)GO TO 210
        GO TO 220
210     CONTINUE
        PR(L)=PR(JL)
        GO TO 300
220     CONTINUE
        L=PR(L)
        GO TO 200
C
C       DELETE
C
300     CONTINUE
        PB(JL)=0
        PR(JL)=0
C
900     CONTINUE
        IF(IPR.EQ.1)WRITE(*,995)KL
995     FORMAT(1X,'DELTRE(KL)                        ',1I10)
        IF(IPR.EQ.2)READ(*,998)IPR
998     FORMAT(I1)
C
        IADDR=900
        IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
        RETURN
        END
```

jen1.for (continued)

```
      SUBROUTINE ADDTRE(KE)
C     FROM FLOWCHART PG. 113  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C     FORWARD
C
100   CONTINUE
      IF(KE.GT.0)GO TO 110
      GO TO 120
110   CONTINUE
      IE=O(KE)
      JE=T(KE)
      GO TO 130
120   CONTINUE
      IE=T(-KE)
      JE=O(-KE)
      GO TO 130
130   CONTINUE
      IF(PF(IE).EQ.0)GO TO 140
      GO TO 150
140   CONTINUE
      GO TO 200
150   CONTINUE
      PR(JE)=PF(IE)
      GO TO 200
C
C     BACK
C
200   CONTINUE
      PF(IE)=JE
      PB(JE)=KE
C
C     DEPTH
C
300   CONTINUE
      PDADJ=PD(IE)-PD(JE)+1
      CALL ROOT(JE,LISA,LISN,IC,CYC)
      DO 390 I=1,IC+1
      PD(LISN(I))=PD(LISN(I))+PDADJ
390   CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)KE
995   FORMAT(1X,'ADDTRE(KE)                                ',1I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
      IADDR=1000
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
```

```
      SUBROUTINE READJB
C     FROM FLOWCHART PG. 101  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
C     PRINT CONTROL LOCATED IN THE FIRST COLUMN OF THE TITLE CARD
C     READ95 MOVED TO MAIN PROGRAM FOR HOUSEKEEPING REASONS
C
C     READ(1,95)IPR
95    FORMAT(I1)
C
C
C     INITIAL
C
100   CONTINUE
C     NMAX IS THE MAXIMUM NUMBER OF NODES
      NMAX=195
C     MMAX IS THE MAXIMUM NUMBER OF ARCS
      MMAX=7900
      M=0
      READ(1,105)N,NS,ND
105   FORMAT(3I10)
      SLACK=N+1
      N=N+1
      IF(N.LT.NMAX)GO TO 109
      WRITE(*,106)
106   FORMAT(///,1X,'ERROR...DIMENSIONS EXCEEDED')
      STOP
109   CONTINUE
      DO 110 I=1,NMAX
      B(I)=0
      PO(I)=0
      PT(I)=0
      PB(I)=0
      PP(I)=0
      PF(I)=0
      PR(I)=0
      PD(I)=0
      PI(I)=0
110   CONTINUE
      DO 120 I=1,MMAX
      O(I)=0
      T(I)=0
      CL(I)=0
      C(I)=0
      H(I)=0
      LT(I)=0
      F(I)=0
```

```
120   CONTINUE
C
C     NODE
C
200   CONTINUE
      READ(1,210)I,BF,BS,CS
210   FORMAT(4I10)
      IF(I.EQ.0)GO TO 220
      GO TO 230
220   CONTINUE
      GO TO 300
230   CONTINUE
      B(I)=BF
      IF(BS.EQ.0)GO TO 240
      GO TO 250
240   CONTINUE
      GO TO 290
250   CONTINUE
      IF(BS.GT.0)GO TO 260
      GO TO 270
260   CONTINUE
      J=I
      I=SLACK
      LOWER=0
      UPPER=BS
      COST=CS
      GO TO 280
270   CONTINUE
      J=SLACK
      LOWER=0
      UPPER=-BS
      COST=CS
280   CONTINUE
      WRITE(*,281)
281   FORMAT(1X,'281')
      CALL ORIGS(I,J,LOWER,UPPER,COST)
      WRITE(*,282)
282   FORMAT(1X,'282')
290   CONTINUE
      GO TO 200
C
C     ARC
C
300   CONTINUE
      READ(1,305)I,J,LOWER,UPPER,COST
305   FORMAT(5I10)
310   CONTINUE
      IF(I.EQ.0)GO TO 320
      GO TO 330
320   CONTINUE
      GO TO 400
330   CONTINUE
      CALL ORIGS(I,J,LOWER,UPPER,COST)
      GO TO 300
```

jen1.for (continued)

```
C
C     EXT
C
400   CONTINUE
      LM=M
      M=0
      DO 410 K=1,LM
      J=T(K)
      M=M+1
      CALL TERMS(K,J)
410   CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)
995   FORMAT(1X,'READJB')
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=1100
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
C
      RETURN
      END
      SUBROUTINE ORIGS(I,J,LOWER,UPPER,COST)
C     FROM FLOWCHART PG. 102  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      NPLUS1=N+1
      IF(M.EQ.0)GO TO 110
      GO TO 120
110   CONTINUE
      DO 115 II=1,NPLUS1
      PO(II)=1
115   CONTINUE
      GO TO 130
120   CONTINUE
      GO TO 130
130   CONTINUE
```

jen1.for (continued)

```
C
C       MOVE
C
200     CONTINUE
        M=M+1
        DO 205 II=I+1,NPLUS1
        PO(II)=PO(II)+1
205     CONTINUE
        IF(PO(I+1).LE.M)GO TO 210
        GO TO 220
210     CONTINUE
        DO 215 L=1,M-PO(I+1)+1
        K=M-L
        O(K+1)=O(K)
        T(K+1)=T(K)
        CL(K+1)=CL(K)
        C(K+1)=C(K)
        H(K+1)=H(K)
215     CONTINUE
220     CONTINUE
C
C       ARC
C
300     CONTINUE
        K=PO(I+1)-1
        O(K)=I
        T(K)=J
        CL(K)=LOWER
        C(K)=UPPER-LOWER
        H(K)=COST
        B(I)=B(I)-LOWER
        B(J)=B(J)+LOWER
900     CONTINUE
        IF(IPR.EQ.1)WRITE(*,995)I,J,LOWER,UPPER,COST
995     FORMAT(1X,'ORIGS(I,J,LOWER,UPPER,COST)',5I10)
        IF(IPR.EQ.2)READ(*,998)IPR
998     FORMAT(I1)
C
        IADDR=1200
        IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
        RETURN
        END
```

jen1.for (continued)

```
      SUBROUTINE TERMS(K,J)
C     FROM FLOWCHART PG. 102  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
C
C
C     INITIAL
C
100   CONTINUE
      IF(M.EQ.1)GO TO 110
      GO TO 120
110   CONTINUE
      DO 115 I=1,N+1
      PT(I)=1
115   CONTINUE
      GO TO 130
120   CONTINUE
      GO TO 130
130   CONTINUE
C
C     MOVE
C
200   CONTINUE
      IF(J.LT.N)GO TO 210
      GO TO 220
210   CONTINUE
      DO 215 JJ=J+1,N
      PT(JJ)=PT(JJ)+1
215   CONTINUE
      GO TO 230
220   CONTINUE
      GO TO 290
230   CONTINUE
      IF(PT(J+1).LE.M)GO TO 240
      GO TO 250
240   CONTINUE
      DO 245 L=1,M-PT(J+1)+1
      KK=M-L
      LT(KK+1)=LT(KK)
245   CONTINUE
      GO TO 290
250   CONTINUE
      GO TO 290
290   CONTINUE
      PT(N+1)=PT(N+1)+1
```

```
C
C     ARC
C
300   CONTINUE
      KK=PT(J+1)-1
      LT(KK)=K
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)K,J
995   FORMAT(1X,'TERMS(K,J)                    ',2I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=1300
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
      SUBROUTINE ARTIFIC
C     FROM FLOWCHART PG. 172  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
      R =9999999
C
C     INITIAL
C
100   CONTINUE
      DO 110 K=1,M
      F(K)=0
110   CONTINUE
      NN=N-1
C
C     ARCS
C
200   CONTINUE
      DO 290 I=1,NN
      IF(B(I).LT.0)GO TO 210
      GO TO 220
210   CONTINUE
      LOWER=0
      UPPER=-B(I)
      II=N
      JJ=I
      GO TO 250
```

jen1.for (continued)

```
220    CONTINUE
       LOWER=0
       UPPER=B(I)
       II=I
       JJ=N
       IF(UPPER.EQ.0)GO TO 230
       GO TO 240
230    CONTINUE
       UPPER=R
       GO TO 250
240    CONTINUE
       GO TO 250
250    CONTINUE
       COST=R
       CALL ORIGS(II,JJ,LOWER,UPPER,COST)
290    CONTINUE
C
C      NEWLSTS
C
300    CONTINUE
       LM=M
       M=0
       DO 310 K=1,LM
       J=T(K)
       M=M+1
       CALL TERMS(K,J)
310    CONTINUE
C
C      FLOWS
C
400    CONTINUE
       NJB=N
       CALL ORIG(NJB,LISA,LISN,L)
       N=NJB
       IF(L.EQ.0)GO TO 410
       GO TO 420
410    CONTINUE
       GO TO 600
420    CONTINUE
       GO TO 500
C
C      POSITIVE
C
500    CONTINUE
       DO 550 I=1,L
       K=LISA(I)
       IF(H(K).LT.R)GO TO 510
       GO TO 520
510    CONTINUE
       GO TO 550
520    CONTINUE
       F(K)=C(K)
       PB(T(K))=K
```

jen1.for (continued)

```
C
550    CONTINUE
C
C      NEGATIVE
C
600    CONTINUE
       CALL TERM(N,LISA,LISN,L)
       IF(L.EQ.0)GO TO 610
       GO TO 620
610    CONTINUE
       GO TO 700
620    CONTINUE
       DO 690 I=1,L
       K=LISA(I)
       IF(H(K).LT.R)GO TO 630
       GO TO 640
630    CONTINUE
       GO TO 690
640    CONTINUE
       F(K)=C(K)
       IF(F(K).GE.R)GO TO 650
       GO TO 660
650    CONTINUE
       F(K)=0
660    CONTINUE
       PB(O(K))=-K
690    CONTINUE
C
C      TREE
C
700    CONTINUE
       PB(N)=0
       CALL TREINT
       NJB=N
       CALL STARTM(NJB)
       N=NJB
900    CONTINUE
       IF(IPR.EQ.1)WRITE(*,995)
995    FORMAT(1X,'ARTIFIC')
       IF(IPR.EQ.2)READ(*,998)IPR
998    FORMAT(I1)
C
       IADDR=1400
       IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
       RETURN
       END
```

jen1.for (continued)

```
      SUBROUTINE STARTM(SN)
C     FROM FLOWCHART PG. 174  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C     TREE
C
100   CONTINUE
      CALL ROOT(SN,LISA,LISN,IC,CYC)
C
C     DUAL
C
200   CONTINUE
      PI(SN)=0
      IF(IC.GT.0)GO TO 210
      GO TO 250
210   CONTINUE
      DO 240 L=1,IC
      K=LISA(L)
      J=LISN(L+1)
      IF(K.GT.0)GO TO 220
      GO TO 230
220   CONTINUE
      I=O(K)
      PI(J)=PI(I)+H(K)
      GO TO 240
230   CONTINUE
      I=T(-K)
      PI(J)=PI(I)-H(-K)
240   CONTINUE
250   CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)SN
995   FORMAT(1X,'STARTM(SN)                    ',1I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=1500
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
```

jen1.for (continued)

```
      SUBROUTINE TREINT
C     FROM FLOWCHART PG. 121  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
C
C
C
C
      DO 100 I=1,N
      PP(I)=I
      PF(I)=0
      PR(I)=0
      PD(I)=0
100   CONTINUE
C
      DO 200 I=1,N
      IF(PB(I).NE.0)GO TO 110
      GO TO 120
110   CONTINUE
      CALL ADDTRE(PB(I))
115   FORMAT(1X,'TREINT                         ',I10)
      GO TO 200
120   CONTINUE
      GO TO 200
200   CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)
995   FORMAT(1X,'TREINT')
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=1600
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
```

jen1.for (continued)

```
      SUBROUTINE TERM(I,LISA,LISN,L)
C     FROM FLOWCHART PG. 103  NETWORK FLOW PROGRAMMING
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C
C
      ISTA=PT(I)
      ISTO=PT(I+1)-1
      L=0
C
      IF(ISTO.GT.ISTA)GO TO 10
      GO TO 20
10    CONTINUE
      DO 19 KK=ISTA,ISTO
      K=LT(KK)
      L=L+1
      LISA(L)=K
      LISN(L)=O(K)
19    CONTINUE
      GO TO 900
20    CONTINUE
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)I,L
995   FORMAT(1X,'TERM(I,L)                          ',2I10)
      IF(IPR.EQ.2)READ(*,998)IPR
998   FORMAT(I1)
C
      IADDR=1700
      IF(IPR.GE.2)CALL OUT(IADDR,LISA,LISN)
C
      RETURN
      END
```

jen1.for (continued)

```
      SUBROUTINE OUT(IADDR,LISA,LISN)
      IMPLICIT INTEGER*4 (A-Z)
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
      DIMENSION LISA(195),LISN(195)
C
C
C
C
C     WRITE(2,10)IADDR
10    FORMAT(' IADDR= ',I5,///)
20    CONTINUE
C
50    CONTINUE
90    FORMAT('      B     PO     PT  LISN     PB     PP     PF     PR',
     .'    PD  LISA           PI')
C
110   CONTINUE
      IF(IPR.NE.1)WRITE(2,140)
140   FORMAT(///)
      WRITE(2,150)
150   FORMAT(4X,'O',4X,'T',4X,'S',4X,'D',8X,'DH',5X,'PI(I)',9X,'H',
     .       5X,'PI(J)',9X,'F')
      DO 200 I=1,M+2
CJFB  IF(F(I)+CL(I).EQ.0)GO TO 200
      IF(O(I).LE.NS)JN=O(I)
      IF(O(I).GT.NS)JN=-1*(O(I)-NS)
      IF(O(I).GT.NS+ND)JN=0
      IF(T(I).LE.NS)KN=0
      IF(T(I).GT.NS)KN=T(I)-NS
      IF(T(I).GT.NS+ND)KN=0
C
      PIOI=PI(O(I))
      PITI=PI(T(I))
      DH=PIOI+H(I)-PITI
C
190   CONTINUE
CJFB  WRITE(*,195)O(I),T(I),JN,KN,DH,PIOI,H(I),PITI,F(I)+CL(I)
      WRITE(2,195)O(I),T(I),JN,KN,DH,PIOI,H(I),PITI,F(I)+CL(I)
195   FORMAT(4I5,5I10)
200   CONTINUE
205   CONTINUE
C
      CALL SCOST(IADDR)
900   CONTINUE
      IF(IPR.EQ.1)WRITE(*,995)
995   FORMAT(1X,'OUT')
      RETURN
      END
```

jen1.for (continued)

```
      SUBROUTINE SCOST(IADDR)
      IMPLICIT INTEGER*4 (A-Z)
      CHARACTER*11 ADLR
      CHARACTER*40 BDLR
C
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
      DIMENSION INUM(10)
      DIMENSION JNUM(10)
      DIMENSION KNUM(40)
C
C
      NFEAS=1
      MMAX=7900
C
C
C
      DO 10 I=1,40
      KNUM(I)=0
      BDLR(I:I)='0'
10    CONTINUE
C
C
      DO 800 II=1,MMAX
C
      IF(H(II).EQ.9999999 .AND. F(II)+CL(II).NE.0) NFEAS=0
C
      DO 16 I=1,10
      INUM(I)=0
      JNUM(I)=0
16    CONTINUE
      WRITE(ADLR,20)F(II) + CL(II)
20    FORMAT(I10)
      KB=11
      DO 25 I=1,10
      IB=11-I
      IF(ADLR(IB:IB).EQ.' ')GO TO 25
      KB=KB-1
      READ(ADLR(IB:IB),21)INUM(KB)
21    FORMAT(I1)
25    CONTINUE
      WRITE(ADLR,20)H(II)
      KB=11
      DO 45 I=1,10
      IB=11-I
      IF(ADLR(IB:IB).EQ.' ')GO TO 45
      KB=KB-1
      READ(ADLR(IB:IB),21)JNUM(KB)
45    CONTINUE
```

```
      DO 200 J=1,10
      JB=11-J
      DO 100 I=1,10
      IB=11-I
      L1=INUM(IB)
      L2=JNUM(JB)
      IF(L2.EQ.0)GO TO 200
      IF(L1.EQ.0)GO TO 100
      KB=40 -(10-JB)-(10-IB)
      IF(KB.LE.0)GO TO 100
      CALL MULT(L1,L2,MT,MU)
      L1=MU
      L2=KNUM(KB)
      CALL ADD(L1,L2,NT,NU)
      KNUM(KB)=NU
      KB=KB-1
      IF(KB.EQ.0)GO TO 100
      L1=MT
      L2=NT
      CALL ADD(L1,L2,NT,NU)
      L1=NU
      L2=KNUM(KB)
      CALL ADD(L1,L2,NT,NU)
      KNUM(KB)=NU
50    CONTINUE
      IF(NT.EQ.0)GO TO 100
      KB=KB-1
      IF(KB.EQ.0)GO TO 100
      L1=NT
      L2=KNUM(KB)
      CALL ADD(L1,L2,NT,NU)
      KNUM(KB)=NU
      GO TO 50
100   CONTINUE
200   CONTINUE
800   CONTINUE
900   CONTINUE
      IF(NFEAS.EQ.0)GO TO 905
      WRITE(*,901)
      WRITE(2,901)
901   FORMAT(1X,'FEASIBLE')
      GO TO 910
905   CONTINUE
      WRITE(*,906)
      WRITE(2,906)
906   FORMAT(1X,'INFEASIBLE')
910   CONTINUE
      DO 916 I=1,40
      WRITE(BDLR(I:I),915)KNUM(I)
915   FORMAT(I1)
916   CONTINUE
```

jen1.for (continued)

```
      DO 918 I=1,40
      IF(KNUM(I).NE.0)GO TO 919
      BDLR(I:I)=' '
918   CONTINUE
919   CONTINUE
      WRITE(*,920)IADDR,BDLR
920   FORMAT(1X,'IADDR  SCOST',I10,5X,A)
      WRITE(2,921)IADDR,(KNUM(I),I=1,40)
921   FORMAT(1X,'IADDR  SCOST',I10,5X,40I1)
C
      RETURN
      END
      SUBROUTINE MULT(L1,L2,MT,MU)
      MT=L1*L2/10
      MU=L1*L2-MT*10
      RETURN
      END
      SUBROUTINE ADD(L1,L2,NT,NU)
      NT=(L1+L2)/10
      NU=L1+L2-NT*10
      RETURN
      END
      SUBROUTINE JB200
      IMPLICIT INTEGER*4 (A-Z)
C
      COMMON IPR,N,NS,ND,M,B(195),T(7900),O(7900),CL(7900),C(7900),
     .       H(7900),PO(7900),PT(195),LT(7900),F(7900),PB(195),PP(195),
     .       PF(195),PR(195),PD(195),PI(195)
C
      DIMENSION BRN(15),BRB(15),FL(15)
C
C
C
C     DETERMINE THE ENDS OF THE BRANCHES
C
      WRITE(*,50)
50    FORMAT(1X,'JB200:')
      NB=0
100   CONTINUE
      DO 150 I=1,N
      IF(PF(I).NE.0)GO TO 150
      NB=NB+1
      IF(NB.GT.15)GO TO 900
      BRN(NB)=I
150   CONTINUE
C
      DO 155 I=1,NB
      FL(I)=0
155   CONTINUE
C
C
160   CONTINUE
      WRITE(*,180)(BRN(I),I=1,NB)
180   FORMAT(1X,15(I4,'N'))
```

jen1.for (continued)

```
      SUM=0
      DO 185 I=1,NB
      BRB(I)=0
      IF(BRN(I).EQ.0)GO TO 185
      BRB(I)=PB(BRN(I))
      SUM=SUM+ABS(BRB(I))
185   CONTINUE
C
      IF(SUM.EQ.0)GO TO 900
      WRITE(*,190)(BRB(I),I=1,NB)
190   FORMAT(1X,15(I4,'A'))
C
C
C
200   CONTINUE
C
C
      DO 220 I=1,NB
      CN=BRN(I)
      BRN(I)=0
      DO 210 J=1,N
204   CONTINUE
      DO 205 K=1,N
      IF(PR(K).EQ.0)GO TO 205
      IF(PR(K).NE.CN)GO TO 205
      CN=K
      FL(I)=1
      GO TO 204
205   CONTINUE
206   CONTINUE
      IF(PF(J).EQ.0)GO TO 210
      IF(PF(J).EQ.CN)BRN(I)=J
210   CONTINUE
220   CONTINUE
C
      GO TO 160
C
900   CONTINUE
      WRITE(*,910)(FL(I),I=1,NB)
C
910   FORMAT(1X,15I5)
      RETURN
      END
```

jfb237.for

```
C      PROGRAM JFB237.FOR  JEN1.OUT    ----->  jen1.pdb
C
       CHARACTER*1 SDLR,DDLR
       OPEN(1,FILE='jen1.out')
       OPEN(2,FILE='jen1.pdb')
C
       WRITE(*,10)
10     FORMAT(1X,  '*******************************',/,
     .        1X,  '* PROGRAM JFB237.FOR           *',/,
     .        1X,  '* UPDATE  PROCESSOR            *',/,
     .        1X,  '* VERSION 06/17/94             *',/,
     .        1X,  '*******************************',///)
C
       DO 20,I=1,17
       READ(1,15)SDLR
15     FORMAT(A1)
20     CONTINUE
       WRITE(2,25)
25     FORMAT(15X,'          PARC_ID','|',
     .            '            PFLOW','|')
       WRITE(2,26)
26     FORMAT(47('*'))
       DDLR='D'
100    CONTINUE
       SDLR='S'
       READ(1,110,END=800)I,J,IFLOW
110    FORMAT(12X,I3,3X,I3,40X,I10)
C
       IF(I.EQ.0)     GO TO 800
       IF(I.LT.0)     SDLR='D'
       IF(I.LT.0)     I=-1*I
       IF(J.EQ.0)     GO TO 100
       IF(IFLOW.LE.0) GO TO 100
       WRITE(2,115)SDLR,I,DDLR,J,IFLOW
115    FORMAT(15X,7X,A1,I3.3,A1,I3.3,'|',I15,'|')
       IF(IFLOW.EQ.0)GO TO 100
C
       GO TO 100
800    CONTINUE
       WRITE(2,26)
       WRITE(*,899)
899    FORMAT(5X,5X,'FROM INPUT', 5X,5X,'CREATED OUTPUT',/,
     .        5X,5X,'----------', 5X,5X,'--------------',/,
     .        5X,5X,'  JEN1.OUT', 5X,5X,'      JEN1.PDB',/)
900    CONTINUE
       CLOSE(1)
       CLOSE(2)
       END
```

import.aml

```
/*  import.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/***********************************************************************
/*
/* USAGE: This aml imports the nexparc.pdb  ASCII datafile into
/*        ARCINFO table  parc.aat
/*
/*
/*
/***********************************************************************
/*
/*  INVOKED BY: [ARC] &run import.aml
/*  RELATED COVERAGES:  PARC
/*  RELATED FILES:  nexparc.pdb
/*
/***********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> PARC   Coverage modification in progress'
&type '<> Processing'
/* &MESSAGES &OFF &ALL
/* &SEVERITY &WARNING &IGNORE
/* &SEVERITY &ERROR &IGNORE
TABLES
SELECT PARC.AAT
CALC PFLOWP = PFLOW
CALC PFLOW = 0
Q STOP
&SYS nxpinfo jen1.pdb parc.aat PARC_ID
&type '<> Complete.    jen1.pdb   to PARC Coverage.'
&type '                Updated PFLOWP (previous allocations).'
&type ' '
QUIT
```

nxpinfo.c

```
/*
*  nxpinfo.c
*  V 1.0
*  8/24/93
*  Tom Evans
*  Center for Research in Water Resources, The University of Texas at Austin
*/
/****************************************************************************
*  nxpinfo.c --  reads item names, widths, and data from an nxpdb file and
*  writes the data to an INFO table.  The program takes as arguments the names
*  of the nxpdb file, the INFO table, and an item to be used as a key thusly:
*
*  USAGE:  nxpinfo nxpdb_filename info_tablename key_item
*
*  For each line (record) in the nxpdb file, nxpinfo will seek a record in the
*  INFO table
*  nxpinfo expects that the names of the items in the nxpdb file and the
*  INFO table will be the same and that the data types will be compatible.
*  It is the responsibility of the user to make sure that the INFO table
*  exists and has the appropriate item definitions.
****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "infolib.h"

#define FatalError( mess ) \
     { \
          fprintf( stderr, "%s\n", mess ); \
          fflush( stdout ); \
          fflush( stderr ); \
          exit( 1 ); \
     }

/* structure definitions */

/* the nxp_item structure holds the name of the item, its field width in the
   nxpdb file, a format control string for input,
   and a pointer to another nxp_item structure,
   so that the items can be strung together in a linked list. */

struct  nxp_item{
  char  namest[17]; /* room for 16-character-item name, the longest
                       item name that INFO permits*/
  char  *valst;     /* pointer to a string that holds the value of the item */
  char  skip_flag;  /* flag for invalid item */
  int  inwi;        /* width of input field */
  InfoItemDef   *INFO_def; /* pointer INFO definition of the item */
  struct nxp_item *next;  /* it's a linked list now */
  };

/* type declarations */

/*  NXP_ELEMENT and NXP_LINK defined for list handling */

typedef  struct nxp_item  NXP_ELEMENT;
typedef  NXP_ELEMENT  *NXP_LINK;
```

infonxp.c (continued)

```
/* global variable declarations */

FILE *nxpfp;           /* the nxpdb file containing the data */
NXP_LINK  key_item;    /* pointer to the nxp_item defined as key */
InfoFile  *INFO_fp;    /* INFO file for output */

/* function declarations */

void  define_nxp_item();
void  define_INFO_items();
void  encode_data_list();
int  read_nxp_line();
int  get_nxp_item();
NXP_LINK  make_nxp_list();

/*****************************************************************************
*  MAIN PROGRAM
*****************************************************************************/

main(argc, argv)
  int  argc;
  char  *argv[];
{
  int  lastli = 0;       /* flag for last line in nxpdb file */
  int  linewi = 0;       /* number of characters in one line of nxpdb file */
  long int  reccntli = 0;    /* number of records read from nxpdb file */
  long int  ereccntli = 0;   /* number of records in nxpdb file that match
                                existing INFO records (e for existing) */
  long int  nreccntli = 0;   /* number of records in nxpdb file that do not
                                match existing INFO records (n for new) */
  long int  keyli = 0;       /* the value of the key item if it's an integer */
  double  valdb = 0;         /* double equivalent of keyli */
  char  c = '\0';            /* a character */
  NXP_LINK  item_list = (NXP_LINK) NULL; /* pointer to list of items */
  long int  INFO_nrecli = 0;  /* Number of records in the INFO table */
  long int  INFO_recnli = 0;  /* Number of INFO record matching key value */

  /* exit with usage message if called with wrong number of arguments */
  if (argc != 4){
    fprintf(stderr, "USAGE:  %s %s %s %s\n", argv[0], "<nxpdb_filename>",
      "<info_table_name>", "<key_item>");
    fflush(stdout);
    fflush(stderr);
    exit(1);}

  /* open the nxpdb file for input READ ONLY */
  if((nxpfp = fopen(argv[1], "r")) == NULL)
    FatalError("Unable to open nxpdb file.  Exiting.")

  /* create the nxpdb item list */
  key_item = NULL;
  item_list = make_nxp_list(argv);

  /* if nxpdb file contains no items, bail out */
  if(item_list == NULL)
    FatalError("No items found in nxpdb file.")
```

infonxp.c (continued)

```
/* if the key item can't be found in the nxpdb file, bail out */
if(key_item == NULL){
  fprintf(stderr,"\n%s%s\n%s%s%s\n%s%s%s\n", "key item :  ", argv[3],
    "not present in nxpdb file:  ", argv[1], ".", "Exiting ", argv[0], ".");
  fflush(stdout);
  fflush(stderr);
  exit(1);}

/* eat the line of asterisks (2nd line) and count the line width */
while((c = getc(nxpfp)) == '*')linewi++;

/* open the INFO file with WRITE access */
if((INFO_fp = InfoOpenFile(argv[2], InfoWRITE))
  == (InfoFile *) NULL)
  FatalError("Unable to open INFO file.")

/* get the INFO item definitions from the INFO table and attach them to
   the items in the nxpdb item list.  */
define_INFO_items(item_list);

/* find out how many records there are in the INFO table */
INFO_nrecli = INFO_fp->NumberRecords;

/* read item values from the nxpdb file one line at a time and place the
   values in the valst elements of the item list.  Until the last line
   in the nxpdb file is encountered, search the INFO table for a record
   with the same value in the key item as the one read from the nxpdb
   file, and update existing records or create new ones to match contents
   of the nxpdb file.  */
while((lastli = read_nxp_line(item_list)) == 0)
{ /* begin data transfer loop */

  /* count the number of records read from the nxpdb file. */
  reccntli++;

  /* if the key is an integer, set keyli equal to its value */
  if (key_item->INFO_def->ItemType == INFO_INTEGER_TYPE ||
      key_item->INFO_def->ItemType == INFO_BINARY_TYPE)
     sscanf(key_item->valst, "%ld", &keyli);

  /* if an INFO record that matches the key value of the current nxpdb
     record can be found, update the items in that record with data from
     the nxpdb file. */
  if(InfoSeqSearch(INFO_fp, key_item->INFO_def, key_item->valst, keyli,
                   &INFO_recnli))
  { /* begin update block */
    /* count the number of records updated in the INFO table */
    ereccntli++;

    /* flush the INFO IOBuffer */
    InfoFileFlush(INFO_fp);

    /* encode the data into the INFO IOBuffer */
    encode_data_list(item_list);

    /* write the INFO IOBuffer to a new record */
    if(! InfoWriteRecord(INFO_fp, INFO_recnli)){
       fprintf(stderr,"Unable to write new record to INFO file.\n");
       fflush(stderr);}
  } /* end update block */
```

infonxp.c (continued)

```
    /* if no INFO record can be found to match the key value of the current
       nxpdb record, add a record to the end of the INFO table and write the
       values from the current nxpdb item to it. */
    else{  /* begin new record block */
      /* count the number of records added to the INFO table */
      nreccntli++;

      /* flush the INFO IOBuffer */
      InfoFileFlush(INFO_fp);

      /* if the key is an integer, put its value into valdb */
      if(key_item->INFO_def->ItemType == INFO_INTEGER_TYPE ||
         key_item->INFO_def->ItemType == INFO_BINARY_TYPE)
        sscanf(key_item->valst, "%lf", &valdb);

      /* encode the key value into the INFO IOBuffer */
      InfoEncode(INFO_fp, key_item->INFO_def, key_item->valst, valdb);

      /* encode the other data into the INFO IOBuffer */
      encode_data_list(item_list);

      /* write the INFO IOBuffer to a new record */
      if(! InfoWriteRecord(INFO_fp, ++INFO_nrecli)){
         fprintf(stderr,"Unable to write new record to INFO file.\n");
         fflush(stderr);}

    } /* end new record block */

  } /* end data transfer loop */

  /* tell the user how many records were read and written */
  printf("\n%s%ld%s%s%s\n%s%ld%s\n%s%ld%s%s%s\n",
       "Read ", reccntli, " records from nxpdb file ", argv[1], ".",
       "Updated ", ereccntli, " existing records and",
       "created ", nreccntli, " new records in INFO table ", argv[2],".");
  fflush(stdout);

  /* close the INFO file */
  if( ! InfoCloseFile( INFO_fp ))
    FatalError( "Error closing INFO file." )

  /* close the nxpdb file */
  if(fclose(nxpfp) != 0)
     FatalError("Error closing nxpdb file.")

}  /* end main program */
```

infonxp.c (continued)

```
/*****************************************************************************
*  FUNCTION make_nxp_list
*
*  Creates a linked list of nxp_item structures and returns a pointer to
*  the first item in the list
*****************************************************************************/

NXP_LINK make_nxp_list(argv)
  char  *argv[];
{
  char  c;
  NXP_LINK  first_item, head;

  /* point first_item at newly-allocated space for the first
    nxp_item structure and set its values */
  first_item = (NXP_LINK)malloc(sizeof(NXP_ELEMENT));
  if(first_item == NULL)
    FatalError("Cannot allocate suficient memory for nxpinfo.  Bailing out.")
  define_nxp_item(first_item);

  /* Return NULL if first line of nxpdb file is blank.  */
  if(first_item->namest[0] == '\n' || first_item->namest[0] == EOF)
    return(NULL);

  /* if the first nxp_item is the key, point the key_item pointer at it. */
  if(strcmp(first_item->namest, argv[3]) == 0)
    key_item = first_item;

  /* Point head pointer to first item to set up for item definition loop.  */
  head = first_item;

  /* until the end of the first line of the nxpdb file is reached,
     keep adding new items to the list and defining their characteristics.
     THIS RELIES ON THE NXPDB FILE TO BE PROPERLY STRUCTURED, WITH NO
     CHARACTERS BETWEEN THE LAST '|' AND THE 'NEWLINE' ON THE FIRST (OR
     ANY SUBSEQUENT) LINE.  */
  while((c = getc(nxpfp)) != '\n'){
    ungetc(c, nxpfp);
    head->next=(NXP_LINK)malloc(sizeof(NXP_ELEMENT));
    if(head->next == NULL)
      FatalError("Cannot allocate suficient memory for nxpinfo.  Bailing out.")
    head = head->next;
    define_nxp_item(head);
    /* if this nxp_item is the key, point the key_item pointer at it. */
    if(strcmp(head->namest, argv[3]) == 0)
      key_item = head;}

  /* set the next pointer to NULL on the last item */
  head->next = NULL;

  /* return the pointer to the head of the list */
  return(first_item);

 }  /* end function make_nxp_list */
```

infonxp.c (continued)

```
/******************************************************************************
*  FUNCTION define_nxp_item
*
*  Extracts an item's name and width from the first line of an nxpdb file
*  and assigns those values to info_item structure elements.
******************************************************************************/

void  define_nxp_item(head)
  NXP_LINK head;
{
  char  c;
  int  ccnti = 0, bflagi = 0, lflagi = 0;

  /* initialize the elements of the nxp_item structure */
  head->namest[0] = '\0';
  head->valst = (char *) NULL;
  head->skip_flag = 0;
  head->inwi = 0;
  head->INFO_def = (InfoItemDef *) NULL;
  head->next = (NXP_LINK) NULL;

  /* count leading blanks as part of the total width of item
     without writing them to the name string. */
  while((c = getc(nxpfp)) == ' ')
    head->inwi++;

  /* if the delimiter is the first nonblank character encountered, set the
     item name to "null string" and skip_flag to 1 for invalid item. */
  if(c == '|'){
    head->namest[0] = '\0';
    head->skip_flag = 1;
    return;}

  /* if first nonblank character is 'newline', set item name to "newline" and
     skip_flag to 1 for invalid. */
  if(c == '\n'){
    head->namest[0] = '\n';
    head->namest[1] = '\0';
    head->skip_flag = 1;
    return;}

  /* if first nonblank character is 'end of file', set item name to "EOF" and
     skip_flag to 1 for invalid. */
  if(c == EOF){
    head->namest[0] = EOF;
    head->namest[1] = '\0';
    head->skip_flag = 1;
    return;}

  /* otherwise, write first nonblank character to item name and increment the
     item width by one. */
  head->namest[ccnti] = c;
  head->namest[++ccnti] = '\0';
  head->inwi++;
```

1.0 HARDWARE & SYSTEM REQUIREMENTS

The Texas Water Development Board Automated Allocation System (AAS) was developed and tested on SUN Sparc Desktop Workstations. These computers come standard with 40 Mhz combined integer and floating-point processors and have 48 megabytes of main memory and 424 megabytes of internal disk memory. The computers were equipped with external hard disk and mass storage devices and were connected to local area networks. The operating system for the computers was SunOS 4.1.1 running OpenWindows Version 3. This is a multi-tasking, multi-user, UNIX windows environment.

The AAS creates several large intermediate data files and in the examples tested to date requires up to 25 megabytes of hard drive memory for each saved plan or project scenario.

2.0 SOFTWARE REQUIREMENTS

Two commercially available software products are required to run the AAS. The first product is the Geographic Information System (GIS) Arc/Info, which is used to store, modify, and select the large amounts of data required for each study. The second is the Expert System Nexpert Object, which is used to implement rules that modify the potential arc network.

3.0 INSTALLATION

The FORTRAN programs and Advanced Macro Language (AML) scripts listed below must be installed in the users directory.

File	Description
demand.aml	Automates creation of the demand coverage.
export.aml	Exports data from GIS potential arc coverage.
import.aml	Imports data into GIS potential arc coverage.
nxpinfo.aml	Facilitates data transfer from GIS to Expert System.
parc.aml	Automates creation of the potential arc coverage.
plot10.aml	Produces a plotfile for printer graphics.
show10.aml	Displays results to the screen in graphical form.
supsrc.aml	Automates creation of the supply source coverage.
jen1.for	Solves Network LP problem.

infonxp.c (continued)

```
  /* allocate memory for the values to be read from the nxpdb file for
     this item and initialize the string to null */
  head->valst = malloc((head->inwi + 1) * sizeof(char));
  if(head->valst == NULL)
    FatalError("Cannot allocate sufficient memory for nxpinfo.  Bailing out.")
  head->valst[0] = '\0';

  return;
}  /* end function define_nxp_item */

/*****************************************************************************
*  FUNCTION define_INFO_items
*
*  Retrieves item definition information from the INFO table for all the items
*  listed in the nxpdb file.
*****************************************************************************/

void  define_INFO_items(head)
  NXP_LINK  head;
{
  /* if end of list reached, return without doing anything more. */
  if(head == NULL)
    return;

  /* if this is an invalid item, go on to the next one. */
  if(head->skip_flag){
    define_INFO_items(head->next);
    return;}

  /* if the item name is OK, look it up in the INFO table */
  head->INFO_def = InfoGetItemDef(INFO_fp, head->namest);

  /* if the item is not in the table, quit if its the key, or mark it
     as invalid if its not the key. */
  if(head->INFO_def  == (InfoItemDef *) NULL){
    if(head == key_item)
      FatalError("Key item not found in INFO file.  Exiting from nxpinfo.")
    else{
      fprintf(stderr,"The item named %s cannot be found in the INFO file.\n",
        head->namest);
      fflush(stderr);
      head->skip_flag = 1;}}

  /* if the key item is not a string or an integer, bail out */
  if(head == key_item
     && head->INFO_def->ItemType != INFO_INTEGER_TYPE
     && head->INFO_def->ItemType != INFO_BINARY_TYPE
     && head->INFO_def->ItemType != INFO_CHARACTER_TYPE)
     FatalError("Key item must be INTEGER, or CHARACTER type.")

  /* get the definition of the next item in the list */
  define_INFO_items(head->next);

  return;
}  /* end function define_INFO_items */
```

infonxp.c (continued)

```
/*****************************************************************************
*  FUNCTION read_nxp_line
*
*  Reads a line of |-delimited fields from the nxpdb file and places them in
*  the valst elements of the item list.  Returns 1 if every character in
*  the present item and all subsequent items in the line is '*'.  Returns
*  0 otherwise.
*****************************************************************************/

int read_nxp_line(head)
  NXP_LINK  head;
{
  int  aflag = 1;  /* flag for last row in nxpdb file (all *) */
  char c;

  /* if the end of the item list is reached, there should only be a
     'newline' remaining on the line.  */
  if(head == NULL){
    if((c = getc(nxpfp)) != '\n')
      FatalError("Uneven line encountered in nxpdb input file.")
    /* if proper 'newline' found, return aflag true (1), since this is
       consistent with a correctly constructed last line. */
    else return(aflag);}

  /* if any non-* characters appear in the present item or anywhere in the
     rest of the present line, set iflag false (0) */
  if(get_nxp_item(head->valst, head->inwi) == 0) aflag = 0;
  if(read_nxp_line(head->next) == 0) aflag = 0;

  return(aflag);
}  /* end function read_nxp_line */

/*****************************************************************************
*  FUNCTION get_nxp_item
*
*  Extracts one item from an ascii data file delimited by | characters.
*  Writes the item to the string pointed to by the argument.  Used for reading
*  data from nxpdb database files.  Returns 1 if every character in the item
*  is an '*', returns 0 otheriwse.
*****************************************************************************/

int  get_nxp_item(string, itemwi)
  char  *string;  /* pointer to string for holding item value */
  int  itemwi;    /* maximum number of characters allowed in item string */

{
  int  ccnt = 0, tcnt =0;
  int  aflag = 1;  /* flag for last line in nxpdb file (all*) will be
                      set to 0 if any non-* characters appear in item */
  char  c;
```

infonxp.c (continued)

```
/* eliminate leading blanks. */
while((c = getc(nxpfp)) == ' ')
  tcnt++;

/* string is null if '|' is the first nonblank character encountered. */
if(c == '|'){
  string[0] = '\0';
  return(0);}

/* set string to "newline" if first nonblank character is 'newline'. */
if(c == '\n'){
  string[0] = '\n';
  string[1] = '\0';
  return(0);}

/* set string to "EOF" if first nonblank character is 'end of file'. */
if(c == EOF){
  string[0] = EOF;
  string[1] = '\0';
  return(0);}

/* otherwise, write first character to string.  */
string[ccnt] = c;
string[++ccnt] = '\0';

/* if any blanks have been found or if the present character is not an
   '*' this is not the last line of the nxpdb file. */
if(tcnt > 0 || c != '*') aflag = 0;
tcnt++;

/* after the first nonblank character, add characters to string until
   '|' is encountered or the total character count (including leading
   blanks) exceeds the item's defined width.  */
while((c=getc(nxpfp)) != '|' && tcnt < itemwi){

  /* if the present character is not an '*' this is not the last line of
     the nxpdb file. */
  if(c != '*') aflag = 0;

  /* set string to "newline" if 'newline' is encountered before '|'. */
  if(c == '\n'){
    string[0] = '\n';
    string[1] = '\0';
    return(0);}

  /* return EOF if end of file is encountered before '|'. */
  if(c == EOF){
    string[0] = EOF;
    string[1] = '\0';
    return(0);}

  /* if the string is OK so far, add the character to the string. */
  string[ccnt] = c;
  string[++ccnt] = '\0';
  tcnt++;}

 return(aflag);

}  /* end function get_nxp_item */
```

infonxp.c (continued)

```
/*******************************************************************************
*  FUNCTION encode_data_list
*
*  Writes encoded data to the INFO IOBuffer for the data contained in a line
*  of the nxpdb file.
*******************************************************************************/

void encode_data_list(head)
  NXP_LINK head;

{
  double  valdb = 0;  /* a long float for holding numerical values */

  /* if end of list is reached, take no action */
  if(head == (NXP_LINK) NULL)
    return;

  /* if the item is invalid, or if it is the key, skip it and go on to the
     next item */
  if(head->skip_flag == 1 || head == key_item){
    encode_data_list(head->next);
    return;}

  /* if the item type is numerical, copy the value to valdb */
  if(head->INFO_def->ItemType == INFO_INTEGER_TYPE ||
     head->INFO_def->ItemType == INFO_NUMBER_TYPE ||
     head->INFO_def->ItemType == INFO_BINARY_TYPE ||
     head->INFO_def->ItemType == INFO_FLOATING_TYPE)
    sscanf(head->valst, "%lf", &valdb);

  /* encode the value into the INFO IOBuffer */
  if(! InfoEncode(INFO_fp, head->INFO_def, head->valst, valdb)){
    fprintf(stderr, "\nError encoding info item %s.\n", head->namest);
    fflush(stderr);}

  /* go on to the next item in the list */
  encode_data_list(head->next);
  return;

}  /* end function encode_data_list */

/* end of nxpinfo.c file */
```

show10.aml

```
/*    SHOW10.AML   (06/01/94)
/*  V 1.0
/*  06/01/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/*************************************************************************
/*
/* USAGE: This aml displays the PARC  coverage on a graphics terminal
/*
/*
/*
/*************************************************************************
/*
/*  INVOKED BY: [ARC] &run show10.aml
/*  RELATED COVERAGES:  basemap
/*                      parc
/*  RELATED FILES:
/*
/*************************************************************************
&LABEL TOP
DISP 9999 SIZE 450 450 POSITION 200 0
&LABEL T1
CLEAR
LINECOLOR 1
MAPEX BASEMAP
POLYS BASEMAP
MOVE 3.75 4.00
TEXT 'TWDB'
MOVE 3.75 3.75
TEXT 'REGION 10'
MOVE 3.75 3.5
TEXT 'COASTAL BEND'
&LABEL T2
MARKERCOLOR 3
MARKERPATTERN 2
POINTS SUPSRC
MARKER 2.75 1.50
MOVE 3.00 1.50
TEXT 'Supply'
&LABEL T3
MARKERCOLOR 3
MARKERPATTERN 0
POINTS DEMAND
MARKER 2.75 1.25
MOVE 3.00 1.25
TEXT 'Demand'
MARKERPATTERN 6
MARKER 2.75 1.00
MOVE 3.00 0.97
TEXT 'Local Allocation'
```

show10.aml (continued)

```
LINE 2.50 0.75 2.75 0.75
LINE 2.70 0.77 2.75 0.75
LINE 2.70 0.73 2.75 0.75
MOVE 3.00 0.72
TEXT 'Distant Allocation'
&LABEL T4
MARKERPATTERN 6
&SETVAR V1 := [RESPONSE 'Enter C P Snnn Dnnn I R Nn']
&IF %V1% CN 'Q' &THEN
  &GOTO BOTTOM
  &ELSE
&IF %V1% CN 'C' &THEN
  &GOTO T1
  &ELSE
&IF %V1% CN 'P' &THEN
  &GOTO T8
  &ELSE
&IF %V1% CN 'D' &THEN
  &GOTO T5
  &ELSE
&IF %V1% CN 'S' &THEN
  &GOTO T5
  &ELSE
&IF %V1% CN 'I' &THEN
  &GOTO T6
  &ELSE
&IF %V1% CN 'R' &THEN
  &GOTO T7
  &ELSE
&IF %V1% CN 'N' &THEN
  &GOTO T9
  &ELSE
&GOTO BOTTOM
&LABEL T5
LINECOLOR 5
MARKERCOLOR 5
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5 AND LENGTH >
0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5 AND LENGTH <
0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOW
&GOTO T4
&LABEL T6
LINECOLOR 4
MARKERCOLOR 4
RESELECT PARC ARCS
```

show10.aml (continued)

```
ASELECT PARC ARCS PFLOW > PFLOWP AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW > PFLOWP AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW > PFLOWP
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFLOW
&GOTO T4
&LABEL T7
LINECOLOR 2
MARKERCOLOR 2
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFLOW
&GOTO T4
&LABEL T8
LINECOLOR 7
MARKERCOLOR 7
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5 AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5 AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP
&GOTO T4
&LABEL T9
LINECOLOR 2
MARKERCOLOR 2
&IF [LENGTH %V1%] = 1 &THEN
  &GOTO T91
  &ELSE
  &GOTO T92
&LABEL T91
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS > 0 AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
```

show10.aml (continued)

```
ASELECT PARC ARCS PFEAS > 0 AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS > 0
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFEAS
&GOTO T4
&LABEL T92
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N] AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N] AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N]
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFEAS
&GOTO T4
&LABEL BOTTOM
QUIT
QUIT
```

plot10.aml

```
/*    PLOT10.AML   (06/01/94)
/*  V 1.0
/*  06/01/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/*********************************************************************
/*
/* USAGE: This aml creates pl1.ai    an Adobe Illustrator file that can
/*        plotted using appropriate software and an HP Laser Jet Printer
/*
/*
/*********************************************************************
/*
/*  INVOKED BY: [ARC] &run plot10.aml
/*  RELATED COVERAGES:  basemap
/*                      parc
/*                      parc
/*  RELATED FILES:  pl1.ai
/*
/*********************************************************************
&LABEL TOP
DISP 1040 3
pl1.ai
&LABEL T1
CLEAR
LINECOLOR 1
MAPEX BASEMAP
MAPLIMITS 1.32 2.01 6.73 7.42
UNITS PAGE
PAGESIZE 7.68 10.16
POLYS BASEMAP
TEXTSIZE 0.25
MOVE 5.75 6.00
TEXT 'TWDB'
MOVE 5.75 5.75
TEXT 'REGION 10'
MOVE 5.75 5.5
TEXT 'COASTAL BEND'
&LABEL T2
MARKERCOLOR 3
MARKERPATTERN 2
POINTS SUPSRC
MARKER 4.75 3.50
MOVE 5.00 3.50
TEXT 'Supply'
&LABEL T3
MARKERCOLOR 3
MARKERPATTERN 0
POINTS DEMAND
MARKER 4.75 3.25
MOVE 5.00 3.25
TEXT 'Demand'
```

plot10.aml (continued)

```
MARKERPATTERN 6
MARKER 4.75 3.00
MOVE 5.00 2.97
TEXT 'Local Allocation'
LINE 4.50 2.75 4.75 2.75
LINE 4.70 2.77 4.75 2.75
LINE 4.70 2.73 4.75 2.75
MOVE 5.00 2.72
TEXT 'Distant Allocation'
&LABEL T4
MARKERPATTERN 6
&SETVAR V1 := [RESPONSE 'Enter C P Snnn Dnnn I R Nn']
&IF %V1% CN 'Q' &THEN
  &GOTO BOTTOM
  &ELSE
&IF %V1% CN 'C' &THEN
  &GOTO T1
  &ELSE
&IF %V1% CN 'P' &THEN
  &GOTO T8
  &ELSE
&IF %V1% CN 'D' &THEN
  &GOTO T5
  &ELSE
&IF %V1% CN 'S' &THEN
  &GOTO T5
  &ELSE
&IF %V1% CN 'I' &THEN
  &GOTO T6
  &ELSE
&IF %V1% CN 'R' &THEN
  &GOTO T7
  &ELSE
&IF %V1% CN 'N' &THEN
  &GOTO T9
  &ELSE
&GOTO BOTTOM
&LABEL T5
LINECOLOR 5
MARKERCOLOR 5
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5 AND LENGTH >
0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5 AND LENGTH <
0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PARC_ID CN [QUOTE %V1%] AND PFLOW > 0.5
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOW
&GOTO T4
```

plot10.aml (continued)

```
&LABEL T6
LINECOLOR 4
MARKERCOLOR 4
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW > PFLOWP AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW > PFLOWP AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW > PFLOWP
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFLOW
&GOTO T4
&LABEL T7
LINECOLOR 2
MARKERCOLOR 2
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOW < PFLOWP
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFLOW
&GOTO T4
&LABEL T8
LINECOLOR 7
MARKERCOLOR 7
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5 AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5 AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFLOWP > 0.5
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP
&GOTO T4
&LABEL T9
LINECOLOR 2
MARKERCOLOR 2
&IF [LENGTH %V1%] = 1 &THEN
  &GOTO T91
  &ELSE
  &GOTO T92
&LABEL T91
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS > 0 AND LENGTH > 0.001
ARCS PARC
```

plot10.aml (continued)

```
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS > 0 AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS > 0
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFEAS
&GOTO T4
&LABEL T92
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N] AND LENGTH > 0.001
ARCS PARC
ARCARROWS PARC
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N] AND LENGTH < 0.001
ARCMARKERS PARC 1000 MIDDLE
RESELECT PARC ARCS
ASELECT PARC ARCS PFEAS = [TRIM %V1% -LEFT N]
LIST PARC ARCS PARC_ID PSCIP PCOST PFLOWP PFEAS
&GOTO T4
&LABEL BOTTOM
DISP 9999
QUIT
QUIT
```

jfb206b.for

```
C      PROGRAM JFB206B.FOR  SUPSRC.DCS+DEMAND.DCS+JEN1.OUT-->TTY
       CHARACTER*1 IANS
       CHARACTER*20 ADLR
       CHARACTER*20 BDLR
       CHARACTER*20 CDLR
       CHARACTER*20 DDLR
       DIMENSION SLON(199),SLAT(199),SELV(199),ISIDN(199),
      .ISTRNS(199),ISCNTY(199),ISCAP(199),ISCIP(199),ADLR(199)
       DIMENSION DLON(199),DLAT(199),DELV(199),IDIDN(199),
      .IDTRNS(199),IDCNTY(199),IDCAP(199),IDCIP(199),BDLR(199)
       DIMENSION DDLR(199),TLAT(199),TLON(199),ITTRNS(199)
C      HARDCOPY FLAG
       IHARD=1
       OPEN(1,FILE='supsrc.dcs')
       OPEN(2,FILE='demand.dcs')
       OPEN(3,FILE='jen1.out')
       IF(IHARD.EQ.1)OPEN(4,FILE='jfb206.ou1')
       IF(IHARD.EQ.1)OPEN(5,FILE='jfb206.ou2')
       WRITE(*,10)
 10    FORMAT(1X,     '*********************************',/,
      .       1X,     '* PROGRAM JFB206B.FOR           *',/,
      .       1X,     '* NETWORK POSTPROCESSING        *',/,
      .       1X,     '* VERSION 06/01/94              *',/,
      .       1X,     '*********************************',///)
C
       NS=0
       ISTOT=0
 100   CONTINUE
       NS=NS+1
       IF(NS.GT.199)GO TO 130
       READ(1,*,ERR=130,END=130)I,SLON(NS),SLAT(NS),SELV(NS),
      .ISTRNS(NS),ISIDN(NS),ISCNTY(NS),ISCAP(NS),ISCIP(NS),ADLR(NS)
       ISTOT=ISTOT+ISCAP(NS)
       GO TO 100
 130   CONTINUE
       NS=NS-1
C
       ND=0
       NT=0
       IDTOT=0
 140   CONTINUE
       ND=ND+1
       IF(ND.GT.199)GO TO 160
       READ(2,*,ERR=160,END=160)I,DLON(ND),DLAT(ND),DELV(ND),
      .IDTRNS(ND),IDIDN(ND),IDCNTY(ND),IDCAP(ND),IDCIP(ND),BDLR(ND)
       IDTOT=IDTOT+IDCAP(ND)
       IF(IDTRNS(ND).EQ.0)GO TO 140
C
       NT=NT+1
       IF(NT.GT.199)GO TO 160
       TLAT(NT)=DLAT(ND)
       TLON(NT)=DLON(ND)
       ITTRNS(NT)=ND
       DDLR(NT)=BDLR(ND)
       GO TO 140
```

jfb206b.for (continued)

```
C
160   CONTINUE
      ND=ND-1
C
C
300   CONTINUE
      IF(IHARD.EQ.0)GO TO 320
      DO 310 I=1,NS
      IS=I
      WRITE(4,405)
      WRITE(4,410) ADLR(IS),SELV(IS),           ISCIP(IS),ISCAP(IS)
      REWIND(3)
      DO 301 K=1,17
      READ(3,414)IANS
301   CONTINUE
C
      NSUM=0
302   CONTINUE
      READ(3,420,ERR=304)IR,ID,IC,IF
      IF(IS.NE.IR)GO TO 302
      IF(IF.LE.0) GO TO 302
      NSUM=NSUM+IF
      WRITE(4,430)BDLR(ID),IC,IF
      GO TO 302
304   CONTINUE
      WRITE(4,455)
      WRITE(4,460)NSUM
305   CONTINUE
310   CONTINUE
315   CONTINUE
C
      DO 319 I=1,NT
      IS=ITTRNS(I)
      ITX=-9999
      WRITE(4,405)
      WRITE(4,410) BDLR(IS),DELV(IS),           ITX,ITX
      IS=-IS
      REWIND(3)
      DO 316 K=1,17
      READ(3,414)IANS
316   CONTINUE
C
      NSUM=0
317   CONTINUE
      READ(3,420,ERR=318)IR,ID,IC,IF
      IF(IS.NE.IR)GO TO 317
      IF(IF.LE.0)GO TO 317
      NSUM=NSUM+IF
      WRITE(4,430)BDLR(ID),IC,IF
      GO TO 317
318   CONTINUE
      WRITE(4,455)
      WRITE(4,460)NSUM
319   CONTINUE
      GO TO 500
```

jfb206b.for (continued)

```
C
C
320   CONTINUE
      WRITE(*,321)
321   FORMAT(1X,'ENTER SUPSRC NAME:')
      READ(*,330)CDLR
330   FORMAT(A20)
      IF(CDLR(1:1).EQ.' ')GO TO 500
      IF(CDLR(1:4).EQ.'QUIT')GO TO 900
C
      DO 335 I=1,20
      IF(CDLR(I:I).NE.' ')L=I
335   CONTINUE
      DO 340 I=1,NS
      IS=I
CJFB  L=LEN_TRIM(CDLR)
      K= INDEX(ADLR(IS),CDLR(1:L))
      IF(K.EQ.1)GO TO 400
340   CONTINUE
      DO 350 I=1,NT
      IS=ITTRNS(I)
      IT=I
      K= INDEX(BDLR(IS),CDLR(1:L))
      IF(K.EQ.1)GO TO 411
350   CONTINUE
      WRITE(*,395)
395   FORMAT(1X,'NO MATCH FOUND')
      GO TO 320
400   CONTINUE
      WRITE(*,405)
405   FORMAT(/,1X,'NAME',20X,'    ELEV          CIPL        TRAN  CAPACITY')
      WRITE(*,410) ADLR(IS),SELV(IS),           ISCIP(IS),ISCAP(IS)
410   FORMAT(1X,A20,1F12.4,2X,I10,10X,I10)
      GO TO 413
411   CONTINUE
      ITX=-9999
      WRITE(*,405)
      WRITE(*,412) BDLR(IS),DELV(IS),           ITX,ITX
412   FORMAT(1X,A20,1F12.4,2X,I10,10X,I10)
      IS=-IS
413   CONTINUE
      REWIND(3)
      DO 415 I=1,17
      READ(3,414)IANS
414   FORMAT(A1)
415   CONTINUE
```

jfb206b.for (continued)

```
      NSUM=0
416   CONTINUE
      READ(3,420,ERR=450)IR,ID,IC,IF
420   FORMAT(10X,I5,I5,20X,I10,10X,I10)
      IF(IS.NE.IR)GO TO 416
      IF(IF.EQ.0)GO TO 416
      NSUM=NSUM+IF
      WRITE(*,430)BDLR(ID),IC,IF
430   FORMAT(2X,A20,23X,I10,I10)
      GO TO 416
450   CONTINUE
      WRITE(*,455)
455   FORMAT(/,55X,'----------')
      WRITE(*,460)NSUM
460   FORMAT(55X,I10)
      GO TO 320
500   CONTINUE
      IF(IHARD.EQ.0)GO TO 515
      DO 510 I=1,ND
      ID=I
      WRITE(5,605)
      WRITE(5,610) BDLR(ID),DELV(ID),          IDCAP(ID)
      REWIND(3)
      DO 501 K=1,17
      READ(3,414)IANS
501   CONTINUE
C
      NSUM=0
      ICIPX=-9999
502   CONTINUE
      READ(3,620,ERR=504)IS,IR,IC,IF
      IF(ID.NE.IR)GO TO 502
      IF(IF.LE.0)GO TO 502
      NSUM=NSUM+IF
      IF(IS.GT.0)WRITE(5,630)ADLR(IS),ISCIP(IS),IC,IF
      IF(IS.LT.0)WRITE(5,630)BDLR(ABS(IS)),ICIPX,IC,IF
      GO TO 502
504   CONTINUE
      WRITE(5,455)
      WRITE(5,460)NSUM
505   CONTINUE
510   CONTINUE
      GO TO 900
515   CONTINUE
      WRITE(*,520)
520   FORMAT(1X,'ENTER DEMAND NAME:')
      READ(*,530)CDLR
530   FORMAT(A20)
      IF(CDLR(1:1).EQ.' ')GO TO 300
      IF(CDLR(1:4).EQ.'QUIT')GO TO 900
C
```

jfb206b.for (continued)

```
      DO 535 I=1,20
      IF(CDLR(I:I).NE.' ')L=I
535   CONTINUE
      DO 590 I=1,ND
CJFB  L=LEN_TRIM(CDLR)
      K= INDEX(BDLR(I),CDLR(1:L))
      IF(K.EQ.1)GO TO 600
590   CONTINUE
      WRITE(*,595)
595   FORMAT(1X,'NO MATCH FOUND')
      GO TO 515
600   CONTINUE
      ID=I
      WRITE(*,605)
605   FORMAT(/,1X,'NAME',20X,'    ELEV          CIPL         TRAN       DEMAND')
      WRITE(*,610) BDLR(ID),DELV(ID),             IDCAP(ID)
610   FORMAT(1X,A20,1F12.4,12X,10X,I10)
      REWIND(3)
      DO 615 I=1,17
      READ(3,414)IANS
615   CONTINUE
C
      NSUM=0
      ICIPX=-9999
616   CONTINUE
      READ(3,620,ERR=650)IS,IR,IC,IF
620   FORMAT(10X,I5,I5,20X,I10,10X,I10)
      IF(ID.NE.IR)GO TO 616
      IF(IF.EQ.0)GO TO 616
      NSUM=NSUM+IF
      IF(IS.GT.0)WRITE(*,630)ADLR(IS),ISCIP(IS),IC,IF
      IF(IS.LT.0)WRITE(*,630)BDLR(ABS(IS)),ICIPX,IC,IF
630   FORMAT(2X,A20,13X,I10,I10,I10)
      GO TO 616
650   CONTINUE
      WRITE(*,655)
655   FORMAT(//,55X,'----------')
      WRITE(*,660)NSUM
660   FORMAT(55X,I10)
      GO TO 515
900   CONTINUE
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      IF(IHARD.EQ.1)CLOSE(4)
      IF(IHARD.EQ.1)CLOSE(5)
      END
```

reset.aml

```
/*  reset.aml
/*  V 1.0
/*  08/31/94
/*  John F. Burgin  Research Associate
/*  Center for Research in Water Resources
/*  The University of Texas at Austin
/***********************************************************************
/*
/* USAGE: This aml resets the PARC   coverage to original values
/*
/*
/*
/*
/***********************************************************************
/*
/*  INVOKED BY: [ARC] &run reset.aml
/*  RELATED COVERAGES:  parc
/*  RELATED FILES:
/*
/***********************************************************************
&type ' '
&type '<> TWDB Automated Allocation System'
&type '<> PARC  Coverage reset in progress'
&type '<> Processing'
&MESSAGES &OFF &ALL
&SEVERITY &WARNING &IGNORE
&SEVERITY &ERROR &IGNORE
&SYS rm nexparc.pdb
&SYS rm jen1.pdb
TABLES
SELECT PARC.AAT
CALC PFEAS = 0
CALC PFLOW = 0
CALC PLOWB = PLOWBO
CALC PUPPB = PUPPBO
CALC PCOST = PCOSTO
Q STOP
&type '<> Complete'
&type ' '
QUIT
```

APPENDIX B

TEXAS AUTOMATED ALLOCATION SYSTEM USER'S MANUAL

Table of Contents

1.0 HARDWARE & SYSTEM REQUIREMENTS

The Texas Water Development Board Automated Allocation System (AAS) was developed and tested on SUN Sparc Desktop Workstations. These computers come standard with 40 Mhz combined integer and floating-point processors and have 48 megabytes of main memory and 424 megabytes of internal disk memory. The computers were equipped with external hard disk and mass storage devices and were connected to local area networks. The operating system for the computers was SunOS 4.1.1 running OpenWindows Version 3. This is a multi-tasking, multi-user, UNIX windows environment.

The AAS creates several large intermediate data files and in the examples tested to date requires up to 25 megabytes of hard drive memory for each saved plan or project scenario.

2.0 SOFTWARE REQUIREMENTS

Two commercially available software products are required to run the AAS. The first product is the Geographic Information System (GIS) Arc/Info, which is used to store, modify, and select the large amounts of data required for each study. The second is the Expert System Nexpert Object, which is used to implement rules that modify the potential arc network.

3.0 INSTALLATION

The FORTRAN programs and Advanced Macro Language (AML) scripts listed below must be installed in the users directory.

File	Description
demand.aml	Automates creation of the demand coverage.
export.aml	Exports data from GIS potential arc coverage.
import.aml	Imports data into GIS potential arc coverage.
nxpinfo.aml	Facilitates data transfer from GIS to Expert System.
parc.aml	Automates creation of the potential arc coverage.
plot10.aml	Produces a plotfile for printer graphics.
show10.aml	Displays results to the screen in graphical form.
supsrc.aml	Automates creation of the supply source coverage.
jen1.for	Solves Network LP problem.

jfb204f.for	Develops the intermediate files necessary for the automatic creation of the potential arc coverage.
jfb205b.for	Preprocesses data for LP solver.
jfb206b.for	Displays and prints results in tabular form.
jfb237.for	Postprocesses output from the LP solver.

The five FORTRAN programs must be compiled to produce executable versions, following standard UNIX naming conventions, the executable versions of each program must the same name as the source FORTRAN without the 'dot for' extender.

The files listed below are not required. They may be installed, if so desired, in the u directory to duplicate the case study that appears in the report.

File	Description
supsrc.dcs	ASCII data file containing supply source information.
demand.dcs	ASCII data file containing demander information.
r10p0.tkb	Expert system knowledge base with plan 0 rules.
r10p4.tkb	Expert system knowledge base with plan 4 rules.
r10p9.tkb	Expert system knowledge base with plan 9 rules.
r10p10.tkb	Expert system knowledge base with plan 10 rules.

4.0 SEQUENCE OF OPERATIONS

The steps listed below establish a framework for execution of the AAS. Once the fun of each step is clear to the user, the collection of programs, datafiles and scripts may be empl in sequences other than the one detailed below. Table 1 illustrates the sequence of command: are described below. Table 2 and Figures 1 through 5 show the input files, programs, and o files used or created during each step in the process.

4.1. ESTABLISH GIS COVERAGES

The first step in the process involves the creation of two files: supsrc.dcs and demanc These files are comma separated, ASCII data and contain one line for each potential supplie one line for each potential demander. The mechanism for creating the files is optional. They be manually created with a text editor, they may be created from automated export processes involving databases, or they may be derived from merge operations that deal with multiple lists.

The files may be transformed into GIS coverages by entering the Arcinfo environment and executing the appropriate GIS macros (supsrc.aml and demand.aml).

System	Command
<UNIX.system.prompt> :	**arc**
arc :	**&run supsrc.aml**
<UNIX.system.prompt> :	**arc**
arc :	**&run demand.aml**

The Parc GIS Coverage is created by executing FORTRAN program jfb204f and GIS macro parc.aml. The FORTRAN program combines the information in files supsrc.dcs and demand.dcs to create the intermediate ASCII files parc.d00 and parc.d01. These, in turn, are used by the GIS macro to create the Parc coverage. Depending upon the size of the input files, these operations may take several minutes to complete. The UNIX commands are as follows:

System	Command
<UNIX.system.prompt> :	**jfb204h**
<UNIX.system.prompt> :	**arc**
arc :	**&run parc.aml**

The information in the Parc Coverage may be inspected, selected, reviewed, modified, amended, or deleted as desired with standard Arcinfo procedures. Arcview may also be employed to display the information in the coverage. In fact, Arcview is probably the most convenient and user friendly way to view the coverages.

4.2. EXPORT POTENTIAL ARC INFORMATION

When the user is satisfied with the contents of the Parc coverage, it is exported to an ASCII file by invoking the GIS macro script export.aml.

System	Command
<UNIX.system.prompt> :	**arc**
arc :	**& run export.aml**

This produces the intermediate ASCII file arcparc.pdb.

4.3. RUN EXPERT SYSTEM WITH DESIRED RULES SET

The expert system shell, Nexpert, is run from UNIX system level and an appropriate temporary knowledge base (*.tkb) is loaded interactively by the user.

System	Command
<UNIX.system.prompt> :	**nexpert**
<Under icon "Expert">	select **load knowledge base**
	select **r10p10.tkb**

Nexpert is an interactive environment and at this point the user may browse the existing objects and rules. Modifications to these may be effected by using the specialized editors and graphical utilities available within Nexpert. When the knowledge base is correct, the process of applying the rules to the parc objects is begun by interactively selecting the following Nexpert commands:

System	Command
<Under icon "Expert">	select **suggest volunteer**
	select **OK & Knowcess**

This will load the data from the file arcparc.pdb, apply the rules from the knowledge base, modify the objects as needed, export data to an intermediate ASCII file named nexparc.pdb, and invoke the FORTRAN programs jfb205b, jen1, and jfb237. Program jfb205b is a preprocessor that takes the data in file nexparc.pdb and reformats it appropriate to the requirements of the LP solver, jen1. The program jfb237 reformats output from jen1 into a form which can be imported into the GIS Parc coverage. Each of these steps is reported to the monitor upon completion. When the process is concluded, exit from Nexpert by selecting

System	Command
<Under icon "System">	select **Quit**

4.4. IMPORT MODIFIED POTENTIAL ARC INFORMATION

The next step involves the uploading the modified Parc data into the GIS. This is accomplished by entering Arcinfo and invoking the import script.

System	Command
<UNIX.system.prompt> :	**arc**
arc :	**&run import.aml**

4.5. DISPLAY RESULTS

Results of the automated allocation may be inspected visually by running the show10.aml and the plot10.aml scripts. The show10.aml script displays results to the screen at the workstation. The plot10.aml script produces a plotfile (pl1.ps) for hardcopy of the same input stream.

System	Command
<UNIX.system.prompt> :	**arc**
arc :	**&run show10.aml**
<UNIX.system.prompt> :	**arc**
arc :	**&run plot10.aml**

Sorted tables of allocations are available from FORTRAN program jfb206b.

System	Command
<UNIX.system.prompt> :	**jfb206b**

The show10.aml and plot10.aml scripts and the program jfb206b.for are interactive and prompt the user for the kinds of output that are desired.
The show10.aml and plot10.aml scripts provide the following prompt to the user:

< Enter C P Snnn Dnnn I R Nnnn >

The user must respond by typing:

C to clear the screen of arcs and display the basemap.

P to display all arcs with flow.

Snnn to display the arcs with flow that attach to Supply #nnn.

Dnnn to display the arcs with flow that attach to Demand #nnn.

I to display the arcs with increased (from previous) flow.

R to display the arcs with reduced (from previous) flow.

Nn to display the arcs declared not feasible by rule #nnn.

The program jfb206b.for begins by prompting for the type of output desired.
Respond by typing:

0 interactive retrieval.

1 a table sorted by Suppliers. (Results sent to file jfb206b.ou1).

2 a table sorted by Demanders (results sent to file jfb206b.ou2).

In the interactive mode, the program will prompt for the name of an individual supply source. The user must respond by typing the name of any one of the suppliers in the supsrc.dcs data file, or by <CR> or QUIT. A carriage return will cause the program to prompt for the name of an individual demander. The user must respond by typing the name of any one of the demanders in the file demand.dcs, or by <CR> or QUIT. A carriage return toggles back to the supply source prompt. Quit exits from the program to UNIX system level.

4.6. SUCCESSIVE APPLICATIONS

Sequential scenarios may be examined by returning to Step 4.2 above and re-executing the process. When the data are re-exported, previous values of flow on each arc are saved in the GIS Parc Coverage in the field PFLOWP for later comparison. Complete restart of the analysis may be achieved by executing the script reset.aml. When a reset is performed the previous flow is set to zero, the costs, flows, feasibilities, upper flow limits, and lower flow limits are reset to their initial values.

System	Command
<UNIX.system.prompt> :	**arc**
arc :	**&run reset.aml**

5.0 ALTERNATIVE INSPECTION OF RESULTS

The data stored in the GIS coverages are directly retrievable through the facilities of the Arcinfo Tables and Arcinfo Arcplot subcommands and also through Arcview. A knowledgeable user will find more flexibility and utility in these environments than can be found in the scripts and program described in Section 4.5. The user-friendly, interactive capabilities of Arcview make it particularly appropriate for selectively displaying the information contained in large coverages.

Table 1. Summary of AAS Procedures

1. Establish Arcinfo Coverages

```
arc < &run supsrc.aml
arc < &run demand.aml
<UNIX.system.prompt> jfb204f
arc < parc.aml
```

2. Export Potential Arc Information

```
arc < &run export.aml
```

3. Run Expert System With Desired Rules Set

```
<UNIX.system.prompt>  nexpert
        <Expert> Load  r10p10.tkb
        <Expert> Suggest volunteer
        <Expert> OK & Knowcess
                        (jfb205b)
                        (jen1)
                        (jfb237)
        <System> Quit
```

4. Import Modified Potential Arc Information

```
arc < &run import.aml
```

5. Display Results

```
arc < arcplot show10.aml
arc < arcplot plot10.aml
<UNIX.system.prompt>  jfb206b
```

Table 2 Summary Of Files And Processes

Input File		Program or Script		Output File
1. Establish Arcinfo Coverages				
supsrc.dcs	⇒	supsrc.aml	⇒	supsrc (coverage)
demand.dcs	⇒	demand.aml	⇒	demand (coverage)
supsrc.dcs demand.dcs	⇒	jfb204f	⇒	parc.d00 parc.d01
parc.d00 parc.d01	⇒	parc.aml	⇒	parc (coverage)
2. Export Potential Arc Information				
parc	⇒	export.aml	⇒	arcparc.pdb
3. Run Expert System With Desired Rules Set				
arcparc.pdb r10p10.tkb	⇒	nexpert	⇒	nexparc.pdb
supsrc.dcs demand.dcs nexparc.pdb	⇒	jfb205b	⇒	jen1.dat
jen1.dat	⇒	jen1	⇒	jen1.out
jen1.out	⇒	jfb237	⇒	jen1.pdb
4. Import Modified Potential Arc Information				
nexparc.pdb jen1.pdb	⇒	import.aml	⇒	parc
5. Display Results				
parc	⇒	show10.aml	⇒	(graphical output)
		plot10.aml	⇒	pl1.ps
		jfb206b	⇒	jfb206.ou1 jfb206.ou2

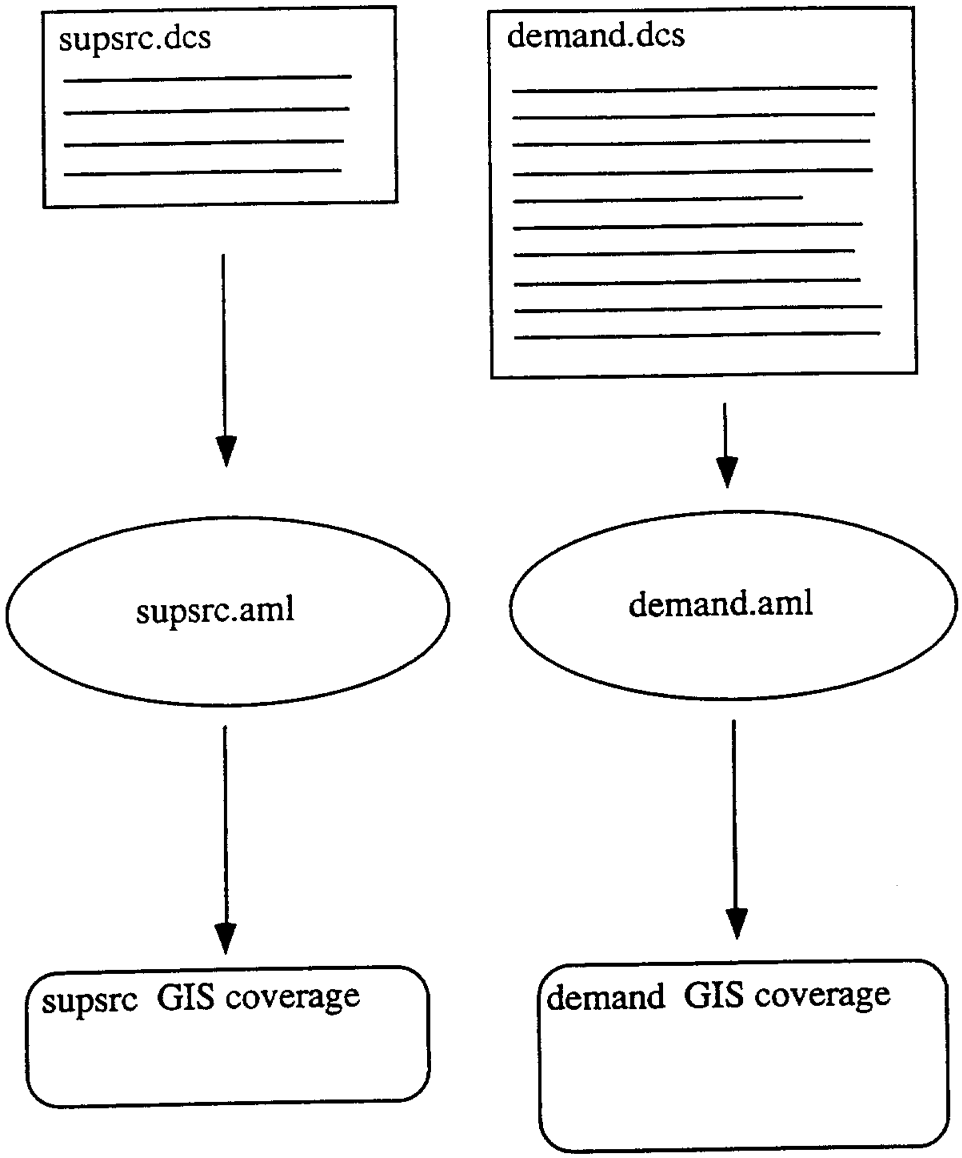


Figure 1. Creation of Supply and Demand Coverages

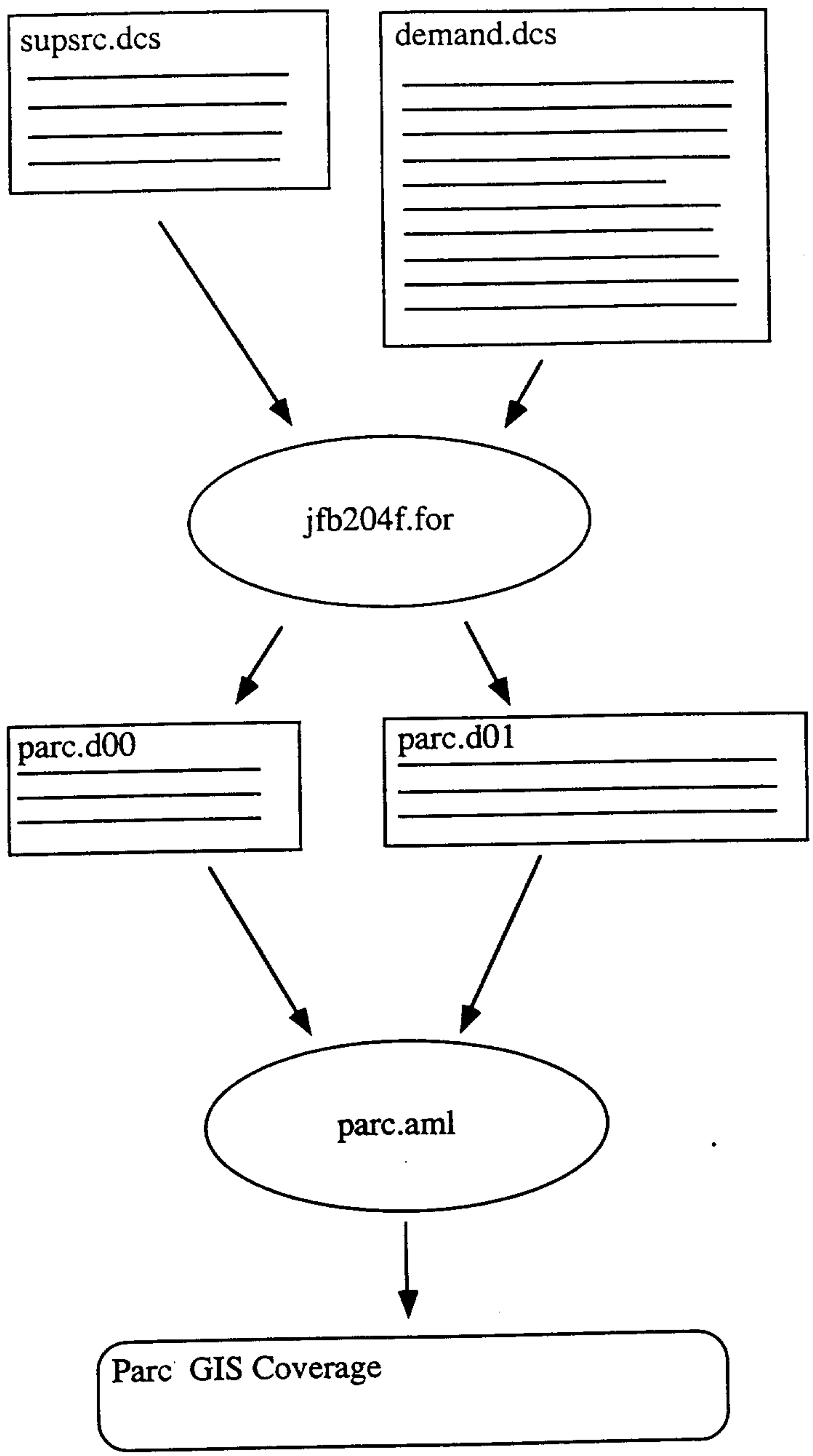


Figure 2. Creation of Parc Coverages

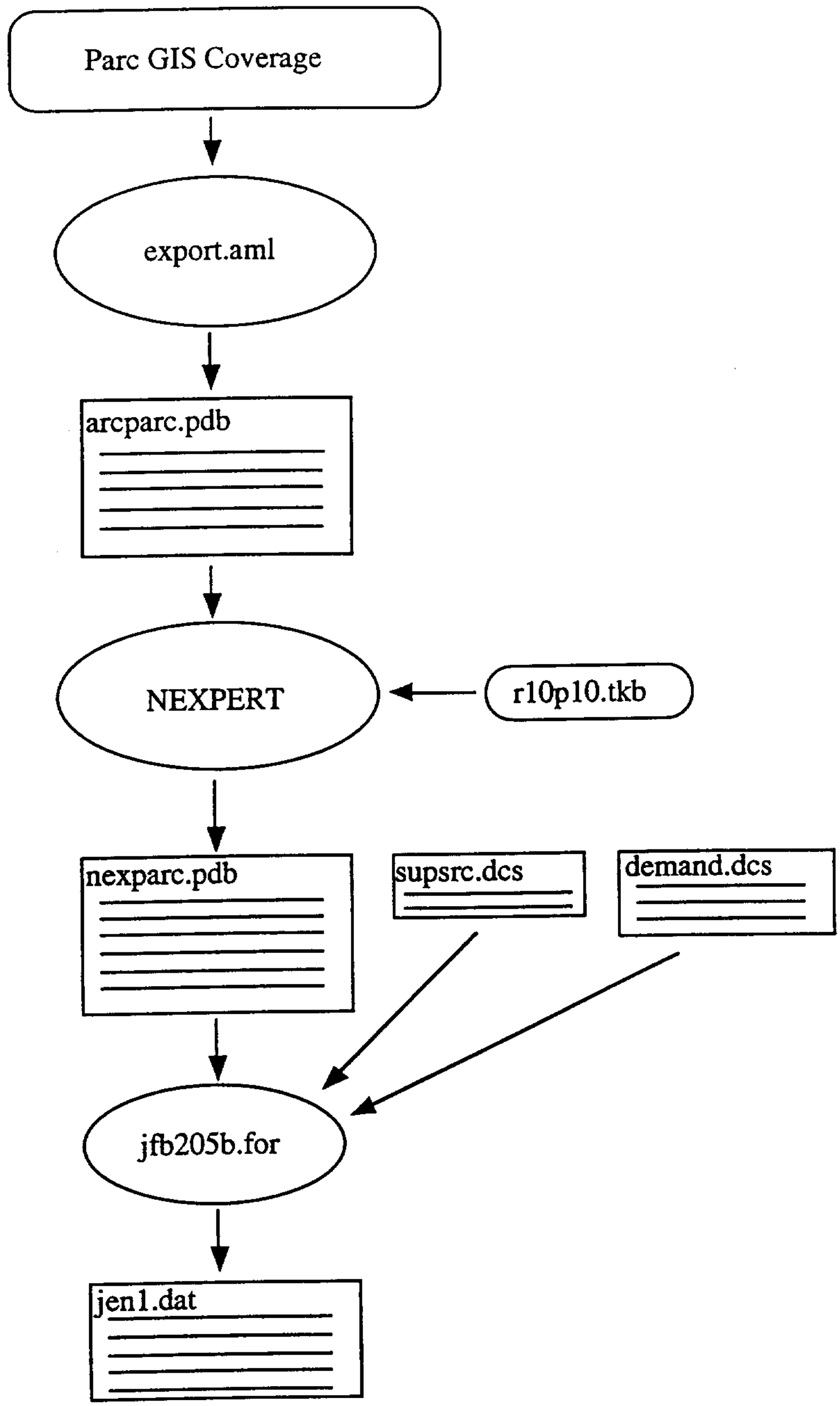


Figure 3. Expert System Execution

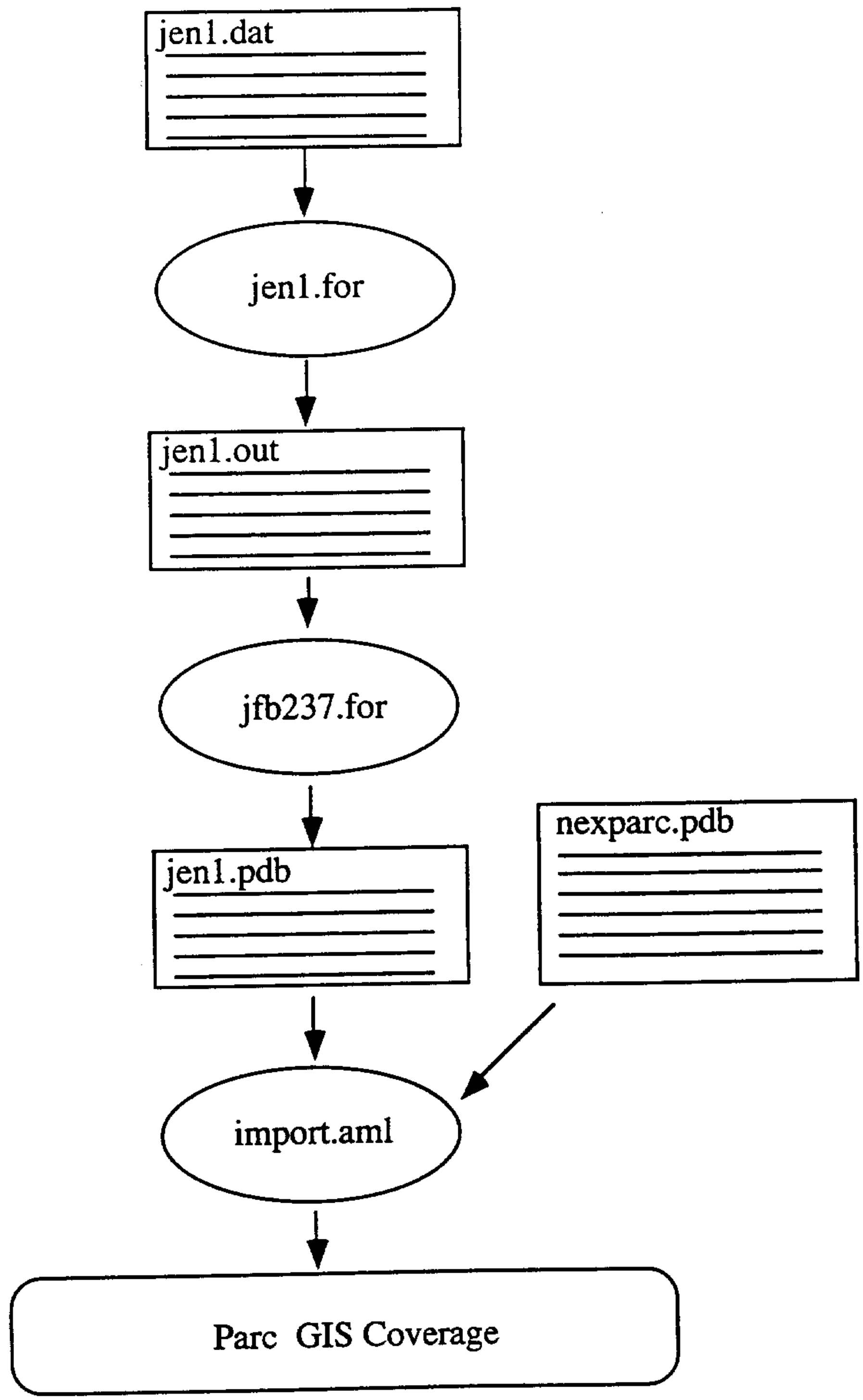


Figure 4. Network Flow Solver Execution

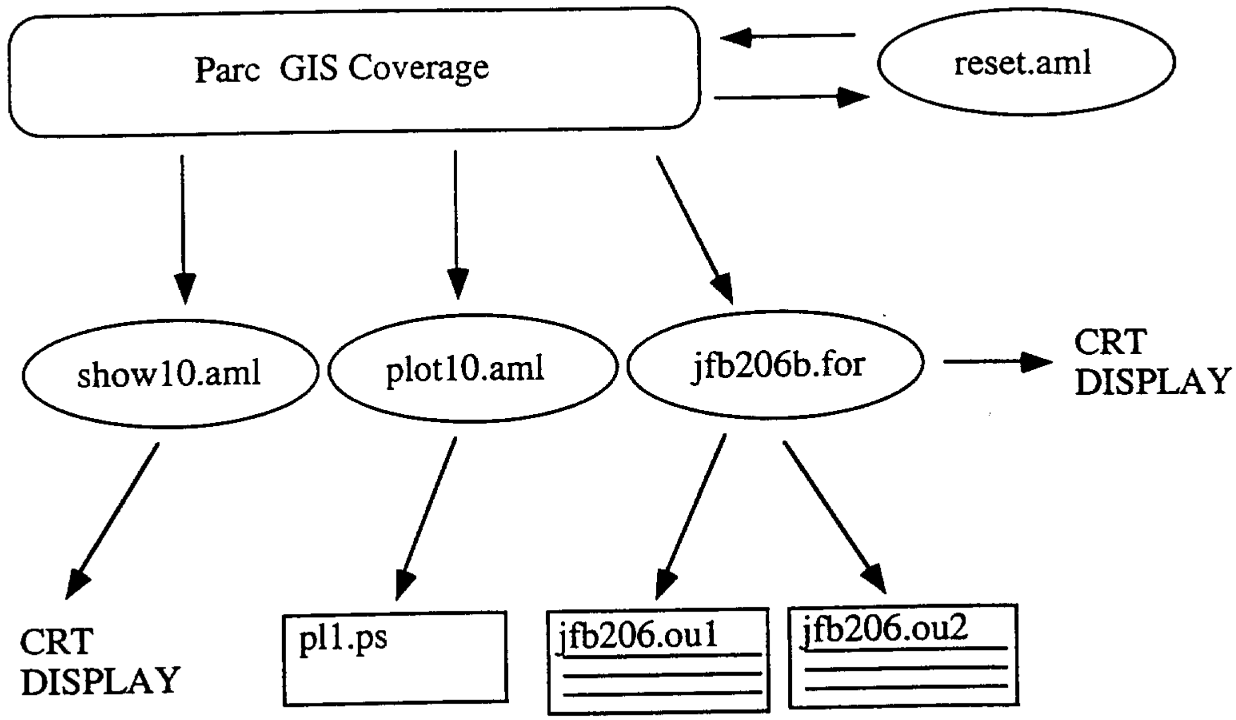


Figure 5. Results Display