LA-SUB-93-226

An Interactive Network Design System for MC&A Applications

by

Paul Helman

Department of Computer Science

University of New Mexico

Albuquerque, New Mexico 87131

June, 1988

RECEIVED

DEC 30 1988

OSTI

1. Overview

This report outlines enhancements to the design system that have been implemented in the Fall semester of 1987 and the Spring semester of 1988. The report includes a conceptual description of the new modes of operation and describes the refined cost model on which design evaluation is based, updating the report *A Design Methodology for MC&A Systems* [1]. The accompanying user's guide details operating procedures for the new software.

The system as currently implemented provides the following modes of operation.

Global System Design: Given a description of an application, find an optimal, or near optimal system graph (i.e., network configuration) and data allocation scheme. Return also detailed cost evaluations of this design.

System Graph Evaluation: Given a system graph and a set of operations, evaluate the system graph assuming an optimal, or near optimal data allocation scheme is used. (Return also the data allocation scheme.) This mode is similar to the *Global System Design Mode* except that the analyst specifies a proposed system graph for evaluation.

Data Allocation: This is the mode previously implemented and described in [1,2]. The mode now is invoked automatically as a subprocess of the two modes described above, and also is available as before as a stand alone option. As we shall describe, the allocation algorithms now permit allocations to far more general graph descriptions than was previously permitted, and support cost evaluation under any one of several reasonable network modeling assumptions.

The design software has been implemented to function conveniently as a decision support system. In addition to providing the analyst with the optimization algorithms referenced above, the system includes facilities that allow the analyst's expertise to be brought to bear on the network design problem. For example, once the optimization algorithms have produced a design, the analyst easily can modify this design and have the system evaluate the effect on cost of these modifications. We anticipate that a design often will be constructed by a process of successive refinements: the system constructs a candidate initial design; the analyst makes some modifications to the design based on his or her knowledge of the design process (invoking the system to analyze the impact of the modifications); the system again runs its optimization algorithms to improve the design that results from the analyst's modification, and so forth. To further enhance this mode of operation, the system includes facilities for saving and restoring in standard VMS text files all data (e.g., the operations associated with an application, an intermediate candidate design) associated with a design problem.

The *Global System Design* and *System Graph Evaluation* modes are detailed in Section 2. The enhancements to the data allocation mode, including the new cost evaluation concepts are presented in Section 4. Section 3 describes the modifications to operation templates that are necessary to support the refined cost modeling. Sections 2 and 3 also describes the new interactive facilities for specifying, modifying, and recalling from system maintained VMS files operation descriptions and candidate designs. The User's Guide gives further operational details.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

MA

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

2. Global System Design and System Graph Evaluation

2.1 Global System Design

The basic principles of global system design are unchanged from what is described in [1]. We now summarize the inputs and outputs for this mode of operation and describe a few differences in detail from what is described in [1].

Given: An application description, consisting of:

1. A collection T of operation templates (see Section 3 for details).
2. A collection of nodes in the plant. The analyst specifies, for each node:
 - a. A name (e.g., MBA1). These names are mnemonic devices that make interaction with the design system more meaningful.
 - b. A memory capacity. Currently, all non-zero capacities allow all data fragments to be allocated to a node. A zero capacity forbids any allocations to that node. An assignment of zero is used to model either a node which, for privacy or security reasons, cannot store any data, or a node which contains only a memoryless terminal.
 - c. A delay. This models the processing rate of the processor at the node. Currently, this processing rate is an estimate of the time required by the processor to process and forward a "typical" byte of data. An infinite delay should be assigned a node that contains only a terminal or other data entry equipment that can do no processing. Any node given an infinite delay is treated by the cost evaluation algorithms as an input station¹. This modeling capability is a major refinement of the previous work, allowing centralized and distributed systems to be more accurately compared. Section 4 discusses these new cost modeling capabilities.
3. A collection of 0 or more analyst-specified inter-node links. The specification of a link consists of:
 - a. The names of the two nodes that the link connects.
 - b. A delay, which models the bandwidth of the link. The delay specifies the bytes per time unit that can be transmitted across the link. The cost models now utilize this information in approximating an operation's response time. See Section 4 for details.
 - c. The cost in dollars of the link.

When the analyst requests that the system search for an optimal network configuration, the links that have been specified will always be included in the candidate configurations. In addition, the design algorithm considers the cost effectiveness of including in the design additional links as is described below.

4. A collection of forbidden fragment-to-station assignments. This is identical to what is described in [1,2].
5. A bound on the total cost in dollars on the links that can be used in the network configuration.

¹We use the term input station rather than MBA. This reflects the fact that an MBA can contain multiple input stations.

Find:

- A. An optimal, or near optimal, system design, subject to the cost bound. The design includes:
1. A linking strategy: The algorithm searches a solution space of connected graphs for the most effective linking strategy. All graphs in the solution space contain the links specified by the analyst as described above. In addition, the algorithms consider linking each node with an infinite delay to any single node with a noninfinite delay. This is done subject to the constraints of the cost bound. The most cost effective subset of links (subject to the cost bound) is added to the network.
 2. An allocation scheme: For each candidate graph in the solution space the system finds an optimal, or near optimal, allocation scheme as described in [1,2] and Section 4. This allocation scheme for the chosen candidate design is returned.
 3. Shortest path routing: For each optimal allocation scheme (for each candidate graph) the algorithm computes the shortest route (in terms of expected delay) through the network between each pair of nodes. These routes are used to calculate the operations' response times. The shortest path routing scheme for the chosen candidate graph and its optimal allocation scheme is available to the analyst for inspection.
- B. Various cost evaluations of the design constructed. In finding the optimal design, the system searches the solution space of candidate graphs and for each candidate graph, the system invokes algorithm Alloc to find the optimal allocation scheme for the graph. The system then performs cost evaluations on this overall design, using the results of the evaluation as the basis for selection. The system then summarizes various cost measures for the selected design. The cost measures now employed are substantially closer to an accurate Level II evaluation than in the previous implementation. Section 4 describes in detail the cost measures employed.

2.2 System Graph Evaluation

The primary differences between the System Graph Evaluation mode and the Global System Design mode described in the previous section is that in the evaluation mode the analyst completely specifies a single system graph for evaluation. The evaluation of the analyst's system graph proceeds exactly as described for each candidate in the Global System Design mode and the same information is returned as is returned for the selected graph in the Design mode.

To appreciate why the System Graph Evaluation mode is a useful complement to the Global System Design mode, we must consider again the solution space searched by the design algorithms. In addition to the links the analyst specifies, the design algorithms consider connecting each terminal (infinite delay node) to each one of the nonterminals (noninfinite delay node). These are the only additional links considered and these links all have the same delay and cost. Notice that while this does yield an enormous solution space (if there are n terminals and m nonterminals, approximately m^n configurations are considered), it is only a small fraction of the possible system graphs. We chose to search a limited solution space for feasibility reasons -- even with the restrictions described, computing time runs into minutes for moderate n and m .

On the other hand, the evaluation mode allows the analyst total flexibility in specifying the system graph to be evaluated. Any linking strategy, with each link having any delay desired, can be specified. It thus is useful to employ the two modes in tandem: the analyst first runs the design mode to get a first approximation of a design (coming from the restricted solution space). The analyst then uses his or her

expertise to modify the design manually (adding edges not considered by the design algorithm, and deleting edges the design algorithm has added). The analyst invokes the evaluation option to assess the effects of these modifications. After a series of such modifications/evaluations, the analyst can again invoke the design system to see if any additional edges (from the restricted solution space) are cost effective. In this sense the software functions as a decision support system.

2.3 Facilities for Specifying, Saving and Manipulating Graphs

The software has been implemented to aid the analyst in managing candidate designs. The user's guide details these capabilities; here we present only a high level overview.

The design system maintains a single *current system graph*. The analyst may invoke on the current system graph several different functions, including:

1. The Global System Design mode, using the current graph as starting point (see Section 2.1).
2. The System Graph Evaluation mode to evaluate the current graph (see Section 2.2).
3. A dialogue to specify manually an allocation scheme for the current graph (see Section 4).
4. A dialogue to modify manually the current graph.
5. A "Save" option to store in a system maintained VMS file the current graph.

The analyst makes a graph current in any one of several ways:

1. An interactive dialogue assists the analyst in building a graph from scratch.
2. An graph previously saved by the analyst (using the system's Save option) can be recalled.
3. The result of the Global System Design mode becomes the current graph.
4. An interactive dialogue assists the analyst in modifying the current instance (e.g., adding or deleting nodes and edges).

These facilities enable the software to function as a decision support system, combining the analyst's expertise with the system's automated design and evaluation algorithms.

3. Operation Template Definitions and the Operation Database

We have refined the definition of an operation template and implemented an interactive facility for specifying new operations and recalling from a database previously defined operations.

The new concepts pertaining to operation templates are:

The Operation Database: The system now maintains databases of user-defined operation classes. When an analyst is building the description of a particular application, he or she can specify the name of a user-defined operation class and the system will retrieve from an operation database the specified

operation and add it to the current application's description. In cases where an application requires an operation class not previously stored in an operation database, the analyst invokes an interactive dialogue facility to help define the operation class. The newly defined operation class then can be stored in a database for later reference. The system also allows all the operations of a particular application to be saved in an external file and recalled all at once (rather than one-at-a-time from an operation database).

Distinction between operation class and operation instance: We envision that the operations that define any MC&A application will be drawn from a relatively small collection of operation *classes*. (Recall that an example of an operation class is Transaction Entry.) However, for each specific application, the details of a class will vary (e.g., the frequency of an operation, the initiation node of an operation), thus leading to a large number of operation *instances*. This is handled by identifying as *parameters* several of the attributes that define an operation. When an operation class is stored in a database, values for only the non-parameterized attributes are stored. Thus, when a user recalls from a database an operation class for a particular application, he or she is prompted to supply values for the parameterized attributes. Specifically, the following attributes of an operation template are parameterized: Initiation Node, Frequency, number of required bytes for each fragment, and RSP and RLB values.

Interactive Traffic: A dominant cost in a typical LAN network (e.g., DEC/NET) is the cost of transmitting many small messages. In the context of our design problem, one source of this cost is the interactive communication between a data entry point which is a terminal and the processor which is handling the data entry (e.g., sending input menus and receiving the data). We now model this cost far better than we did previously, making comparisons between distributed and centralized systems far more meaningful. To support this improved cost modeling, an operation template now includes an attribute, *Interactive Communications*, which is an estimate of the amount (in bytes) of interactive communications that will be necessary to complete the operation.

Core Processes and Sub Processes: The analyst now has the option of partitioning an operation into *core processes* and *sub processes*. Conceptually, a core process is to be performed immediately after it is initiated, while a sub processes is "queued-up" and performed at the discretion of network scheduling algorithms. Operationally, when calculating how quickly an operation can be performed against a candidate design, the times to complete only the core processes are considered. In the future, if queuing models are employed, the sub processes will contribute to cost indirectly. *Important Note: Under the current implementation, processes designated as sub processes are recorded, but do not figure into the cost calculations.*

Reflecting these modifications, the current form of an operation template is as follows. A (*) indicates the attribute is parameterized for each specific instance.

Operation Class Name:

Frequency(*):

Initiation Node(*):

Core Processes:

{ This is a list of one or more units. Each unit specifies the following. }

Retrieve or update {all fragments in a unit are either retrieve or update}

Data Classes involved in unit

For each Data Class in unit:

Fragment within data class and number of bytes(*)

Interactive Communications:

{ The expected amount of interactive communications in bytes required to complete the operation. This value should include an estimate of the "protocol" bytes required to initiate the messages. }

Processing Intensity of Unit

{ An integer 1 - 5. This value currently has no affect on the cost calculations. }

Sub Processes:

{ This is a list of one or more units. Each unit specifies the following. }

Retrieve or update {all fragments in a unit are either retrieve or update}

Data Classes involved in unit

For each Data Class in unit:

Fragment within data class and number of bytes(*)

Processing Intensity of Unit

{ An integer 1 - 5. This value currently has no affect on the cost calculations. }

Evaluation Vector for Core Components(*):

{ The Core units collectively have one value for each quantity -- the evaluation considers the core complete when each unit completes. }

RSP { response time }

RLB { reliability }

4. Algorithm Alloc and the Refined Cost Model

In order to evaluate a candidate system graph, an allocation scheme of data fragments to the nodes of the network must be made. In addition, when estimating the cost of processing each operation against the candidate network (and its allocation scheme) a routing of the fragments through the network must be assumed. Finding optimal allocation and routing schemes are difficult tasks, but are important to the proper evaluation and eventual utilization of the candidate system graph.

In [1] we describe preliminary cost evaluation functions and the algorithm Alloc which finds the (true) optimal allocation scheme with respect to these functions. We also outline the far more complex Level II evaluation criteria which, among other things, better models data communication costs and queuing delays. The current implementation moves the design system considerably closer to the goals of a Level II-type evaluation, though we certainly are not completely there. It is conjectured, however, that for the type of applications of interest, and for the kinds of questions the design system will be called upon to answer, the current cost model is an appropriate compromise between efficiency of computation and accuracy of the results. It is anticipated that Gail Barlich's research in connection with her Master's project will test this conjecture by means of simulations and other empirical methods. Hopefully, this research also will identify

the aspects of the cost model that it would be most productive to improve.

4.1 Enhancements to the Cost Models: An Overview

Compared to the previous implementation of the design system, the enhancements to the cost models allow the current implementation to:

1. More accurately assess the benefits of increasing the number of processors in a system graph (e.g., allowing the system to more accurately compare a configuration specifying one or two central processors and only terminals at the MBA's to a configuration specifying processors at each MBA).
2. Compare different linking strategies for the nodes in the network (e.g., allowing the system to compare configurations that connect the nodes in a tree structure to a configuration with far more direct connections).
3. Predict system performance under different assumptions of network communication protocols and network load (e.g., allowing the system to predict the performance of a network under the assumption that the data necessary for an operation flows through the network in either a parallel or serial manner).

More specifically, the enhancements to the cost model that are currently implemented include:

Better Modeling of Terminals: We now allow a topology to specify that a node contains only a terminal node. This differs from a 0-memory node in that the terminal can do no processing. When an operation is initiated from such a node, a host processor node "closest" to the initiation node (see the description of shortest paths below) is selected to host the operation. The estimate of the cost of processing such an operation now includes the cost of data transmission for an interactive dialogue between the data entry person (or measurement device) at the initiation node and the host processor. This new cost factor allows the design system to far better evaluate the benefits of a distributed design.

Better Modeling of Network Distances: In the previous implementation, the cost models assumed that all nodes were directly connected. In the current implementation, arbitrary linking strategies, using links with arbitrary delays (bytes per time unit), are permitted. Consequently, it is now possible to analyze the cost effectiveness of different linking strategies. In order to support this analysis, the design system builds a shortest path matrix (based on the expected time required to transmit data) between each pair of nodes in the network. This matrix is then used by the evaluation and design algorithms in computing distances between pairs of nodes and in routing communications through the network.

Better Modeling of Network Characteristics: Cost evaluation is now performed with respect to more realistic and more flexible models of network communication characteristics. (See Section 4.2 for specific details of the cost functions used in the evaluations.) The cost to perform an operation's data updates is now computed in a manner that better reflects the characteristics of most networks. Previously [1,2], the time to update a data fragment was *summed* over all copies of the fragment. In the new implementation, this time is taken to be the *maximum* of the times to update the copies of the fragment (taking into account the distance of each copy from the node which initiates the update). This change reflects the fact that an update usually will be broadcast to, and processed at, the nodes containing fragment copies in *parallel*. It must be noted, however, that this change to the cost model makes the problem of finding the optimal allocation scheme NP-hard. Consequently, the new implementation of algorithm Alloc finds only an approximate solution with respect to this new cost model.

Two modes for evaluating retrieval times are now available. In addition to the previous evaluation of data transmission times based on the sum of the required bytes, a second type of evaluation is now available. This new evaluation models networks in which the data required for an operation can be transmitted in parallel across different links. The evaluation based on the parallel transmission model can be considered to be a "best case" analysis, reflecting the behavior of a lightly loaded network, while the evaluation based on the serial transmission model (which is the model that was used in the previous implementation) can be considered to be a "worst case" analysis, reflecting the behavior of a heavily loaded network.

It is important to note that algorithm Alloc finds an optimal allocation of fragments to nodes with respect to the serial transmission model for retrieval costs (but uses the new parallel model for update costs). If the serial and parallel retrieval evaluations for a given application are significantly different, and if the latter evaluation is deemed to be more appropriate than the former, the analyst can use the results of algorithm Alloc as a first approximation to a solution. As Section 4.3 describes, the system can be used to help the analyst consider refinements of the allocation scheme constructed by algorithm Alloc.

4.2 Details of the Cost Functions

This section supplies the cost functions used to compute the various evaluations returned by the system. See [1,2] for a more detailed explanation of some of the quantities referenced in this section.

Let N be the set of nodes of the network. Every node n has associated with it a delay value, $W(n) \geq 0$, and a 0/1 storage value, $S(n)$. $W(n)$ is a measure of the forwarding_time per storage_unit of node n (possibly infinite). If $S(n)$ is 0 then n has no storage capacity (eg. a terminal node), otherwise n has infinite storage capacity. Each edge, (n_1, n_2) , of the network has associated with it a transmission delay value, $W(n_1, n_2)$, in units of hop_time per storage_unit. Every path $(n_0, \dots, n_k)$ from n_0 to n_k has the delay value

$$\sum_{i=1}^{k-1} W(n_i) + \sum_{i=1}^k W(n_{i-1}, n_i).$$

Let $dist(n, N)$ be the minimal delay value over all paths from node n to some node in the node set N . If $n \in N$ then $dist(n, N) = 0$.

Let t be any transaction, f be any fragment, and A be an allocation scheme. Then

$freq(t)$ = a measure of the frequency of t ; either per unit time or relative to the other transactions.

$RSP(t)$ = a measure of the importance of a fast response to t .

$AMT_R(t, f)$ = the amount of f in storage_units retrieved by t .

$AMT_U(t, f)$ = the amount of f in storage_units updated by t .

$I(t)$ = the processing node of t (different from the t 's initiation node if the initiation node has infinite delay).

$RLB(t)$ = a measure of the importance of being able to perform t at any time.

$$D(f) = \sum_{t \text{ accesses } f} RLB(t).$$

$B(c, f) = \frac{2^c - 1}{2^c} \cdot D(f)$; a measure of the reliability value of allocating c copies of f .

$A(f)$ = the set of nodes containing a copy of f under scheme A .

$RLB(A) = \sum_f B(A(f), f)$; the reliability value of scheme A .

The new implementation of algorithm Alloc is invoked on the current candidate network by specifying Main Menu option 4. Algorithm Alloc attempts to minimize the cost function:

$$cost(A) = TRM(A) - RLB(A)$$

where we have the following $TRM(A)$ breakdown:

$$TRM(A) = \sum_t \sum_f TRM_R(t, f) + TRM_U(t, f),$$

$$TRM_R(t, f) = freq(t) \cdot RSP(t) \cdot dist(I(t), A(f)) \cdot AMT_R(t, f), \text{ and}$$

$$TRM_U(t, f) = freq(t) \cdot RSP(t) \cdot \max_{n \in A(f)} \{dist(I(t), \{n\})\} \cdot AMT_U(t, f)$$

In order to compute these quantities, a processing node is chosen for each initiation node with infinite delay, and the total amount of interactive dialogue between the processing and initiation node pairs is computed. It is assumed that all fragments are made forbidden to a node with infinite delay or such a node is given 0-memory (there is a single command that does this).

The value $TRM_R(t, f)$ is meant to reflect serial transmission times (required fragments are transmitted one-by-one to the processing node) or transmission times in a heavily loaded network. In either case, the value can serve as a "worst case" approximation of processing time. The values $TRM_R(t, f)$ and $TRM_U(t, f)$ are used to determine the following quantities as reported by Main Menu option 6:

1. Serial Response Time Values:

- Min response time over all operations :
- Max response time over all operations :
- Avg response time over all operations :

These values are computed with all $freq(t)$ and $RSP(t)$ values set to 1.

2. Weighted Serial Response Time Values:

- Min response time over all operations :
- Max response time over all operations :
- Avg response time over all operations :

These values are computed with all $freq(t)$ and $RSP(t)$ as specified in the definition of transaction t .

3. Serial response time. This unweighted value is reported for each operation individually. This value is exactly $\sum_f (dist(I(t), A(f)) \cdot AMT_R(t, f) + \max_{n \in A(f)} \{dist(I(t), \{n\})\} \cdot AMT_U(t, f))$.

The parallel response time for each transaction t is also computed. This computation uses the cost

function:

$$\max_f \{ \text{dist}(I(t), A(f)) \cdot \text{AMT}_R(t, f) \} + \sum_f \left(\max_{n \in A(f)} \{ \text{dist}(I(t), \{n\}) \} \cdot \text{AMT}_V(t, f) \right).$$

This value is meant to reflect parallel transmission times (required fragments are transmitted in parallel to the processing node). The calculation is optimistic and best models a lightly loaded network. This cost function is used to determine the following quantities as reported by Main Menu option 6:

1. Parallel Response Time Values

- Min response time over all operations :
- Max response time over all operations :
- Avg response time over all operations :

These values are computed with all $\text{freq}(t)$ and $\text{RSP}(t)$ values set to 1.

2. Weighted Parallel Response Time Values

- Min response time over all operations :
- Max response time over all operations :
- Avg response time over all operations :

These values are computed with all $\text{freq}(t)$ and $\text{RSP}(t)$ as specified in the definition of transaction t .

3. Parallel response time (lower bound). This unweighted value is reported for each operation individually. This value is exactly

$$\max_f \{ \text{dist}(I(t), A(f)) \cdot \text{AMT}_R(t, f) \} + \sum_f \left(\max_{n \in A(f)} \{ \text{dist}(I(t), \{n\}) \} \cdot \text{AMT}_V(t, f) \right).$$

Important Notes:

1. The old version of algorithm Alloc remains available as Main Menu option 3. This option is most appropriate when evaluating a completely connected network with processors at each node. Under this option, costs are computed somewhat differently than described above. Node delay values are 0 (so that initiation and processing node are always identical), node storage values are ignored (it is assumed that all fragments are made forbidden to a node with no storage), and all $\binom{V}{2}$ edges have delay value 1. The algorithm will find the allocation scheme minimizing the quantity $\text{cost}(A) = \text{TRM}(A) - \text{RLB}(A)$, where $\text{TRM}(A)$ and $\text{RLB}(A)$ are as above, except $\text{TRM}_V(t, f)$ is redefined to be $\text{freq}(t) \cdot \text{RSP}(t) \cdot |A(f) - \{I(t)\}| \cdot \text{AMT}_V(t, f)$.

$\text{cost}(A)$ is reported as

Under a complete topology ...

- Total weighted cost of allocation scheme :
- Storage units transmitted for retrieval per unit time :
- Storage units transmitted for update per unit time :

This value is supplied only for consistency with the previous implementation of algorithm Alloc and should be ignored in other contexts.

2. All cost evaluation options assume that the current system graph is *connected*. If the current system graph is not connected, most cost quantities will be meaningless. The design algorithms will detect a disconnected system graph and print an appropriate warning message.

4.3 Facilities for Specifying and Manipulating Allocation Schemes

The design system supports utilities that allow the analyst to manipulate allocation schemes for the current system graph. It is possible for the analyst to use this facility to create an allocation scheme from scratch, or to modify the allocation scheme constructed by algorithm Alloc.

Specifically, the design system allows the analyst to:

1. Make current an empty allocation scheme as the starting point for analyst specified allocations.
2. Make current the result of algorithm Alloc as the starting point for analyst specified modifications.
3. Allocate a fragment to a network node.
4. Deallocate a fragment from a network node
5. Assess the effects on cost of each allocation or deallocation. The change in cost is computed in terms of the cost function $cost(A) = TRM(A) - RLB(A)$ as defined in the previous section.

The user's guide contains details of these modes of operation.

References

1. Helman, P., A Design Methodology for MC&A Systems, internal report.
2. Helman, P., A fast, optimal data allocation algorithm with sensitivity analysis, Technical Report CS87-5, Department of Computer Science, University of New Mexico.

A User's Guide for the Interactive Network Design System

Paul Helman

Craig Morgenstern

Department of Computer Science

University of New Mexico

Albuquerque, New Mexico 87131

June, 1988

AUG 11 1988

1. Purpose of the Software

The software described in this document implements the network design and evaluation algorithms described in [1,2,3]. The software is intended as an interactive aid in the design of distributed MC&A systems. Specifically, the software is given as input a description of an MC&A application and produces as output a network topology, an allocation of data to the nodes of the network, and an evaluation of the overall design.

The optimization and analysis are performed with respect to cost functions that measure the quality of a design in terms of the expected times required for processing and data transmission, and the expected reliability provided by the design. The optimization respects a collection of user supplied cost and privacy constraints. See [1,2,3] for details of the cost functions minimized by the design algorithms.

Because the design problem is so difficult (even the simplest subproblems are NP-hard), the design algorithms produce only approximate solutions. A good design process requires the interaction of a database designer with the software. To this end, the software has been implemented to serve as a decision support system. In particular, the software allows the analyst to run a design algorithm to address any stage of the design problem, modify manually the results of the design algorithms, automatically evaluate the effects of these modifications, and then run the design algorithms on the next stage of the design problem. We believe this iterative approach to network design will be appropriate in many cases.

This User's Manual consists of two additional sections. Section 2 presents the design system's menus and explains the options. (Much of these explanations are available in the form of on-line help facilities.) Section 3 presents the results of a sample session with the design system.

2. System Menus and Options

MAIN MENU

1. Construct or modify a problem instance.

A problem instance consists of 3 sub-instances:

- a) A network instance (nodes, edges, delay and storage values),
- b) An operations instances (initiation nodes, frequencies, fragments updated and retrieved, etc...),
- c) A forbidden (fragment,node) pair instance.

These sub-instances must all exist and be consistent with each other before any of the other other MAIN MENU options are exercised. To ensure consistency, these rules of thumb should be followed:

- 1. Any change to any of the sub-instances invalidates any existing allocation scheme.
- 2. Any change to a node name invalidates the existing operations instance.
- 3. Any change to any node delay or storage value invalidates the existing forbidden (fragment,node) pair instance.
- 4. A change to the operations instance may invalidate the existing forbidden (fragment, node) pair instance if the change involves new classes/fragments not in the current instance. If the operations instance is invalidated, then the forbidden pair instance is likewise invalidated.

2. Manually specify an allocation scheme.

The analyst can specify that a fragment be assigned to a node, or that a current fragment assignment be removed.

3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.

This option ignores the current edge set of the current network, and assumes that a complete topology exists in which all edges have unit delay value. The analyst is given the option of disregarding any previous allocations. If this is not done, then ALLOC is run with respect to the existing allocations. Otherwise, ALLOC will produce an optimal allocation scheme based on the amount of data transmitted under the assumed complete topology. If no allocation scheme exists, this option will initialize to an empty scheme before invoking ALLOC.

4. Invoke ALLOC to construct an approximate allocation scheme for current topology.

ALLOC is run with respect to the current network topology. The allocation scheme produced is an approximation to the optimal scheme based on the time required to complete an operation; i.e., time to get retrievals and make updates. The analyst is given the option of disregarding any previous allocations made. If this is not done, then ALLOC

is run with respect to the existing allocations. If no allocation scheme exists, this option will initialize to an empty scheme before invoking ALLOC.

5. Best topology search for current instance.

The analyst specifies the nodes of the network using the network instance subsystem, as well as any edges that must be included. A search is made over all possible configurations of additional edges that don't exceed a specified ceiling cost for the topology that gives the best allocation scheme based on the time required to complete an operation. If there are two edge configurations that give the same allocation scheme, the configuration is chosen with minimal total cost. When the search terminates, both the best allocation scheme and the corresponding additional edge configuration are current.

6. Analyze the current allocation scheme.

Use this option to examine the current allocation scheme and various scheme statistics.

INSTANCE MENU

This menu is reached by specifying option 1 of the main menu.

1. Network instance subsystem -- construct, view or modify the current network instance.
2. Operations instance subsystem -- construct, view or modify the current operations instance.
3. Forbidden fragment instance subsystem -- construct, view or modify the current instance of forbidden (fragment,node) pairs.

Network Instance Subsystem (NIS)

VIEW NETWORK v [+]

The current network is displayed in the screen. An optional + flag gives the shortest path matrix.

PRINT NETWORK p [+]

The current network made readable in an external file. An optional + flag gives shortest path matrix. The analyst is queried for the file name and if the file already exists then the analyst is given the option of appending to the existing file. This command will not overwrite an existing file.

SAVE NETWORK s

The current network is copied into an external file. The analyst is queried for the file name and if the file exists, then the analyst is given the option of overwriting the file. The file formatted for later retrieval and is not intended to be readable.

GET NETWORK g

A network existing in an external file becomes the new current network. The analyst is queried for the file name, and it is assumed that the file will contain a network previously loaded by the SAVE NETWORK command.

CREATE NETWORK c

An empty network becomes the new current network.

ADD NODE n+ node_name storage [delay]

Adds a new node with the specified name and storage value to the network. Storage values are 0/1 for no storage capacity or infinite storage capacity. A delay value is optional -- the default is infinite delay.

DELETE NODE n- node_name

Removes the specified node from the network.

CHANGE NODE DELAY n\$ node_name [delay]

Changes the delay value of the specified node. If no delay value is given, then the node is assigned an infinite delay.

CHANGE NODE STORAGE .. n@ node_name [storage]

Changes the storage value of the specified node. If no storage value is given, then the node is assigned infinite storage.

CHANGE NODE NAME n& node_name new_node_name

Change the name of the specified node to the new specified name.

ADD EDGE e+ node_name node_name delay_value

Adds an edge to the network with the specified end-nodes and delay value.

DELETE EDGE e- node_name node_name

Removes the edge with the specified end-nodes from the network.

CHANGE EDGE DELAY e\$ node_name node_name delay_value

Changes the delay value of edge with the given end-nodes to specified value.

Operation Instance Subsystem (OIS)

CREATE INSTANCE C

Create (initialize) a new empty operations instance.

VIEW INSTANCE V [operation name]

The current operations instance is formatted on the screen. If an optional operation name is given, only that operation is shown.

LIST INSTANCE OPERATIONS .. L

List all operation names in the current instance on the screen.

PRINT INSTANCE P

The current operations instance is made readable in an external file. The analyst is queried for the file name and if the file already exists then the analyst is given the option of appending to the existing file. This command will not overwrite an existing file.

SAVE INSTANCE S

The current operations instance is saved in an external file for later retrieval.

The analyst is queried for the file name and if the file exists, then the analyst is given the option of overwriting the file. The file formatted for later retrieval and is not intended to be readable.

GET INSTANCE G

Initialize and load a new current operations instance from an external file. The analyst is queried for the file name, and it is assumed that the file will contain an operations instance previously loaded by the SAVE INSTANCE command.

CREATE DATABASE c

Create a new (empty) operations database. The analyst is queried for the database name and if the database exists, then the analyst is given the option of overwriting the database. The database is not intended to be readable.

VIEW OPERATION v

Specified operations in a specified database are formatted on the screen. The analyst is queried for the database name, and the desired operation names.

LIST OPERATIONS l

Lists all operation names contained in a specified database on the screen. The analyst is queried for the database name.

PRINT OPERATION p

Specified operations in a specified database are made readable in an external file. The analyst is queried for the file and database names, and if the file already exists then the analyst is given the option of appending to the existing file. This command will not overwrite an existing file.

SAVE OPERATION s

Specified operations of the current instance are saved in a specified database. The analyst is queried for the database name, and the desired operation names.

GET OPERATION g

Make specified operations from a specified database part of the current instance. The analyst is queried for the database name, the operation names and the parameterized operation attributes.

REMOVE OPERATION r

Remove specified operations from a specified database. The analyst is queried for the database name and the desired operation names.

NEW CURRENT OPERATION n

Create a new operation in the current instance. The analyst is queried for the operation attributes.

DELETE CURRENT OPERATION .. d

Remove an operation from the current instance. The analyst is queried for the operation name.

CREATE FORBIDDEN (fragment,node) PAIRS .. c

Initialize the forbidden pairs instance, and make all nodes with zero storage or infinite delay forbidden to all fragment assignments.

ADD FORBIDDEN (fragment,node) PAIR ... a

The analyst is prompted for (node,class,frag) triples, and the fragment of the specified class can not be allocated to the specified node. A wild card entry (*) in the (node,class,frag) triple refers to :

(n, *, ?) -- All triples with node = n.

(n, c, *) -- All triples with node = n and class = c.

DELETE FORBIDDEN (fragment,node) PAIR ... d

The analyst is prompted for (node,class,frag) triples, and the forbidden pair constraint is removed for the fragment of the specified class and the specified node. A wild card entry (*) in the (node,class,frag) triple refers to :

(*, ?, ?) -- All triples.

(n, *, ?) -- All triples with node = n.

(n, c, *) -- All triples with node = n and class = c.

VIEW FORBIDDEN (fragment,node) PAIRS v

Forbidden pairs are formatted to the screen.

PRINT FORBIDDEN (fragment,node) PAIRS ... p

Forbidden pairs are made readable in an external file. The analyst is prompted for the file name. Should the file already exist, then the analyst is given the option of appending to the file. This operation will not overwrite an existing file.

SAVE FORBIDDEN PAIR INSTANCE s

Forbidden pairs are saved in an external file for later retrieval. The analyst is queried for the file name, and if the file already exists, an overwrite option is presented. The file is not intended to be readable.

GET FORBIDDEN PAIR INSTANCE g

Forbidden pairs are loaded from an external file produced by the SAVE option. In addition, all nodes with zero storage or infinite delay are made to be forbidden to all fragment assignments.

MANUALLY SPECIFYING AN ALLOCATION SCHEME (ALLOC0)

This is option-2 of the main menu. If a current allocation scheme already exists, the analyst is given the option of removing all current allocations. If no current allocation scheme exists, then an empty allocation scheme is created.

ALLOCATE a <class> <frag> <node>

Allocate the specified fragment of the specified class to the specified node. If a forbidden constraint exists, no allocation will be performed.

DEALLOCATE d <class> <frag> <node>

Remove the specified fragment of the specified class from the specified node.

Topology Search Subsystem (TSS)

This is option 5 of the main menu.

VIEW EDGE COSTS v

The current edge cost matrix is formatted for the screen.

PRINT EDGE COSTS p

The edge cost matrix is made readable in an external file. The analyst is prompted for the file name, and if the file already exists, the analyst is queried for permission to append to the file. This option will not overwrite an existing file.

SAVE EDGE COSTS s

Save edge cost matrix in an external file for later retrieval. The analyst is prompted for the file name, and if the file exists, then the analyst is queried for overwrite permission. The file produced is not formatted to be readable.

GET EDGE COSTS g

Retrieve edge cost matrix from an external file. The analyst is queried for the file name. It is assumed that the file was produced by the SAVE option.

ADD/CHANGE EDGE COST .. a node1 node2 cost

Specify the cost of adding edge (node1,node2). Costs must be non-negative.

EDGE COST QUERY LOOP .. l

Analyst is queried for the edge costs of all possible edges, one at a time.

INVOKE SEARCH i

A search is made over all possible configurations of additional edges for the one that gives the best allocation scheme. The analyst is queried for ceiling value and an edge delay value. All edges added will have the same specified delay value. The sum of the costs of all edges included will not exceed the specified ceiling. The search is done with respect to any edges that exist in the network instance when this option is invoked. If two edge configurations give the same allocation scheme cost, then the edge configuration of minimal edge cost sum is selected. The allocation scheme cost is based on the time required to complete the operations on the current operations instance (as in option 4 of the main menu).

Fragment Allocation Scheme (FAS)

This is option 6 of the main menu.

VIEW SCHEME V

The current allocation scheme is formatted for the screen.

PRINT SCHEME P

The current allocation scheme is made readable in an external file. The analyst is queried for the file name and append permission if the file already exists. This option will not overwrite an existing file.

VIEW COST STATISTICS ... v

A detailed breakdown of the allocation scheme is formatted for the screen.

PRINT COST STATISTICS .. p

A detailed breakdown of the allocation scheme is made readable in an external file. The analyst is queried for the file name and append permission if the file already exists. This option will not overwrite an existing file.

3. Sample Session with the Design System

 * THIS IS A SAMPLE RUN USING THE EXISTING ".ex" EXAMPLE INSTANCE FILES. *

NETWORK DESIGN SYSTEM

MAIN MENU

select an option

0. Quit system.
1. Construct or modify a problem instance.
2. Manually specify an allocation scheme.
3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.
4. Invoke ALLOC to construct an approximate allocation scheme for current topology.
5. Best topology search for current instance.
6. Analyze the current allocation scheme.
7. Help.

selection --> 1

INSTANCE MENU

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

selection --> 1

Network Instance Subsystem (NIS)

QUIT SUBSYSTEM q
 VIEW NETWORK v
 PRINT NETWORK p
 SAVE NETWORK s
 GET NETWORK g
 CREATE NETWORK c

ADD NODE n+
 DELETE NODE n-
 CHANGE NODE DELAY n\$
 CHANGE NODE STORAGE .. n@
 CHANGE NODE NAME n&

ADD EDGE e+
 DELETE EDGE e-

CHANGE EDGE DELAY e\$

HELP ? or !

<NIS> --> g
 enter network instance file name (CR to exit) --> nw.ex
 New network loaded from file nw.ex.
 <NIS> --> v

CURRENT NETWORK INSTANCE

Total number of nodes in network : 10

Node MBA1 : delay = 1.000000 and storage = 0.000000
 Node MBA2 : infinite delay and storage = 0.000000
 Node MBA3 : delay = 1.000000 and storage = 0.000000
 Node MBA4 : infinite delay and storage = 0.000000
 Node MBA5 : delay = 1.000000 and storage = 0.000000
 Node MBA6 : infinite delay and storage = 0.000000
 Node INTER1 : delay = 0.500000 and storage = 1.000000
 Node INTER2 : delay = 0.500000 and storage = 1.000000
 Node INTER3 : delay = 0.500000 and storage = 1.000000
 Node CENTRAL : delay = 0.100000 and storage = 1.000000

Total number of edges in network : 3

Edge between INTER1 and CENTRAL has delay value of 0.250000.
 Edge between INTER2 and CENTRAL has delay value of 0.250000.
 Edge between INTER3 and CENTRAL has delay value of 0.250000.

Edge cost sum : 0.000000
 Network is not connected.

<NIS> --> q

INSTANCE MENU

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

selection --> 2

Operation Instance Subsystem (OIS)

QUIT SUBSYSTEM q
 HELP ? or !

CREATE INSTANCE C
 VIEW INSTANCE V
 LIST INSTANCE OPERATIONS .. L

PRINT INSTANCE P
 SAVE INSTANCE S
 GET INSTANCE G

CREATE DATABASE c
 VIEW OPERATION v
 LIST OPERATIONS l
 PRINT OPERATION p
 SAVE OPERATION s
 GET OPERATION g
 REMOVE OPERATION r

NEW CURRENT OPERATION n
 DELETE CURRENT OPERATION .. d

<OIS> --> G

enter instance file name (CR to exit) --> op.ex

Operations instance loaded from file op.ex.

<OIS> --> V

CURRENT OPERATIONS INSTANCE

OPERATION : op2

PROCESS : 1

process type - core
 initiation node name - INTER1
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f5 of class C is retrieved -- 10.000000 units.

OPERATION : op3

PROCESS : 1

process type - core
 initiation node name - MBA5
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f10 of class C is updated -- 100.000000 units.

OPERATION : op4

PROCESS : 1

process type - core
 initiation node name - MBA6
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f9 of class C is retrieved -- 1.000000 units.
 Fragment f10 of class C is retrieved -- 2.000000 units.
 Fragment f10 of class C is updated -- 100.000000 units.

OPERATION : op5

PROCESS : 1

process type - core
 initiation node name - MBA2
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f10 of class C is updated -- 100.000000 units.

OPERATION : op6

PROCESS : 1

process type - core
 initiation node name - MBA3
 frequency value - 10.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f10 of class C is updated -- 100.000000 units.

OPERATION : op7

PROCESS : 1

process type - core
 initiation node name - MBA1
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f10 of class C is retrieved -- 1000.000000 units.

OPERATION : op8

PROCESS : 1

process type - core
 initiation node name - MBA1
 frequency value - 20.000000
 interaction value - 10.000000
 processing intensity value - 1.000000
 response time factor - 50.000000
 reliability factor - 10.000000

Fragment f10 of class C is retrieved -- 4000.000000 units.

OPERATION : op9

PROCESS : 1

process type - core
 initiation node name - MBA2

frequency value = 20.000000
 interaction value = 10.000000
 processing intensity value = 1.000000
 response time factor = 50.000000
 reliability factor = 10.000000

Fragment f10 of class C is retrieved -- 5000.000000 units.

OPERATION : opl

PROCESS : 1

process type = core
 initiation node name = MBAl
 frequency value = 5.000000
 interaction value = 10.000000
 processing intensity value = 1.000000
 response time factor = 20.000000
 reliability factor = 5.000000

Fragment f3 of class C is retrieved -- 15.000000 units.

Fragment f5 of class C is retrieved -- 20.000000 units.

Fragment f9 of class C is updated -- 20.000000 units.

OPERATION : opl0

PROCESS : 1

process type = core
 initiation node name = MBA4
 frequency value = 25.000000
 interaction value = 10.000000
 processing intensity value = 1.000000
 response time factor = 20.000000
 reliability factor = 10.000000

Fragment f1 of class C is retrieved -- 150.000000 units.

Fragment f2 of class C is retrieved -- 25.000000 units.

Fragment f4 of class C is retrieved -- 17.000000 units.

Fragment f10 of class C is updated -- 100.000000 units.

OPERATION : opl1

PROCESS : 1

process type = core
 initiation node name = INTER2
 frequency value = 75.000000
 interaction value = 10.000000
 processing intensity value = 1.000000
 response time factor = 10.000000
 reliability factor = 10.000000

Fragment f1 of class C is retrieved -- 234.000000 units.

Fragment f2 of class C is retrieved -- 26.000000 units.

Fragment f3 of class C is retrieved -- 2000.000000 units.

Fragment f6 of class C is retrieved -- 10.000000 units.

Fragment f7 of class C is retrieved -- 189.000000 units.

Fragment f8 of class C is updated -- 1500.000000 units.

Fragment f4 of class C is updated -- 330.000000 units.

OPERATION : opl2

PROCESS : 1

process type = core
 initiation node name = INTER3
 frequency value = 25.000000
 interaction value = 10.000000
 processing intensity value = 1.000000
 response time factor = 20.000000
 reliability factor = 50.000000

Fragment f9 of class C is retrieved -- 500.000000 units.

Fragment f3 of class C is retrieved -- 1300.000000 units.

Fragment f1 of class C is retrieved -- 1200.000000 units.

Fragment f7 of class C is retrieved -- 250.000000 units.

Fragment f10 of class C is updated -- 4000.000000 units.

Fragment f5 of class C is updated -- 500.000000 units.

<OIS> -->

<OIS> --> q

INSTANCE MENU

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

selection --> 3

Forbidden Fragment Subsystem (FFS)

QUIT q
 CREATE FORBIDDEN (fragment,node) PAIRS .. c
 ADD FORBIDDEN (fragment,node) PAIR a
 DELETE FORBIDDEN (fragment,node) PAIR ... d
 VIEW FORBIDDEN (fragment,node) PAIRS v
 PRINT FORBIDDEN (fragment,node) PAIRS ... p
 GET FORBIDDEN PAIR INSTANCE g
 SAVE FORBIDDEN PAIR INSTANCE s
 HELP ? or !

<FFS> --> c

<FFS> --> v

FORBIDDEN (fragment,node) PAIRS

FORBIDDEN FRAGMENTS FOR NODE : MBAl

- fragment f5 of class C.
- fragment f10 of class C.
- fragment f9 of class C.
- fragment f3 of class C.
- fragment f1 of class C.
- fragment f2 of class C.

```
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : MBA2

```
-- fragment f5 of class C.
-- fragment f10 of class C.
-- fragment f9 of class C.
-- fragment f3 of class C.
-- fragment f1 of class C.
-- fragment f2 of class C.
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : MBA3

```
-- fragment f5 of class C.
-- fragment f10 of class C.
-- fragment f9 of class C.
-- fragment f3 of class C.
-- fragment f1 of class C.
-- fragment f2 of class C.
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : MBA4

```
-- fragment f5 of class C.
-- fragment f10 of class C.
-- fragment f9 of class C.
-- fragment f3 of class C.
-- fragment f1 of class C.
-- fragment f2 of class C.
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : MBA5

```
-- fragment f5 of class C.
-- fragment f10 of class C.
-- fragment f9 of class C.
-- fragment f3 of class C.
-- fragment f1 of class C.
-- fragment f2 of class C.
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : MBA6

```
-- fragment f5 of class C.
-- fragment f10 of class C.
```

```
-- fragment f9 of class C.
-- fragment f3 of class C.
-- fragment f1 of class C.
-- fragment f2 of class C.
-- fragment f4 of class C.
-- fragment f6 of class C.
-- fragment f7 of class C.
-- fragment f8 of class C.
```

FORBIDDEN FRAGMENTS FOR NODE : INTER1

FORBIDDEN FRAGMENTS FOR NODE : INTER2

FORBIDDEN FRAGMENTS FOR NODE : INTER3

FORBIDDEN FRAGMENTS FOR NODE : CENTRAL
<PFS> --> q

I N S T A N C E M E N U

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

selection --> 0

M A I N M E N U

select an option

0. Quit system.
1. Construct or modify a problem instance.
2. Manually specify an allocation scheme.
3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.
4. Invoke ALLOC to construct an approximate allocation scheme for current topology.
5. Best topology search for current instance.
6. Analyze the current allocation scheme.
7. Help.

selection --> 5

Topology Search Subsystem (TSS)

QUIT SUBSYSTEM q

VIEW EDGE COSTS v

PRINT EDGE COSTS p

```
SAVE EDGE COSTS ..... s
GET EDGE COSTS ..... g
ADD/CHANGE EDGE COST .. a
EDGE COST QUERY LOOP .. l
INVOKE SEARCH ..... i
HELP ..... ? or !
```

```
<TSS> --> g
enter edge cost matrix file name (CR to exit) --> costs.ex
Edge costs obtained from file costs.ex.
<TSS> --> v
```

Edge Cost Matrix

```
Cost of adding edge (MBA1, MBA2) : 1000000.000000
Cost of adding edge (MBA1, MBA3) : 1000000.000000
Cost of adding edge (MBA1, MBA4) : 1000000.000000
Cost of adding edge (MBA1, MBA5) : 1000000.000000
Cost of adding edge (MBA1, MBA6) : 1000000.000000
Cost of adding edge (MBA1, INTER1) : 1.000000
Cost of adding edge (MBA1, INTER2) : 1.000000
Cost of adding edge (MBA1, INTER3) : 1.000000
Cost of adding edge (MBA1, CENTRAL) : 100000.000000
Cost of adding edge (MBA2, MBA3) : 100000.000000
Cost of adding edge (MBA2, MBA4) : 100000.000000
Cost of adding edge (MBA2, MBA5) : 100000.000000
Cost of adding edge (MBA2, MBA6) : 100000.000000
Cost of adding edge (MBA2, INTER1) : 1.000000
Cost of adding edge (MBA2, INTER2) : 1.000000
Cost of adding edge (MBA2, INTER3) : 1.000000
Cost of adding edge (MBA2, CENTRAL) : 100000.000000
Cost of adding edge (MBA3, MBA4) : 100000.000000
Cost of adding edge (MBA3, MBA5) : 100000.000000
Cost of adding edge (MBA3, MBA6) : 100000.000000
Cost of adding edge (MBA3, INTER1) : 1.000000
Cost of adding edge (MBA3, INTER2) : 1.000000
Cost of adding edge (MBA3, INTER3) : 1.000000
Cost of adding edge (MBA3, CENTRAL) : 100000.000000
Cost of adding edge (MBA4, MBA5) : 100000.000000
Cost of adding edge (MBA4, MBA6) : 100000.000000
Cost of adding edge (MBA4, INTER1) : 1.000000
Cost of adding edge (MBA4, INTER2) : 1.000000
Cost of adding edge (MBA4, INTER3) : 1.000000
Cost of adding edge (MBA4, CENTRAL) : 100000.000000
Cost of adding edge (MBA5, MBA6) : 100000.000000
Cost of adding edge (MBA5, INTER1) : 1.000000
Cost of adding edge (MBA5, INTER2) : 1.000000
Cost of adding edge (MBA5, INTER3) : 1.000000
Cost of adding edge (MBA5, CENTRAL) : 100000.000000
Cost of adding edge (MBA6, INTER1) : 1.000000
Cost of adding edge (MBA6, INTER2) : 1.000000
Cost of adding edge (MBA6, INTER3) : 1.000000
Cost of adding edge (MBA6, CENTRAL) : 100000.000000
Cost of adding edge (INTER1, INTER2) : 10.000000
Cost of adding edge (INTER1, INTER3) : 10.000000
Cost of adding edge (INTER1, CENTRAL) : 1.000000
```

```
Cost of adding edge (INTER2, INTER3) : 10.000000
Cost of adding edge (INTER2, CENTRAL) : 1.000000
Cost of adding edge (INTER3, CENTRAL) : 1.000000
```

```
<TSS> --> i
```

SEARCHING FOR BEST TOPOLOGY FOR CURRENT INSTANCE

```
Enter total edge cost ceiling --> 25
Enter new edge delay value --> 1.0
Topology search begins ...
```

New best topology has edge cost = 15.000000 and ALLOC cost = 6366214.687500

```
*
*
*
```

THIS SEARCH TOOK ALMOST 1 HOUR ON A MICRO_VAX_II.

```
<TSS> --> q
```

MAIN MENU

select an option

0. Quit system.
1. Construct or modify a problem instance.
2. Manually specify an allocation scheme.
3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.
4. Invoke ALLOC to construct an approximate allocation scheme for current topology.
5. Best topology search for current instance.
6. Analyze the current allocation scheme.
7. Help.

```
selection --> 1
```

INSTANCE MENU

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

```
selection --> 1
```

Network Instance Subsystem (NIS)

```

QUIT SUBSYSTEM ..... q

VIEW NETWORK ..... v
PRINT NETWORK ..... p
SAVE NETWORK ..... s
GET NETWORK ..... g
CREATE NETWORK ..... c

ADD NODE ..... nt
DELETE NODE ..... n-
CHANGE NODE DELAY .... n$
CHANGE NODE STORAGE .. n@
CHANGE NODE NAME ..... n&

ADD EDGE ..... et
DELETE EDGE ..... e-
CHANGE EDGE DELAY .... e$

HELP ..... ? or !

```

<NIS> --> v

CURRENT NETWORK INSTANCE

Total number of nodes in network : 10

```

Node MBA1 : delay = 1.000000 and storage = 0.000000
Node MBA2 : infinite delay and storage = 0.000000
Node MBA3 : delay = 1.000000 and storage = 0.000000
Node MBA4 : infinite delay and storage = 0.000000
Node MBA5 : delay = 1.000000 and storage = 0.000000
Node MBA6 : infinite delay and storage = 0.000000
Node INTER1 : delay = 0.500000 and storage = 1.000000
Node INTER2 : delay = 0.500000 and storage = 1.000000
Node INTER3 : delay = 0.500000 and storage = 1.000000
Node CENTRAL : delay = 0.100000 and storage = 1.000000
Edge between MBA1 and INTER3 has delay value of 1.000000.
Edge between MBA2 and INTER3 has delay value of 1.000000.
Edge between MBA3 and INTER3 has delay value of 1.000000.
Edge between MBA4 and INTER3 has delay value of 1.000000.
Edge between MBA5 and INTER3 has delay value of 1.000000.
Edge between MBA6 and INTER3 has delay value of 1.000000.
Edge between INTER1 and CENTRAL has delay value of 1.000000.
Edge between INTER2 and CENTRAL has delay value of 1.000000.
Edge between INTER3 and CENTRAL has delay value of 1.000000.

```

Edge cost sum : 9.000000
Network is connected.

<NIS> --> v

I N S T A N C E M E N U

select an option

0. Return to MAIN menu.
1. Network instance subsystem.
2. Operations instance subsystem.
3. Forbidden fragment instance subsystem.
4. Help

selection --> 0

M A I N M E N U

select an option

0. Quit system.
1. Construct or modify a problem instance.
2. Manually specify an allocation scheme.
3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.
4. Invoke ALLOC to construct an approximate allocation scheme for current topology.
5. Best topology search for current instance.
6. Analyze the current allocation scheme.
7. Help.

selection --> 6

Fragment Allocation Scheme (FAS)

```

QUIT ..... q
VIEW SCHEME ..... v
PRINT SCHEME ..... p
VIEW COST STATISTICS ... v
PRINT COST STATISTICS .. p
MORE HELP ..... !

```

<FAS> --> v

CURRENT ALLOCATION SCHEME

NODE : MBA1

NODE : MBA2

NODE : MBA3

NODE : MBA4

NODE : MBA5

NODE : MBA6

NODE : INTER1

- fragment f3 of class C
- fragment f1 of class C
- fragment f2 of class C
- fragment f6 of class C
- fragment f7 of class C

NODE : INTER2

- fragment f3 of class C
- fragment f1 of class C
- fragment f2 of class C
- fragment f4 of class C
- fragment f6 of class C
- fragment f7 of class C
- fragment f8 of class C

NODE : INTER3

- fragment f5 of class C
- fragment f10 of class C
- fragment f9 of class C
- fragment f3 of class C
- fragment f1 of class C
- fragment f2 of class C
- fragment f6 of class C
- fragment f7 of class C

NODE : CENTRAL

- fragment f3 of class C
- fragment f1 of class C
- fragment f2 of class C
- fragment f6 of class C
- fragment f7 of class C

<FAS> --> v

ALLOCATION SCHEME ANALYSIS

Fragment reliability factors

- * fragment f5 of class C : 65.000000
- * fragment f10 of class C : 140.000000
- * fragment f9 of class C : 65.000000
- * fragment f3 of class C : 65.000000
- * fragment f1 of class C : 70.000000
- * fragment f2 of class C : 20.000000
- * fragment f4 of class C : 20.000000
- * fragment f6 of class C : 10.000000
- * fragment f7 of class C : 60.000000
- * fragment f8 of class C : 10.000000

Scheme reliability measure : 360.937500

Under a complete topology ...

- Total weighted cost of allocation scheme : 5173639.062500
- Storage units transmitted for retrieval per unit time : 100800.000000
- Storage units transmitted for update per unit time : 3100.000000

Under the current topology ...

- Storage units transmitted for interactive dialog per unit time : 850.000000
- Do you wish to see a cost break-down by operation? [y,n] : y

OPERATION : op2

* Initiation node : INTER1

* Processing node : INTER1

- * Storage units transmitted for interaction : 0.000000
- * Storage units transmitted to processing node for retrieval : 10.000000
- * Storage units transmitted from processing node for updates : 0.000000
- * Serial response time : 21.000000
- * Parallel response time (lower bound) : 21.000000

OPERATION : op3

* Initiation node : MBA5

* Processing node : MBA5

- * Storage units transmitted for interaction : 0.000000
- * Storage units transmitted to processing node for retrieval : 0.000000
- * Storage units transmitted from processing node for updates : 100.000000
- * Serial response time : 100.000000
- * Parallel response time (lower bound) : 100.000000

OPERATION : op4

* Initiation node : MBA6

* Processing node : INTER3

- * Storage units transmitted for interaction : 10.000000
- * Storage units transmitted to processing node for retrieval : 0.000000
- * Storage units transmitted from processing node for updates : 0.000000
- * Serial response time : 0.000000
- * Parallel response time (lower bound) : 0.000000

OPERATION : op5

* Initiation node : MBA2

* Processing node : INTER3

- * Storage units transmitted for interaction : 10.000000
- * Storage units transmitted to processing node for retrieval : 0.000000
- * Storage units transmitted from processing node for updates : 0.000000
- * Serial response time : 0.000000
- * Parallel response time (lower bound) : 0.000000

OPERATION : op6

* Initiation node : MBA3

* Processing node : MBA3

- * Storage units transmitted for interaction : 0.000000
- * Storage units transmitted to processing node for retrieval : 0.000000
- * Storage units transmitted from processing node for updates : 100.000000
- * Serial response time : 100.000000
- * Parallel response time (lower bound) : 100.000000

OPERATION : op7

* Initiation node : MBAl

* Processing node : MBAl

- * Storage units transmitted for interaction : 0.000000
- * Storage units transmitted to processing node for retrieval : 1000.000000
- * Storage units transmitted from processing node for updates : 0.000000
- * Serial response time : 1000.000000
- * Parallel response time (lower bound) : 1000.000000

OPERATION : op8

* Initiation node : MBAl

* Processing node : MBAl

```
* Storage units transmitted for interaction : 0.000000
* Storage units transmitted to processing node for retrieval : 4000.000000
* Storage units transmitted from processing node for updates : 0.000000
* Serial response time : 4000.000000
* Parallel response time (lower bound) : 4000.000000
```

```
OPERATION : op9
* Initiation node : MBA2
* Processing node : INTER3
* Storage units transmitted for interaction : 10.000000
* Storage units transmitted to processing node for retrieval : 0.000000
* Storage units transmitted from processing node for updates : 0.000000
* Serial response time : 0.000000
* Parallel response time (lower bound) : 0.000000
```

```
OPERATION : opl
* Initiation node : MBAl
* Processing node : MBAl
* Storage units transmitted for interaction : 0.000000
* Storage units transmitted to processing node for retrieval : 35.000000
* Storage units transmitted from processing node for updates : 20.000000
* Serial response time : 55.000000
* Parallel response time (lower bound) : 40.000000
```

```
OPERATION : opl0
* Initiation node : MBA4
* Processing node : INTER3
* Storage units transmitted for interaction : 10.000000
* Storage units transmitted to processing node for retrieval : 17.000000
* Storage units transmitted from processing node for updates : 0.000000
* Serial response time : 35.700000
* Parallel response time (lower bound) : 35.700000
```

```
OPERATION : opl1
* Initiation node : INTER2
* Processing node : INTER2
* Storage units transmitted for interaction : 0.000000
* Storage units transmitted to processing node for retrieval : 0.000000
* Storage units transmitted from processing node for updates : 0.000000
* Serial response time : 0.000000
* Parallel response time (lower bound) : 0.000000
```

```
OPERATION : opl2
* Initiation node : INTER3
* Processing node : INTER3
* Storage units transmitted for interaction : 0.000000
* Storage units transmitted to processing node for retrieval : 0.000000
* Storage units transmitted from processing node for updates : 0.000000
* Serial response time : 0.000000
* Parallel response time (lower bound) : 0.000000
```

```
Serial Response Time Values
Min response time over all operations : 0.000000
Max response time over all operations : 4000.000000
Avg response time over all operations : 442.641667
```

```
Weighted Serial Response Time Values
Min response time over all operations : 0.000000
Max response time over all operations : 200000.000000
Avg response time over all operations : 21905.333333
```

```
Parallel Response Time Values
Min response time over all operations : 0.000000
Max response time over all operations : 4000.000000
Avg response time over all operations : 441.391667
```

```
Weighted Parallel Response Time Values
Min response time over all operations : 0.000000
Max response time over all operations : 200000.000000
Avg response time over all operations : 21880.333333
```

<FAS> --> q

MAIN MENU

select an option

0. Quit system.
1. Construct or modify a problem instance.
2. Manually specify an allocation scheme.
3. Invoke ALLOC to construct an optimal allocation scheme for a complete topology.
4. Invoke ALLOC to construct an approximate allocation scheme for current topology.
5. Best topology search for current instance.
6. Analyze the current allocation scheme.
7. Help.

selection --> 0

References

1. Helman, P., A Design Methodology for MC&A Systems, internal report.
2. Helman, P., A fast, optimal data allocation algorithm with sensitivity analysis, Technical Report CS87-5, Department of Computer Science, University of New Mexico.
3. Helman, P., An interactive network design system for MC&A applications, internal report.