RESQ—A package for solution of generalized queueing networks

by C. H. Sauer, M. Reiser and E. A. MacNAir

IBM Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT

RESQ (RESearch Queueing) is a tool for solution of queueing networks. The class of networks treated includes general multi-server queues, passive queues and complex routing decisions. Multiple solution techniques are provided, including numerical solution of separable balance equations and regenerative simulation. User access is provided through both interactive dialogue and a subroutine level interface.

INTRODUCTION

RESQ allows explicit consideration of many system features which are often ignored in queueing models. The goal of RESQ is to provide facilities for convenient model construction and efficient model solution so that the user can concentrate on formulating models. The user, e.g., a system designer or developer, need not be sophisticated with respect to the methods of solution. Since several solution methods are provided, the user can use the method most appropriate to the model and can use two or more methods in a hybrid solution. The constructs of RESQ are oriented toward computer and communication system features, but the terminology is strictly in terms of queueing networks.

RESQ employs state of the art techniques for solution of queueing networks. Depending on the particular model, the solution may be obtained by numerical analysis of separable balance equations or simulation. The separable balance equation solutions are available for a subset of RESQ models. The simulation techniques include the regenerative method for determination of confidence intervals and a sequential sampling method for determination of appropriate run lengths. By providing a high level framework for model definition and appropriate analysis of simulation results, RESQ alleviates two of the common problems with simulation: expense of constructing simulation models and insufficient statistical analysis of simulation results. Other techniques and solution methods are being included in RESQ on an experimental basis.

All RESQ capabilities are provided through a set of PL/I programs. In addition, the user interface components of RESQ are duplicated in APL. Queueing networks can be defined, listed, evaluated and revised either interactively or by writing programs which call RESQ routines.

This paper is organized as follows: The second section briefly summarizes related previous work on application of queueing network models, solution techniques for queueing networks and solution packages. The third section describes the generalized class of networks provided for in RESQ and the dialogues for network definition. The last section considers listing, evaluation and revision of networks. Additional details are given in Reference 4.

QUEUEING NETWORKS

In analyzing the performance of computing and communication systems, one usually finds the dominant factor to be contention for resources such as processors, memory, secondary storage, communication links, etc. Therefore queueing network models can be used to characterize this contention and estimate system performance. Some of the earlier efforts in this area were those of Kleinrock, Smith and Buzen. More recently there has been much work in this area, see References 8, 9, and 10 for examples and further references.

Corresponding to the activity in application of queueing network models, there has been much progress in the solution of queueing networks. Very complex queueing networks can be represented as Markovian processes and most of the solution efforts have done so. The direct numerical solution of these processes can be attempted for modest size problems but this approach is not practical in general. Representing the solution of the process as that of a collection of separable balance equations has made possible the solution of very large problems with restricting assumptions. There is hope that approximate solutions will alleviate the need for such assumptions. Finally, there has been much recent effort to improve statistical analysis of simulation, in particular the simulation of regenerative systems such as Markov processes.

There have been a variety of packages proposed and
implemented for the solution of queueing networks. However, all of these provide only a single solution technique and thus are only useful when the solution technique is appropriate to the problem. One of the major advantages of RESQ is that it provides several solution techniques. Thus the user can solve similar models and study the effects of different restricting assumptions (it is rare that the modeler can explicitly consider all system characteristics) and can construct hybrid solutions using more than one solution technique.

RESQ CONSTRUCTS AND INTERACTIVE DIALOGUES

This section briefly describes the constructs of RESQ and illustrates some of them with a model of a simple terminal oriented computing system. This model is an extension of the central server model proposed by Buzen, and is similar to models discussed in References 8 and 9.

The elements of RESQ include:

1. A population of jobs. Each job has an attached variable which can be used to retain job attributes.
2. A set of queues. There are two types of queues, active and passive. Active queues are queues in the traditional sense. Passive queues are used to represent contention for secondary resources and regulate sub-network job populations.
3. A set of nodes. Some types of nodes are parts of queues. Other nodes are used for auxiliary functions such as creation of jobs or changing the value of a job's variable.
4. A set of routing rules. These rules allow probabilistic and deterministic routing of jobs from among the nodes of the network.

In the example, jobs represent users of the system. A job alternates between think times at a terminal and use of the computational facilities. When a job is to perform computations, it first acquires memory and then alternates between use of a central processor and use of the input/output devices. The job variable is used to count the number of cycles of alternating computation and input/output. A passive queue is used to represent memory; active queues are used to represent other components. A model such as this can be used to estimate response times, device utilizations, queue lengths and other performance metrics.

In interactive usage of RESQ the command SETUP is used to define a network. As with other RESQ commands, SETUP enters the user into a dialogue where RESQ will prompt the user for information, e.g., the name of the model, queue characteristics, etc. If the user's response seems correct to RESQ, more information will be requested by RESQ until the command is finished. If RESQ discovers an error in the user's response then it will produce an error message and repeat the prompt. If the user wishes clarification of the prompt, then the user replies "how," RESQ responds with a detailed description of the information needed, and then RESQ repeats the request. If there is a default value for a particular prompt, then the user may effect the default value by entering a return of a null line. In this section, user responses will always be given in lower case.

Active queues

An active queue consists of a set of servers, a set of waiting areas for jobs requesting or receiving service and a control mechanism for allocating the servers to the jobs.

Single server queues

The waiting areas of the queue are called classes. These classes are local to the queue and are to be distinguished from the global "classes" often used in queueing literature. As described in a later section, a job is routed to one of these classes and joins the queue. Upon arrival the work demanded by the job is determined as follows: First, a sample is taken from the work demand distribution associated with the job's class. Second, the flag for job variable scaling is tested. If there is such a flag for each class, Job variables will be discussed in a later section. If the flag is set, then the sample taken in the first step is multiplied by the job variable.

Once placed in the waiting area, the job remains there until all of the work demanded is completed. When all of the job's work is complete, the job instantaneously departs from the queue. The server chooses which job to serve according to the overhead mechanism, if any, and the queueing discipline.

Multi-server queues

Queues with more than one server allow all of the capabilities described above except for the cyclic priority queueing discipline. Each server has associated with it an effective rate and a set of classes which it will serve. The rate of a server may be a function of the total number of jobs at the queue. If all servers at a queue have exactly the same characteristics, the queue is considered symmetric. Otherwise it is asymmetric.

Dialogue for active queues

The first prompt is for the queue type. The subsequent prompts are strongly dependent on the queue type. In
setup
MODEL NAME: ecsm
METHOD: aplomb
NUMBER OF:
  CHAINS: 1
  QUEUES: 5
  CLASSES: 4
  ALLOCATE NODES: 1
  RELEASE NODES: 1
  DESTROY NODES: 0
  CREATE NODES: 0
  SET NODES: 2
  FISSION NODES: 0
  FUSION NODES: 0
  SPLIT NODES: 0
  NUMBERED SOURCES: 0
  DUMMY NODES: 0

COMMENTS? yes
COMMENT: extended central server model

CHAIN: 1 TYPE: closed
COMMENT: jobs have a think time at a terminal.
: then they request memory.
: after being allocated a partition, they determine
: their number of processing/data transfer cycles.
: after this number of cycles they release their
: partition and go back to the thinking state.
:
( 1): 1->2->3->4->5 6; .2 .8
( 2): 5 6->7->4 8;jv=0 jv=0
( 3): 8->1
( 4):
CHAIN POPULATION: 20

QUEUE 1 TYPE: is
COMMENT: terminals (is -- infinite server)
:
CLASS LIST: 1
STME. DISTR: 5

QUEUE 2 TYPE: passive
COMMENT: memory partitions
:
TOKENS: 5
QDSPL: fcfs
ALLOCATE NODE LIST: 2
  AMOUNT(S): 1
RELEASE NODE LIST: 8

QUEUE 3 TYPE: active
COMMENT: central processing unit
:
SERVERS: 1
QDSPL: ps
CLASS LIST: 4
WORK DMND. DISTR: <.002,1>
JV SCALED: no

SETUP for extended central server model
addition to the two basic queue types, *active* and *passive*, there are several special cases for simplified active queues. These simplified cases are indicated by responding with a queueing discipline to the prompt for queue type. The response to the queue type prompt of "active" allows all of the options described above. The distributions, e.g., the work demand distribution for each class, may be specified as either a single value which is interpreted as the mean of an exponential distribution, or as a pair of values in angular brackets ("<", ")") which are interpreted as a mean and coefficient of variation, respectively. If all classes of the queue are to have the same distribution, the list may be replaced by a single distribution which will be used for all classes.

**Passive queues**

A passive queue consists of a pool of tokens, a non-empty set of waiting areas for jobs requesting or possessing tokens, a possibly empty set of other nodes for actions on the queue and a control mechanism for the tokens and jobs.

```
SERVER 1:
  RATE: .1
  ACCEPTS: all

QUEUE 4 TYPE: active
COMMENT: disk
  SERVERS: 1
  QOSPL: fcfs
  OVHD: none
  CLASS LIST: 5
  WORK DMND. DISTR: .044
  JV SCALED: no
  SERVER 1:
    RATE: 1
    ACCEPTS: all

QUEUE 5 TYPE: fcfs
COMMENT: drum
  CLASS LIST: 6
  STME. DISTR: .008

SET NODES: 3 7
  SET TO: r -1

  DISTRIBUTION FOR NODE 3
  VALUES: 10 20
  PROBS: .5 .5

END OF SETUP.
```

**Definition of Set Nodes**

```
SETUP for Extended Central Server Model

CLASS
{ } ←→ { }
CLASS
{ } ←→ { }
SERVERS
Active queue with two classes and two servers
```

From the collection of the Computer History Museum (www.computerhistory.org)
The tokens of the passive queue are analogous to the servers of an active queue. The waiting areas are called allocate nodes. There are three other types of nodes which may be associated with a passive queue: release nodes, destroy nodes and create nodes. The usual purpose of passive queues is to limit or measure the population of subnetworks.

Allocate nodes

A job arriving at an allocate node requests possession of a number of the queue's tokens. If the tokens requested by a job are available at the time of arrival at an allocate node, then the request is satisfied instantaneously. Otherwise the job must wait until sufficient tokens become available and are assigned to the job. (Tokens become available through the action of other jobs at other nodes.) As soon as the request for tokens is satisfied, the job is allowed to visit other nodes of the network. However, as long as the job possesses tokens of a given queue, it is considered to be part of that queue. Thus a single job may be a member of one or more passive queues and one active queue simultaneously.

Release nodes

When a job visits a release node associated with a queue which the job is a part of, the job instantaneously returns all its tokens belonging to the queue. When a job visits a release node associated with a queue which the job is not a part of, there is no effect on the job or the queue. In either case the job's visit to the release node is instantaneous and the job proceeds without delay.

Destroy nodes

When a job visits a destroy node associated with a queue which the job is a part of, the job instantaneously destroys all its tokens belonging to the queue; then the job is no longer part of the queue. When a job visits a destroy node associated with a queue which the job is not a part of, there is no effect on the job or the queue. In either case the job's visit to the destroy node is instantaneous and the job proceeds without delay.

Create nodes

A job visiting a create node adds new tokens to the pool of its associated queue. The number added is determined by sampling from a discrete distribution associated with the node. There is no effect on the job; its visit is instantaneous and it proceeds without delay.

Sources

A source emits jobs one at a time. The time between a given arrival from a source and the next arrival from a source is determined by a sample from a continuous distribution associated with the source. The job variable is set to zero when the job is emitted. The description of sources is included in the dialogue describing the routing.

Sinks

Sinks are nodes which allow jobs to exit from the network. A job exiting from the network releases all tokens held, if any, and returns them to the appropriate pools. The exiting process is instantaneous. The description of sinks is included in the dialogue describing the routing.

Set nodes

A set node is used to affect the value of a job variable. A job's variable will be zero unless it has been given some other value by a set node. Job variables are useful for making work and overhead demands job dependent. They are also especially useful for effecting deterministic job dependent routing, e.g., to cause a job to cycle through a set of nodes for a predetermined number of cycles. There are five kinds of set nodes, assignment set nodes, increment set nodes, decrement set nodes, change sign set nodes and previous node set nodes.

An assignment set node assigns a non-negative value to a job's variable, an increment set increments a job's variable by a non-negative value and a decrement set node decrements a job's variable by a non-negative value. In any of these cases, the values used are samples from a distribution. The distribution is associated with the set node and may be either continuous or discrete. A change sign set node changes the sign of the job variable. A previous node set node assigns to the job variable the identity of the node the job just left. A job's visit to a set node is instantaneous.

Fission and fusion nodes

A job arriving at a fission node generates one or more additional jobs. The generating job is referred to as the
parent and the generated jobs are referred to as offspring. A parent job and its offspring are considered to be related and know the identities of each other. Each of these jobs has a separate routing from the fission node. The visit of the parent job is instantaneous; the offspring depart from the node immediately after generation. The offspring do not possess any tokens; their job variables have the value zero. Combinations of fission and fusion nodes are useful for representing packetizing of messages in a communication network. They are also useful in models of computing systems to represent overlap of processing and data transfer.

A fusion node provides a waiting area for related jobs. Related jobs wait at a fusion node until they have no relatives; they then depart instantaneously. If a job arrives at a fusion node where it has a related job, one of the jobs is eliminated from the network instantaneously. If one of the jobs is the parent of the other then the other job is eliminated. If both of the jobs are offspring then it is left undefined which job will be eliminated. Any tokens possessed by the eliminated job are returned to the appropriate pool. (Fusion nodes have no effect on jobs without relatives. If a job is waiting at a fusion node and all of its relatives leave the network, the job departs from the node immediately.)

The description of fission nodes appears in the routing dialogue. Fusion nodes are identified by a single prompt for a list of fusion nodes.

Split nodes

Split nodes are like fission nodes with the difference that the generating job and the generated jobs are independent of each other; they are not considered to be related. Split nodes are useful in representing bulk arrivals. They are also useful in communication network models to represent the generation of control messages. The description of split nodes appears in the routing dialogue.

Routing

All nodes except fission nodes, split nodes and sinks may have several alternate routing paths for jobs leaving the node. Fission and Split nodes have separate fixed routing paths for the creating job and each created job. (Dummy nodes may be used with fission or split nodes to provide alternate routing paths for jobs leaving those nodes.) A node with alternate routing has a list of possible routings. Each item on the list consists of the identity of a possible destination node and either a predicate or a probability. The predicates are statements about values of job variables, availability of tokens, etc. A job leaving a node with alternate routing selects a destination by scanning the list until it finds a predicate which is true or until it succeeds at a Bernoulli trial with one of the given probabilities. If the list is exhausted without a node being selected, the results are undefined. A job travels from one node to the next instantaneously.

The nodes of a network are separated into one or more disjoint sets called chains. A chain is defined as the largest subset of nodes such that all nodes of the chain are connected. Open chains are those that include sources and/or sinks. All other chains are closed.

The routing is described separately for each chain of nodes. Before prompting for the routing description, SETUP prompts for the type of the chain, open or closed. Then there is prompting for routing transitions. A routing transition consists of a list of nodes, a list of alternative destinations for those nodes and a list of probabilities and predicates. Transitions may be entered individually or the concatenation of two or more transitions may be entered simultaneously. The prompt for a transition or concatenation of transitions is a parenthesized integer. The value of the integer has no significance except to number the entries the user has made during the input dialogue. The prompting for transitions or concatenations of transitions continues until a null line is entered.

Consistency checks and error messages

After the dialogue is complete, SETUP goes through a variety of consistency checks to look for errors in the model or incompatibilities with the solution technique. If errors are found, an error message (presumably a self-explanatory error message) will be given.

MODEL LISTING, SOLUTION AND EVALUATION

LIST

The command LIST produces a tabular listing of a model. The listing includes, in order, the solution method, the model name, the numbers of elements, the characteristics of the chains, the characteristics of the queues and the characteristics of the nodes. Headings are suppressed for columns which have no entries, e.g., "classes accepted" for networks with only symmetric queues.

EVAL

The command EVAL is used to apply the solution method specified with the model and to examine the results produced by the solution method. The most general solution technique is simulation (APLOMB). Numerical (QNET4) solutions are available for a subset of the class of models simulated.

APLOMB

APLOMB is a simulation program specifically designed for the class of queueing networks represented in RESQ. A major feature of APLOMB is its capability for determining confidence intervals for simulation results. (Note that even if this capability is used incorrectly and APLOMB is unable
list
MODEL NAME: ecrm
SOLUTION METHOD: APLOMB
MODEL NAME: ECSM

1 CLOSED CHAIN(S)
0 OPEN CHAIN(S)
5 QUEUE(S)
4 CLASS(ES)
8 NODE(S)

COMMENT: EXTENDED CENTRAL SERVER MODEL

CHAIN| TYPE | POP | COMMENT
-----|------|-----|---------------------------------------------
1 | CLOSED | 20 | JOBS HAVE A THINK TIME AT A TERMINAL.
    |       |    | THEN THEY REQUEST MEMORY.
    |       |    | AFTER BEING ALLOCATED A PARTITION, THEY DETERMINE
    |       |    | THEIR NUMBER OF PROCESSING/DATA TRANSFER CYCLES.
    |       |    | AFTER THIS NUMBER OF CYCLES THEY RELEASE THEIR
    |       |    | PARTITION AND GO BACK TO THE THINKING STATE.

QUEUE| TYPE | Q | DSP | MS | MQ | RATE(S) | C.A. | COMMENT
-----|------|---|-----|----|------|--------|-----|---------------------------------------------
1 | ACTIVE | 1 | 1 | 1 | 1.00 | TERMINALS (IS -- INFINITE SERVER)
2 | PASSIVE | FCFS | 5 | 1 | MEMORY PARTITIONS
3 | ACTIVE | PS | 1 | 1 | 0.10 | 4 | CENTRAL PROCESSING UNIT
4 | ACTIVE | FCFS | 1 | 1 | 1.00 | 5 | DISK
5 | ACTIVE | FCFS | 1 | 1 | 1.00 | 1 | DRUM

NODE| Q | CHN| TYPE | WD/ST | CW/ST | DDV | DDP
-----|---|----|------|-------|-------|-----|-----
1 | 1 | 1 | CLASS | 5.00 | 1.00 |
2 | 2 | 1 | ALLOC | 1.0 | 1.0 |
3 | 1 | 1 | SET | 10.20 | .5 .5 |
4 | 3 | 1 | CLASS | .002 | 1.00 |
5 | 4 | 1 | CLASS | .044 | 1.00 |
6 | 5 | 1 | CLASS | .008 | 1.00 |
7 | 1 | 1 | SETDC | 1.00 | 0. |
8 | 2 | 1 | RELSE | |

FROM| TO | INDICATOR | VALUE
-----|----|----------|-----
1 | 2 | 1.00 |
2 | 3 | 1.00 |
3 | 4 | 1.00 |
4 | 5 | .20 |
4 | 6 | .80 |
5 | 7 | 1.00 |
6 | 7 | 1.00 |
7 | 4 | JV= | 0.
7 | 8 | JV= | 0.
8 | 1 | 1.00 |

LIST for central server model
evaluation
MODEL NAME: ecsm
INITIALIZE: 20 0 0 0 0 0
REGEN: 20 0 0 0 0 0

SEQUENTIAL SAMPLING LIMITS:
  CYCLES: 10
  STATE CHANGES: 50000
  CONFIDENCE LEVEL: 95
  CHECK QUEUE: 2
    RELATIVE PCT. INTERVAL WIDTH: 10
  SEED: 314159

NO ERRORS DETECTED DURING SIMULATION

SIMULATED TIME: 1217
NUMBER OF STATE CHANGES: 114020
NUMBER OF CYCLES: 140
CORRELATION OF CYCLE LENGTHS: .010

WHAT: how
UT=UTILIZATION, QL=MEAN QUEUE LENGTH, SDQL=STD. DEV. OF Q.L.,
QT=QUEUEING TIME, TP=THROUGHPUT, PO=POPULATION, RT=RESPONSE TIME,
ALL=ALL OF ABOVE.
TRY AGAIN:
WHAT: qt
CONFIDENCE INTERVALS, POINT ESTIMATES, OR BOTH? both
QT:
  Q 1: 4.97E+00 1.67E+00 6.55E-02
  (4.82E+00,5.13E+00) (1.58E+00,1.75E+00) (6.37E-02,6.72E-02)
  Q 4: 6.84E-02 1.10E-02
  (6.61E-02,7.08E-02) (1.09E-02,1.12E-02)
WHAT: ut
CONFIDENCE INTERVALS, POINT ESTIMATES, OR BOTH? point
UT:
  Q 1: .690
  Q 4: .396
  Q 2: .797
  Q 5: .901
  Q 3: .291
WHAT: ql
CONFIDENCE INTERVALS, POINT ESTIMATES, OR BOTH? con
QL:
  Q 1: 1.46E+01 1.52E+01
  (1.46E+01,1.52E+01) (4.74E+00,5.31E+00) (2.85E+00,3.08E+00)
  Q 4: 5.85E-01 6.45E-01
  (5.85E-01,6.45E-01) (3.92E-01,4.12E-01)
WHAT:

EVAL for central server model

The occurrences of this state divide the simulation into independent cycles. Presumably the simulation will enter this state frequently. Usually the regeneration state will be chosen so that there are no jobs in any nodes belonging to open chains. Usually the regeneration state will be chosen so that jobs of closed chains are distributed among the nodes according to expected populations of these nodes. Of course one usually will not know in advance the populations at the various nodes; reasonable guesses are usually

to provide confidence intervals, APLOMB will still provide the user with point estimates.) When the solution method specified for the model is APLOMB, EVAL prompts the user for parameters used in determining confidence intervals and in controlling the simulation.

Initialization and regeneration
In order to apply the confidence interval techniques, one must specify a system state called the "regeneration" state.
sufficient and poor guesses are often workable. Since visits to most nodes are instantaneous, non-zero expected populations are only reasonable at classes, allocate nodes and fusion nodes. The first prompt from EVAL is for the number of jobs to be initialized at each node. The reply should be a list of non-negative integers, with as many elements in the list as named nodes in the network. The second prompt is for the number of jobs at each node in the regeneration state.

Sequential sampling procedure

The next series of prompts determines when the simulation will stop. APLOMB uses a sequential sampling procedure to run the simulation until satisfactory confidence intervals are obtained. The procedure has two limits, on the number of regeneration cycles and on the number of state changes, to control the period between samples. The sampling period ends when the first of the two limits is reached for that period. If after the first sampling period has ended too few cycles have been completed to compute confidence intervals, the simulation ends and only point estimates are provided. Otherwise, the width of the confidence interval for the mean queueing time at a given queue, relative to the point estimate of the queueing time for that queue, is compared to a threshold. If the threshold is exceeded, then a new sampling period is begun. Sampling periods continue until the relative width does not exceed the threshold. The first prompt of the series requests the sampling period limit on the number of regeneration cycles. The second prompt of the series requests the sampling period limit on the number of state changes. The third prompt is for the confidence level of the intervals. The fourth prompt is for the number of the queue to be used in determining whether to continue sampling or not. The final prompt is for the threshold for the relative width of the confidence interval for the mean waiting time at the queue. The relative width is expressed in percent of the point estimate. Notice that the sequential sampling procedure can be defeated by specifying a very large threshold, e.g. 200 percent.

Seeds for pseudo-random streams

The final prompt before simulation begins is for an integer to be used as a seed for the pseudo-random number streams. Each random variable in the network has its own stream. Each of these streams has its own seed. The user specified seed is used to start a stream which produces seeds for all of the other streams.

Simulation results

After the simulation ends, EVAL will either respond "NO ERRORS DETECTED DURING SIMULATION" or will give an error message if an error was discovered during simulation. Then EVAL gives the simulated time, the number of state changes that occurred, the number of regeneration cycles completed and an estimate of the correlation between the lengths of successive regeneration cycles. (This estimate should be near zero if the state chosen is actually a regeneration state.)

EVAL now prompts the user with "WHAT:" and is ready to provide simulation results. The user replies with a code indicating the type of results desired. Results of that type are given for all appropriate elements (queue, node or chain).

QNET4

If the solution method is QNET4, then no solution dependent information is required. The prompting begins immediately with "WHAT:" and the same codes are used as with APLOMB. There is no option for confidence intervals since the QNET4 values are exact within the limits of numerical error.

CHANGE

CHANGE allows more or less arbitrary revisions of a model. Most of the dialogues are similar to dialogues occurring in SETUP. Unless otherwise stated, the same responses may be given in CHANGE as may be given for the corresponding prompts in SETUP. Often additional information is provided for the user's reference.

SUMMARY

RESQ makes possible and convenient the solution of a great variety of queueing network models. The many constructs allowed given the modeler the freedom to study a variety of system characteristics and to determine the degree that various characteristics impact performance.

In addition, RESQ provides a variety of solution techniques. Thus the modeler can make tradeoffs between expense of solution and model accuracy. The modeler can use a combination of techniques to form a hybrid solution. The solution techniques provided are the best available for this class of problems. Proposed solution techniques are being included in RESQ on an experimental basis. The multi-solution technique capability is also helpful in studying these proposed techniques.

Finally, the user interface is designed for convenience for a variety of users. Interactive dialogues are provided. Subroutine level interfaces are also available for repeated or specialized usage of RESQ.

REFERENCES