Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

A TUTORIAL ON GENETIK SIMULATION AND SCHEDULING

Kieran Concannon
Paul Becker

Insight International
2 Robert Speck Parkway
Mississauga, Ontario, Canada L4Z 1H8

ABSTRACT

GENETIK is pioneering the application and development of the Visual Interactive approach for solving a wide range of management and operational problems. With this technique we have developed a number of tools that can be applied to decision-making in a variety of situations. Included amongst this growing list of products is **GENETIK**.

GENETIK is also used as the base for three other simulation and scheduling systems. **G99** is a higher level simulation system, **P99** is a scheduling system also designed for the non-specialist, and **PLEXUS** is a Local and Wide Area Network telecommunications simulator.

1. OVERVIEW

GENETIK is a powerful general purpose visual interactive modelling system that includes both **Simulation** and **Scheduling** modules. This tutorial provides a detailed description of the **GENETIK** system and its features, with examples of its use as a **DSS** generator, a **simulation** modelling tool and its growing application in the role of **planning and scheduling**.

GENETIK is a family of software products that are closely related to each other. The system provides a fully interactive environment that enables you to develop friendly models and turn-key decision support systems with ease. Your application users can be offered customized interactive systems supported by mouse, function keys and menus. In addition, special modules for **simulation** and **scheduling** are provided to bring these techniques into the exciting realm of visual interactive models.

2. CORE SYSTEM

GENETIK's core system provides a user-friendly modelling environment, where 'turn-key' application models can be created quickly and easily. Models are built in a **fully interactive** environment by defining the units to make up your application. All models consist of four basic unit types: **Graphic, Logic, Data** and **Interactions.**

2.1 Graphic

Graphic units which are used to show and animate your model, include multiple **pictures** of infinite size. These can be edited directly, drawing any size of graphic, icons, queues, data variables etc., onto the picture. **Windows** are then used to map, pan and overlay pictures onto multiple **Screens**. Specific graphics in the form of **Icons** can be created, of any shape and size, and added to pictures or saved in your personal library. Visual displays can be easily specified and invoked at any time to show model data in the form of histograms, tables, graphs, etc. The appearance and contents of these screen displays is completely flexible.

2.2 Logic

Logic units are written with **GENETIK**'S own language syntax which is small, simple to learn yet very powerful. Logic is written in the form of blocks, that may have conditional entry, exit and loops. A full-screen editor is used to build and edit logic units and provides automatic syntax checking, on-line help, cross-referencing of units and variables, copying and moving blocks, etc.

2.3 Data

Data within your model can be global or local variables, or two-dimensional **Tables** of information. **Tables** are a unique and powerful construct within **GENETIK** that allow you to build relational data structures for any application. A table can be user-defined with multiple column types and a large number of rows (up to 32,000 rows each). Cells may contain either explicit data or pointers to other tables or model logic. Rows can be added to these tables and dynamically organized using powerful utilities within **GENETIK** to sort, scan and update table contents. **Data** can be imported from other external sources such as dBASE files. Similarly, **GENETIK** data can be exported to files or applications.

2.4 Interactions

Interactions form a vital ingredient to any model providing user control over your specific application. User-friendly interactions can be developed using **menus, function keys** or **mouse** pointers.

2.5 Extensions

A wide variety of problems can be modeled using the **GENETIK** core system. But additional modules are provided to help with specific types of applications. These are the **Simulation** and **Scheduling** modules which are explained later.

3. SIMULATION

GENETIK simulation provides the following powerful features:

General

Direct Modelling
Trace Routines
Debugging Messages
Open Architecture
Stop and Save Anytime
Copy Model/Elements
Capture any Statistic
Live Reporting and
Animation

Queuing Facilities
- Automatic Graphics
- Member Positioning
- Setting and Checking
- Multiple Memberships
- Dynamic Sizing
- Emptying, Linking
- Inspection

Distributions
- Weibull
- Uniform Real
- Uniform Integer
- Triangular
- Binomial
- Poisson
- Lognormal
- Uniform

Animation
- Timed or Immediate
- Icon Movements

Random # Streams
- Infinite # Seeds
- Resetting Streams

Simulation Types
- Discrete Event
- Continuous

Simulation is a useful tool to analyze dynamic operations. With the **GENETIK** Simulation module you get a ready-made framework for building discrete-event and continuous simulation models quickly and easily. A common use of this **Simulation** module is to model product flows in a plant or warehouse; it is also used in situations as diverse as modelling telecommunication networks or hospital accident and emergency units.

Live visual displays of information with animation are standard features of this system. In addition, an extensive library of statistical functions and queuing utilities go into to make up the Simulation module. User-friendly, on-screen interactions can be specified to front-end your application. Models may be stopped at any point in time, and the user may then make changes to any element of the model.

For example, conveyor speeds, machine characteristics and resource availability directly from the screen without editing the underlying logic. After this, the model run can be continued to observe the effects of such changes.

GENETIK Simulation provides real-time screen displays of the 'system' working with colour animated graphics. In the case of a plant, products can be seen on the shop-floor waiting to be machined, conveyors moving materials to work areas, fork-lift trucks scurrying around the plant, machines manned and working or sitting idle, etc., - watching it will give you new ideas on how you can improve your current operating methods.

An important aspect about a **GENETIK Simulation** is that you can mimic the control rules used in your factory exactly and experiment with them directly. For example, the priority and sequencing rule for processing jobs queuing in front of your machines may be more sophisticated than 'first-come, first-served' when deciding which job to do next. You might have a rule which looks at the work remaining on each job in the queue and the work in front of an up-stream (bottleneck) machine.

Through these **GENETIK Simulations** you can test out ideas such as facility changes, replace old equipment with the latest technology, try out J.I.T. methods in your plant, all in a safe, quick and realistic manner.

4. SCHEDULING

GENETIK scheduling provides these powerful features:

Standard Interactions
- Panning across Time
- Zooming the Schedule
- Scrolling Resource Profiles

Scheduling Algorithms
- Standard Algorithms
- Tailored to Suit
- Manual Override

Screen Presentations
- Gantt Chart Views
- Split Screen Views
- Customized Layouts
- Exception Reporting
- Resource Profiles
- Order Status Reports
- Shop Floor Screens

Data Linkages
- Standard ASCII Files
- Direct Link to Dbase
- Linkage to MRP Files
- Export to Mainframe
- Order-Entry Files

On-Screen Interactions
- Mouse Pointers
- Menu & Fkey Support
- Job Selections for Resequencing or Status Enquiries

Save & Restore Plans
- Save Schedules onto Disk
- Restore, Review Schedules
- Experimental Versions

Output of Schedules
- Customized Hardcopy Prints
- Shop-Floor Work-Orders
- Colour Prints of Schedules

4.1 Planning Boards

Effective scheduling is one of the keys to profitability in any organization. The **GENETIK** Planning Board Extension enables you to have an electronic planning board with which to perform your scheduling quickly, effectively and knowledgeably. The **Planning Board** presents schedules in the form of Gantt charts. Each row typically represents a machine or work centre against which jobs or activities are shown over time.

Planning Board solutions can take account of any set of constraints or objectives relevant to your plant. For example, it can consider...

- order delivery dates
- inventories
- production dates
- process rules
- labour availability
- shift patterns
- multiple routes
- priorities
- unexpected breakdowns
- raw materials
- replenishment times
- work-in-process
- JIT philosophy
- maintenance
- product routes
- interrelated orders
- late orders

... any factor relevant to your plant.

Its benefit to scheduling are threefold:-

1. It gives the scheduler direct access to the schedule, through on-screen interactions, to instantly update or 'what-if' the plan.

   Thus when a change to the plan is required (perhaps an urgent order arrives or a machine breaks down) a change can be made directly and the schedule automatically updated by the computer.

2. It can be used to produce an 'Initial Solution' that is designed to meet the various company objectives and plant constraints that you impose.

   This initial plan can be manipulated or 'fine-tuned' to meet your specific requirements. These revisions can be made showing you the consequences to operating costs, manpower, space, due-dates...

3. The **Planning Board** can highlight potential scheduling problems in advance and profile resource requirements through sophisticated graphical displays.

   For example, late orders or resource shortages can be identified in the plan ahead of time, through flashing colour codes on the planning board. The plan can then be revised interactively to avoid potential problems.

GENETIK Planning Boards provide the Scheduler with the means to see and develop the current schedule, **react quickly** to changing conditions, **test-out ideas**, and **maintain** a good feasible schedule.

5. GENETIK UNIT STRUCTURE

GENETIK models are made up of Units of different types. These units are classified into four major categories that are described in the following sections.

5.1 Visual Units

When you use the Display Utilities, you are outputting information to the 'current' picture.

5.2 Table Units

Table Units provide a powerful means of storing and manipulating data in a **GENETIK** model. A Table is a two-dimensional array of data in the form of Columns and Rows. **GENETIK** allows several different types of data to be held in a Table.

5.2.1 Table Structure

GENETIK Tables can be structured to contain integer, real and text information within the columns of the table. Cell entries can also be used to reference other tables, or table rows or table columns. In this way the user can develop a relational data structure for his or her application. One other type of column heading within a table is that called an action unit, which allows you to specify within that column logic units contained within the model for execution within a simulation model.

An **EVENTS** table, for example, can contain an action column and a table row column. The sysindex column which is standard within all tables, is used to order the rows in the table in chronological or ascending order. If you alter the value of a specific row sysindex you will see that it automatically relocates itself to conform to the ascending order rule within the table. Following sections explain how these features are used to access data from tables.

5.2.2 Table Handling Utilities

Here are some of the more frequently used Table Utilities. An asterisk (*) indicates a parameter whose value is returned by **GENETIK**.

SIZE - Find the size of a table
For a specified Table, this utility will return the maximum size of the Table and its current size.
SIZE : *Current size : *Maximum size : Table
Example **SIZE** : INUMROWS : IMAXROWS :TPRO

GETROW - Get a Table/Row Pointer from a Table and position.
For a specified position in a specified Table, this Utility will return the Table/Row Pointer to the Row in that position.
GETROW : *Table/Row Pointer : Table : Position
Example **GETROW** : TROW : TPRODUCT : 1

ADDROW - Add a new Row to a Table
This Utility will add a new Row into a specified Table and return the Table/Row Pointer allocated to this new Row.
ADDROW : *Table/Row Pointer : Table
Example **ADDROW** : TROW : TPRODUCT

DELROW - Delete a row from a table
This Utility will delete a particular Row from a Table.
DELROW : Table/Row Pointer
Example **DELROW** : TROW

EMPTY - Empties a table
This Utility will delete all the Rows in a particular Table.
EMPTY : Table
Example **EMPTY** : TPRODUCT

5.2.3 Accessing Data in Tables

For every Row that is added into a Table, **GENETIK** maintains internally the position of the Row in the Table and also a unique identifier of that Row in that Table.

When a Table is first created, Rows are added in strict ascending order - the position and Row Id are identical. If we delete a Row other than the last Row, or change the value in Column SYSINDEX (see later), the position of a Row may change, but the Row identifier does not change. When the table is displayed on the screen, the position and identifier (RowId) are

shown on the left, and cannot be edited. The Row identifier is known in **GENETIK** terminology as the Table/Row Pointer.

In order to access an item of data within a Logic Unit, the Table/Row Pointer to the Row containing the required item of data must first be obtained. **GENETIK** provides a Utility to perform this task (and Utilities for other Table handling functions). Once we have the Table/Row Pointer (in Table/Row variable TRP, say) we can combine it with the required Column name (CNAME, say) thus:

TRP.CNAME
to obtain the data item.
We can now use this form of variable in the logic. For example, if CMACH1 is an integer column, then
TPRODUCT.CMACH1
is an integer variable; and we can write statements such as
TPRODUCT.CMACH1 = 5
to assign a value in a Table, or
ITOT = ITOT + TPRODUCT.CMACH1 * TPRODUCT.CQTY
where we are using values extracted from a Table.

5.3 Special Tables

There are several special forms of Tables that are used within **GENETIK** these are **ENTITY, LISTS, HISTOGRAM, GRAPHS, SCHEDULES.**

5.3.1 ENTITY Tables

Entity tables are use to contain common elements within a simulation model such as machines or parts that are changing states as they progress through time. These elements or entities, which will have the same set of attributes, are expressed in individual tables where the attributes of the entity such as size, speed or shape, are expressed through table columns.

A special feature of **ENTITY** tables is that the rows of these tables which can be referred to as an individual entity of that type, can be added to queues or **LISTS** as they are known in **GENETIK**, that may serve to show the state of the entity and its position relative to other entities. A standard use of this feature would be a queue of parts waiting to be machined in front of a machine. The individual parts would be shown on the computer screen in the order they were added to the **LIST**.

5.3.2 LIST Tables

These tables allow the user to specify queues within a simulation model and if required, display the contents of the queues on the computer screen. The queues will typically be changing over time with respect to the number of entities that are queuing at any point in time. **GENETIK** automatically displays the contents of the queues as entities are added and deleted to the **LIST** tables.

5.3.3 HISTOGRAM and GRAPH Tables

These tables allow the user to collect statistics from the model and display them in the form of a histogram or graph on the computer screen. The row elements of these tables are the x,y observation in the case of the **GRAPH** table and the cell-interval observation frequency in the case of the **HISTOGRAM** table.

5.3.4 SCHEDULE Tables

Tables Sequencing And Ranging

It is sometimes useful to sequence the Rows in a Table in a particular order. For example, if we have a Table of orders for a product, then we may want to process the smallest orders first and hence it would be useful to sequence them in the Table in ascending size.

The position of each Row in a Table can be controlled by the value in column SYSINDEX. In the example above, if we were to enter the order quantity for each order into column SYSINDEX in the Table, the Table would appear sequenced in ascending order size. The Row Pointer for each Row will be unchanged; it is only the position of each Row in the Table that has changed.

Given this ability to maintain the Rows of a Table in a particular sequence, it now becomes desirable to access a particular section of a Table. For example, returning to the above Table of orders, we may only be able to process an order quantity of 1000 today and subsequently only need to access those orders in our order Table.

GENETIK provides a set of Utilities which will scan a section of a Table and return the Table\Row Pointers of any Rows that it finds within that section. The scan is controlled by the values in the SYSINDEX Column; thus careful use of this Column means that we can arrange and access individual areas within a Table. This selective accessing is known as **Ranging** in **GENETIK** terminology.

Typical Range Structure

Having sequenced the Rows in a Table by entering values into the SYSINDEX Column of the Table, we can search through the Table from one value of SYSINDEX to another (these values specify the Range), and pick up the Table/Row Pointer to successive rows within that range. This procedure is executed as follows:

```
Setrange : TABLE : VALUE1 : VALUE2
[LOOP] Loop
   Getrange : TROW : ROWSLEFT
   Leave [LOOP] if ROWSLEFT << 0

                |

[LOOP] Next
Endrange
```

VALUE1 and VALUE2 are two integer numbers. **GENETIK** will search the SYSINDEX column of the specified Table looking for any values which lie between these two limits (inclusive). If VALUE1 is less than VALUE2, the search will proceed 'down' the Table; if the reverse is true, the search will proceed 'up' the Table.

Utility 'Getrange' will return the Table/Row Pointer (TROW) to any Row found within the range currently set by Utility **'Setrange'**. 'Getrange' is usually placed within a loop to enable all the Rows within the range to be accessed, should there be more than one. The number of Rows left within the range (ROWSLEFT) is also returned. This tells us when there are no more Rows left in the Range.

At the end of the 'Ranging' process the range must be 'closed' so that there is no confusion with any ranges which we may set up later. This is done by invoking the Utility 'Endrange' (no parameters). An 'Endrange' must accompany every 'Setrange'. If no Rows are found within a range, or there are no more Rows left in that range, then 'Getrange' returns a value of -1 in parameter ROWSLEFT.

6. LOGIC UNITS

Logic Units are the mechanism by which we specify the 'processing' which our model is going to perform.

A Logic Unit consists of statements. One simple form of statement is an assignment, such as:

```
LIX = 21
```

which assigns the constant value 21 to the variable LIX.

The right hand side of an assignment can be an expression, such as

```
LIX = 2*LIX + LIW/10
```

which means 'multiply the current value of LIX by 2, divide the current value of LIW by 10, add the results, and put the answer in LIX'.

We can also invoke other Units by name. For example, the statement

```
SCDURN
```

will clear the screen and set up **SCDURN** as the current Screen. Whatever is windowed to Screen Unit **SCDURN** will appear on the screen.

6.1 Action Units and Utility Units

There are two basic types of Logic Unit - Action Units and Utility Units. You can pass values to and receive values from a Utility Unit by using parameters. **GENETIK** contains many 'ready-made' Utilities, and you can write your own. Note: every **GENETIK** model must contain an Action Unit called STARTUP. This Action Unit is invoked whenever F7 (Run) is pressed to start running the model.

6.2 Editing Logic Units

GENETIK usually progresses line-by-line through the logic. However, it is often necessary to change this order of execution depending on some condition or other. For example, we may want to follow one set of logic for product A, a different set for product B, and so on. **GENETIK** allows us to do this by means of blocks and loops.

There are several kinds of block and loop constructs in **GENETIK**:-

'Begin if...' (created using **F7:Begin**):-

```
[BLOKNAME] Begin (if ... )
        :
        :
        :
                Else (if .... )
        :
                Else
[BLOKNAME] End
```

'Loop', 'Loop while', 'Next until' (created using **F8:Loop**):-

```
[LOOPNAME] Loop (while ... )
                :
                :
                :
[LOOPNAME] Next (until ... )
```

'Loop for' (created using **F9:Loop for**):-

```
[LOOPNAME] Loop for COUNT = MIN to MAX step
STEPSIZE
                :
                :
                :
[LOOPNAME] Next
```

We can also jump around the logic (i.e. decide where to go next) by using one of these 'control statements':-

Enter [BLOKNAME] (**if** ...) - jumps to the start of block or loop BLOKNAME

Leave [BLOKNAME] (**if** ...) - jumps to the end of block BLOKNAME

Next [LOOPNAME] (**if** ...) - jumps to the top of the loop without resetting the count.

The Logic Editor uses Function Keys to input control words like **'Enter'** and **'Leave'** - you must not type them.

7. INTERACTION UNITS

Menu Units and Function Key Units (called FKEYMODE Units) are often employed to allow the user to control the operation of a **GENETIK** model.

7.1 Menu Units

A **GENETIK Menu** consists of a number of options. Each option has a string of text which will be displayed when the **Menu** is invoked, and a corresponding Action Unit which will be invoked if this option is selected.

Within the logic, a **Menu** is invoked by typing its name. When it is invoked, the **Menu** must be on a Picture in such a position that it is currently windowed to the screen, otherwise an error will occur. When **GENETIK** encounters a **Menu**, control is passed to the user to select one of the **Menu** options. Once an option is selected the Action associated with that option is executed and then **GENETIK** returns to the line following the **Menu** invocation. To return to the same **Menu** after invoking the selected Action, place the **Menu** invocation in a loop, e.g.

[MENULOOP] **Loop**
▌**MENU1**
[MENULOOP] **Next**

7.2 Function Key Units

A **GENETIK** Function Key Unit allows you to associate up to ten Action Units with the keyboard Function Keys F1 to F10. Like each option in a Menu Unit, you can specify a text string for (optional) display and an Action Unit to be invoked when the corresponding Function Key is pressed.

The text strings of a Function Key Set can be displayed on a picture. The current Function Key Set will still be operative even if not displayed. If more than one Function Key Set is required they can all be placed at the same location on a Picture - only the current set will be displayed.

7.3 Mouse Utilities

All **GENETIK** interactions can be performed with the use of a mouse. The mouse can be used when building a model to select commands or locations on the screen for graphics creation. **GENETIK** models can also be built to be completely mouse and menu interactive so that minimal modelling and computer skills are required to set the system parameters and to test alternate scenarios.

8. GENETIK LIST HANDLING UTILITIES

In these descriptions of the List-manipulating Utilities, the word 'Member' means an Entity, ie a Table/Row Pointer to a Row in an Entity Table, and a List is a Table in **GENETIK** terms. An asterisk (*) indicates a parameter whose value is returned by **GENETIK**.

ADDMEMBER - Adds a Member to a list
This Utility adds a Member (ie an Entity) to a List in a specified position.
ADDMEMBR : Member : List : Position in List
Example: ADDMEMBR : PRODUCT : QWORLD : -1
Note : Position -1 adds a member to the back of a list.
DELMEMBR - Deletes a Member from a List
This Utility deletes a Member (ie an Entity) from a specified position in a List and returns the Entity Pointer (Table/Row Pointer) to the deleted member.
DELMEMBR : *Member : List : Position in List
Example: MEMBR : PRODUCT : QWORLD : 1
Note : Position 1 deletes a Member from the front of a List.
GETMEMBR - Gets the Entity Pointer to a Member in a List
This Utility gets the pointer to a member in a given position in a List.
GETMEMBR : *Member : List : Position in List
Example GETMEMBR : PRODUCT : QWORLD : 1
EMPTY - Empties a List
This utility deletes all the Members from a List.
EMPTY : List name
Example: EMPTY : QWORLD

9. SUMMARY

GENETIK provides a Visually Interactive means of providing analysts a fast, easy and powerful tool for developing customized simulation and scheduling systems. Data, Pictures, Logic and Interactions are the four key requirements in building a **GENETIK** model. By using simple facilities to handle these four requirements, **GENETIK** allows you to build a model of any type of problem, however complex it is. Answers from a **GENETIK** simulation or scheduling model allows one to re-assess what the real problem is or prompt fresh ideas for solving it.

GENETIK models are quick to build and requires no prior computer skills and is easy to learn. There is no limit on the size of the problem it can handle or the logical complexity that can be expressed. Models are built interactively; as ideas are recast, the model is adapted interactively; so the process is fast. **GENETIK** pictures have no size limitation and include features such as windows, panning, icons and zooming. Help facilities aid in the building up of the picture, data, logic and interaction units and **GENETIK'S** own syntax illuminates syntax and typing errors making the building of simple or complex models even easier.