SIGN — SIMULATION TO INVESTIGATE GENERALIZED NETWORKS

Isadore L. Goldhirsh
AUERBACH Corporation
Philadelphia, Pennsylvania

ABSTRACT

SIGN is a computer program package developed by AUERBACH Corporation for studying, by discrete simulation, large-scale complex generalized information and transportation networks. SIGN is driven by a network applications language called SIGNAL whose commands enable a user to automatically synthesize networks, compose networks from a data base of established components, compute optimal network routes, generate traffic distributions of mixed transport vehicle and cargo types, and pass traffic through the network in time-step sequence under programmed (SIGNAL) monitorship and control. SIGN will perform queueing, timing, loading, and other analyses associated with network performance, including transport vehicle scheduling, traffic pool placement, node placement, link utilization, and network cost effectiveness.

The representation and solution of problems by the methods of network analysis have intrigued analysts throughout history. Though considerable progress has been made in areas of theoretical network analysis, practical achievements have been frustrated by the enormity of computational difficulties associated with even relatively small networks of perhaps 20 nodes or more. Consequently, network analysis has tended to remain more in the province of mathematicians and theoreticians, and less in the province of the large class of analysts burdened with day-to-day practical network problems.

The advent of the computer brought about a surge of successful techniques and computer programs designed to study and find solutions for steady-state network problems. The attack on dynamic network problems, however, brought a recurrence of the specter of large computational burdens, beyond even modern computers. Many efforts to handle large dynamic networks by computer means failed for want of well-designed, efficient, and user-operable software.

As a conclusion to several years of investigation into the area of network methodology, AUERBACH conducted a study to determine the proper balance of features and design constraints required to develop a viable computer tool for the analysis and synthesis of networks.

It was determined that a generalized network simulation computer program capability was needed to represent the dynamic interactive information flows of various categories within a single integrated network of node and link facilities of various types. It was determined that the computer program should be structured as a system program package, self-contained, modular, expandable, and adaptable to network problem applications. The program package should be supported by a network data base to facilitate network modeling and a library of steady-state type application programs that could be freely called upon during the course of a simulation to perform selected operations of network analysis and synthesis. The library should include program units to modify networks, generate traffic, determine routing paths, invoke queueing and scheduling procedures, compute node placement and traffic pool placement, and evaluate during dynamic periods of simulation, node, link, and network performance, including traffic throughput, transmission delays, effectiveness, and cost benefit of network facilities and operations.

Most important, it was determined that the computer program package should be equipped with an applications language specifically oriented toward network modeling and network problems. As a result, SIGN (Simulation to Investigate Generalized Network) and its applications language (SIGNAL) were developed.

SIGN is a systems program package consisting of an Executive program module, a Simulator program module, a network data base and associated data management program module, and a program library. Minimum computer requirements are a medium-scale computer, 131,000 bytes of core memory, a high-speed mass storage device, and three magnetic tape units. Estimated operational speed for a minimal computer system, with an average instruction execution time of four microseconds, is five seconds of real time per simulation step time.

Networks in SIGN are modularly structured into six interactive, but distinct components: Nodes, Lines, Links, Routes, Cargo Traffic, and Vechicle Traffic. Nodes are the terminals, depots, processors, exchangers, and decision centers of traffic flow in a network. Lines are the physical connections between nodes, such as roads, wires, air ways. Links are the logical traffic paths between nodes; any link may consist of one or more lines of the same class. Routes are the possible network paths admissible to traffic. Cargo traffic is the representation of cargo loading and cargo destinations at each node of a network. In general, cargo may originate at any node and be destined to any other node in a network. Vehicle traffic is the representation of transport vehicle pools at each node of a network. Vehicles, of course, are associated with cargo traffic for their transport through a network. Vehicles, however, can also transport other vehicles so that, in general, many levels of cargo transportation are possible.

A network in SIGN may consist of sets of different classes of nodes (such as air terminals, harbors, train stations, communications centers) connected by links of different classes (such as air lanes, water lanes, rail tracks, radio links, telephone cable). A network will contain vehicle classes (such as planes, ships, trains, carrier frequencies) appropriate to at least one link class. Similarly, a network will contain cargo classes appropriate to at least one vehicle class (such as people, crates, messages). Traffic flow is constrained to class compatibilities so that, for example, a boat will not move through an air lane, and a radio link will not transport a crate of oranges. Therefore, a unit of cargo may traverse a network through a series of links of different classes in link-compatible transport vehicles. For example (see Figure 1), a unit of cargo may start by truck at a trucking terminal (T_1) and be transported via road to a water pier, continue its journey via boat to another water pier, and take up the last leg of its journey by truck, via road, to its destination (T_2).

SIGN simulates a dynamic network by moving traffic at regular time-step intervals in discrete bundles across links, from node to node, to final destinations. The traffic movement is constrained by routes, schedules, service rules, traffic congestion, node and link capacities, cargo demands, vehicle capacities, availability, and speeds.

Since SIGN treats generalized objects, cargo may in fact not be cargo in the tangible sense but an item of information. Transport vehicles need not be tangible vehicles, but may be necessary logical conditions associated with information flow or transmission. Nodes may be ideas or data pools. Links may be logical relationships between nodes.

SIGN is supported by a network data base and a library of program units which can be called upon to synthesize network models, generate traffic distribution loads, establish routing plans, determine traffic service rules, and simulate general operating and environmental conditions.

The SIGN network data base consists of files of network node, line, link, route cargo traffic, and vehicle traffic components. Full networks, or selected network components, may be entered into the data base manually through card input data, or be generated automatically by means of network synthesis routines of the program library. During a simulation active networks and network components may also be logged into the data base as distinct entries. To construct a network from the data base for a simulation exercise, a user will specify six basic network components. Since a network may be composed from the data base by selection of any mutually compatible set of the six basic network component types, a large number of networks may be derived from even a relatively small network data base, because of the many combinatorial possibilities. The network data base, therefore, is a principal resource for network simulation. In SIGN, the network data base is adaptable to users' needs, and may grow or contract with usage as full networks or individual network components are entered and deleted.

The SIGN system program is capable of handling a network as a group of connected subnetworks, or conversely, can connect a group of individual networks (subnetworks) into a single integrated multi-network. Each subnetwork within an integrated multi-network may operate at a different simulation time rate chosen to best suit the needs of the subnetwork. This capability permits subnetworks to be studied separately for a time, and later, as part of a whole network. This capability will also permit greater economy in the operation of the SIGN system program package by making the most efficient use of computer storage and operation time.

The design and operation of a simulation in SIGN is determined by a user's program written in SIGNAL. The structure and vocabulary of SIGNAL is simple and requires no special knowledge of computers or computer programming. The design of SIGNAL grammar is based on three classes of statements: Network Composition Statements, Network (Dynamic) Operation and Simulation Statements, and Control Statements. The first group of statements provides the means for establishing a desired network model and environment. The second group calls upon operations existing in the SIGN program library to adjust or otherwise affect the network situation during simulation. The third group includes statements which enable a set of SIGNAL statements to perform as a programmed procedure; it includes branching and conditional branching statements.

The conditional branch statements have been devised to enable a simulation to be driven in desired directions during the course of a network simulation. For example, conditional branch statements may specify a threshold level for traffic backlog at a given node in the network. This may be utilized by a SIGNAL program as a condition in which alternate routing operations are to be invoked (to relieve the node backlog). The statement may more simply indicate the desire to have a network status summary report at that time.

Sample Program

Table I illustrates the actual card format for a simple SIGNAL program of 11 statements which can set up and execute a substantial network simulation exercise.

Statements 1 and 11 are opening and closing statements and do little except formally identify and log a network simulation run.

Statement 2 calls for a network previously established in the network data base file, or composes a new network from six selected compatible network components maintained in the network data base.

Statement 3 modifies some of the elements of the composed network with input data associated with the statement. Links and nodes may be added, deleted, or their performance parameters modified for the exercise.

Statement 4 files the network into the data base for future use, either in the current run or subsequent simulations.

Statement 5 specifies an operation that is to be executed, dynamically, during the simulation run. In this example, the operation is a traffic generator which, according to the algorithm identified in the statement, enters traffic units into the network at prescribed nodes. The algorithm is executed at time intervals whose duration is also specified in the statement.

Statement 6 is similar to Statement 5 except that the operation is a failure generator which induces failures into the elements of a network over a specified period of the simulation, again in accordance with the algorithm identified in the statement.

Statement 7 is a display statement and generates a report which may be a summary status and analysis of the network at the time the display is to be executed. The times of display execution are specified within the statement.

Statement 8 is a conditional statement which enables the SIGNAL program to alter its statement sequence on the basis of the status of a specified node or link performance level or network condition. If the condition occurs during the course of a simulation run, the simulation will be interrupted, discontinued from its current course, and redirected on a new course given by the new stream of SIGNAL statements. By this means, it is possible to proceed automatically through a simulation toward optimal solutions without manual intervention.

Statement 9 is a simple command to start or reactivate a simulation run. At this point, the translation of SIGNAL statements stops and the network simulation begins. When the duration of the simulation period as specified in the simulation statement has transpired, translation of SIGNAL program statements is resumed.

Statement 10 is a control statement which allows the SIGNAL program to move out of sequence, as shown, in the manner of ordinary computer program languages. It is translated after the simulation period specified in Statement 9 has transpired.

Statement 11 is a jump to the End Statement, and completes the run.

SIGN is designed to provide a computational environment for the development of network algorithms appropriate to the solution of network problems. New network algorithms, based on practical constraints and realistic criteria, otherwise difficult to use, are computationally feasible when operating in SIGN. Such operations as compute optimal network routines, balance traffic and balance node capacities, when developed according to standards imposed by SIGNAL can be employed in conditional combinations and sequences, in the manner of a calculus, to investigate the behavior of dynamic network conditions and to fashion optimality stratagems for network situations.

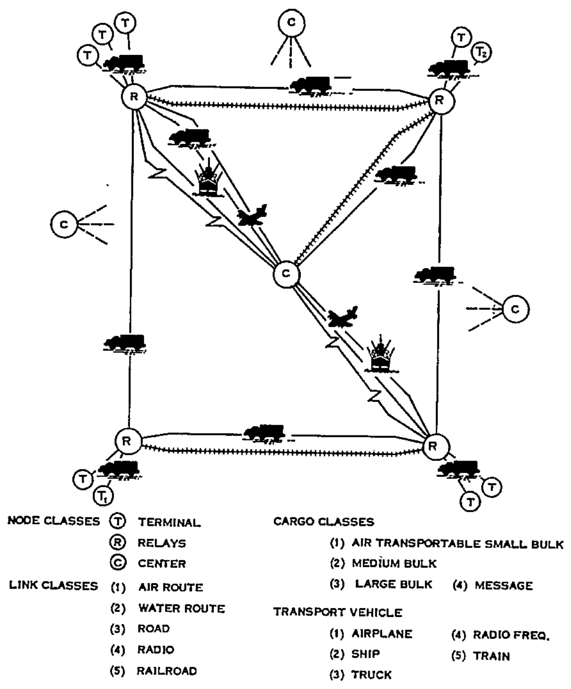

Figure 1. Generalized Network Model

TABLE 1. CARD FORMATS FOR SAMPLE SIGNAL PROGRAM

Card No.	Field I	Field II	Field III	Field IV	Field V	Field VI	Field VII
1	Start Signal	Job ID	User ID	Date of Run	Simulation start Time	Spec. equipment Requirements	Run Time Estimate
2	Compose Network from Components in Data Base	Node Component ID	Link Component ID	Route Component ID	Line Component ID	Traffic Component ID	-
3	Enter Change Cards (to modify parameter of network component)	Component Type	Network ID	Number of Data Cards	-	-	-
4	File Network	Network ID	-	-	-	-	-
5	Operation	Traffic Generator ID	Δt_o, Periodicity of Operation	Duration Time	-	-	Network ID
6	Operation	Failure Generator ID	Δt_o, Periodicity of Operation	Duration Time	-	-	Network ID
7	Display	Format ID	Δt_d, Periodicity of Display	Duration Time	-	-	Network ID
8	Conditional Jump	-	SIGNAL Statement Address	Test Condition Specification	Test Parameter	-	-
9	Simulate GO	-	Δt Simulation Time Step	Duration Time	-	-	-
10	Jump	-	SIGNAL Statement Address	-	-	-	-
⋮		-	-	-	-	-	-
11	End SIGNAL	-	-	-	-	-	-