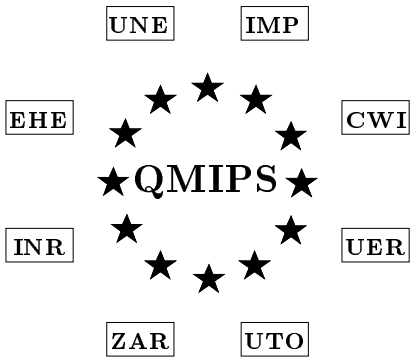

Quantitative
Modeling
In
Parallel
Systems

G-networks: A survey of results, applications and solutions

Sophie Chabridon (1), Erol Gelenbe (2), Marisela Hernández (3)
and Ali Labed (1)

(1) EHEI
Université René Descartes
45 rue des Saints-Pères
75270 PARIS CEDEX 06, France

(2) Department of Electrical Engineering
Duke University
Durham, N.C. 27708-0291

(3) LAMIFA
Université d'Amiens
80039 Amiens, France

D W2.T1.6-8.v1

19 September 1995

©EHE

Distribution Level: All partners

Approved by: E. Gelenbe

The QMIPS Project: The QMIPS project is a collaborative research project supported by the CEC as ESPRIT-BRA project no 7269. It is being carried out by the following organisations: CWI (Amsterdam), EHEI (University of Paris V), Imperial College (London), INRIA (Sophia-Antipolis), University of Erlangen, University of Newcastle, University of Torino and University of Zaragoza.

Abstract In this paper, we first present a brief survey of the main theoretical results providing product forms for networks of queues with positive and negative customers and with signals (G-networks). Then we present a graphical tool for solving a G-network model in steady state, i.e. for finding the steady-state probabilities of the number of positive customers in a G-network. The user will draw a G-network on the screen and input the parameters of each queue (mean service time, arrival rates, routing probabilities, etc.). Then the solver will provide the user with the solution of the system of non-linear traffic equations, and the stationary distribution of queue length, and performance measures such as sojourn times, mean number of customers in the system. Finally we show how these theoretical results and the tool we describe can be used to obtain an analytical solution to a problem which until now has resisted to such a treatment: the performance evaluation of receiver initiated load balancing algorithms in distributed systems.

1 Introduction

A new class of queueing networks has been recently introduced by Gelenbe [17, 21, 22] which unify different stochastic networks such as queueing networks and neural nets [13, 14, 15, 18, 20]. The G-network model was initially motivated by the analogy with neural networks [15]. The theory of the random neural network is developed by Gelenbe [14, 15, 19, 23]. G-networks have been widely used as neural networks, in particular with applications to the traveling salesman problem [30], minimum graph covering [25], graph partitioning [1], load balancing [2] and task assignment in a distributed system subject to failures [40]. Other applications concern problems of supervised learning of images [42], image and pattern recognition [44], texture generation [4, 3] and associative memory [28]. G-networks have been applied to evaluate the performance of unreliable flow systems [41] and to model virus behavior in a computer network [39]. In [24], it has been proved that the solution of a G-network provides a local minimum to a quadratic cost function.

When viewed as queueing networks, these new models called G-networks have essentially two types of customers: positive and negative. Positive customers have the same behavior as ordinary queueing network customers. If a positive customer joins a queue it waits until it receives service or it can be destroyed by a negative customer arriving to the queue. A negative customer joining a *non-empty* queue has the power to move or destroy a positive customer, or it will vanish immediately if the queue is empty. Negative customers will not be stored in a queue and will disappear as soon as they have accomplished their task. In the literature these negative customers are referred to as signals if their role is to displace a normal customer from one queue to another. They are referred to as negative customers if their role is to destroy one normal customer or a batch [22] of normal customers at some queue. Negative customers do not receive service and their actions are supposed to be taken instantaneously, that is, they do not consume any time. A positive customer leaving a queue after service can join another queue remaining positive or it can become a negative customer according to some probability. Although these networks have product form, their solution is quite different from that of usual BCMP models [5]. Specifically, the traffic equations for these networks, which allow one to calculate the rate of arrival of all types of customers to each queue, are non-linear. Furthermore, the probability intensity at each queue will have in its denominator the arrival rate of negative customers to that queue, while its denominator will contain the arrival rates of positive and negative customers. Even though G-networks have appeared recently, they have already motivated a substantial amount of research. Product form solutions have been characterized in [6, 7, 32, 33, 34, 36]. In [21], an extension of the previous work is presented where a different behavior of negative customers is considered. That is, the effect of a negative customer, now considered as a signal, arriving to a non-empty queue is to move a positive customer to another queue instead of just destroying it. This is called triggered customer movement. G-networks with signals also have product form solution [21]. In [22] the model is extended to networks where negative customers destroy batches of positive customers. Stability conditions are provided in [16].

Gelenbe, Glynn and Sigman [27] have studied several single server policies related to the arrival of negative customers to a non-empty queue, like removal of the customer in service and removal of the customer in the tail. Several extensions to this model have then

been considered. In [9, 10], an extension of G-networks has been discussed where positive customers could have different classes. Henderson et al. have proposed several extensions allowing state-dependent rate and batch transitions [35, 36, 37, 38]. [11] considers multiple class G-networks with jumps back to zero and [12] proposes G-networks with triggered batch state-dependent movement.

This paper begins with a survey of theoretical results on G-networks. Then we describe a simple software tool which allows the user to draw a G-network on the screen and to input the parameters of each queue (mean service time, external arrival rates, routing probabilities, etc.) for positive and negative customers. Then the solver will provide the user with the solution of the system of non-linear traffic equations, and of the stationary distribution of queue length, and performance measures such as sojourn times, mean number of customers in the system, etc. Finally, we will show how this class of models can be applied to one important problem in system performance modeling: the evaluation of the performance of a distributed system under the effect of receiver initiated load balancing.

In the rest of this paper, Sections 2 to 4 are devoted to the G-networks formalism for the cases mentioned above: positive and negative customers, signals, batch removals. Section 5 describes the tool we have designed for solving G-networks. In section 6 we present the numerical solution algorithm which is used and discuss some numerical aspects involved on it and its convergence. In section 7 an example of the use of G-networks for load balancing in distributed systems is presented. This example addresses and provides a first solution to one of the classical unsolved problems in system performance modeling: the analysis of systems with explicit on-line controls.

2 G-networks with positive and negative customers

In the simplest G-network model, customers are either *negative* or *positive*. Positive customers behave like ordinary queueing network customers. If a positive customer joins a queue it could wait until it receives service or it could be destroyed by a negative customer arriving to the queue. An arriving negative customer joining the queue instantaneously destroys a positive customer, if there is any in the queue, or it would vanish immediately if the queue is empty. Negative customers do not receive service and their actions are supposed to be taken instantaneously, that is, they do not consume any time. External positive or negative customer arrivals to queue i constitute independent Poisson processes with rate Λ_i for positive customers and rate λ_i for negative customers. Positive customers have iid exponential service distribution times with rate r_i at queue i. Positive customers leaving a queue after the completion of their service may join another queue either as a negative or as a positive customer. That is, a positive customer could become negative after its service is completed. The movement of customers between queues is represented by a Markov chain. A positive customer leaving queue j (after finishing service) joins queue i as a positive customer with probability $P^+_{j,i}$, or as a negative customer with probability $P^-_{j,i}$. It may leave the network with probability d_i. Let $P_{i,j} = P^+_{i,j} + P^-_{i,j}$; it represents the transition probability of a Markov chain modeling the movement of customers between queues. Therefore, we have for n such queues the following relation:

$$\Sigma_{j=1}^{n} P_{i,j}^{+} + \Sigma_{j=1}^{n} P_{i,j}^{-} + d_i = 1 \qquad 1 \leq i \leq n$$

G-networks have a non standard product form solution for the distribution of the queue length ($P(\vec{x})$) which is given in the following theorem.

Theorem 1 *[14, 17] Consider a G-network with N queues or nodes and positive and negative customers. If the system of non-linear equations*

$$q_i = \frac{\Lambda_i + \sum_{j=1}^{n} r_j q_j P_{j,i}^{+}}{r_i + \lambda_i + \sum_{j=1}^{n} r_j q_j P_{j,i}^{-}} \tag{1}$$

has a positive solution, such that for any queue i, $q_i < 1$ holds, then the stationary distribution $P(\vec{x})$ for the queue length at each station i of the network exists and has the following product form solution:

$$P(\vec{x}) = \prod_{i=1}^{n} p_i(x_i = k_i) \quad \textit{where} \quad p_i(x_i = k_i) = (1 - q_i)(q_i)^{k_i}$$

This model has been extended in [9, 10], to networks with multiple classes of positive customers and just one class of negative customers and three types of service policies (Processor Sharing, FIFO, LIFO). Note that the only customers which sojourn, i.e. actually spend any length of time, in G-networks are positive ones because negative customers have the strange property of appearing and disappearing instantaneously even though during this infinitesimal time they do have an important effect in the network. It is easy however to represent situations where negative customers would actually spend time traveling through a network, queueing up at some servers, etc.: during these activities they would be represented as normal positive customers and only change nature when they reach a queue where their "negative" behavior is effectively carried out. Thus the measures considered (sojourn time, number of customers, etc.) correspond to positive customers. Positive customer departures take place when:

- a service is completed,
- a negative customer arrives to a non empty line and destroys a positive customer.

Thus the mean sojourn time W of positive customers in the system is given by Little's formula (where q_i is given by (1)):

$$W = \frac{1}{\lambda_i^{+}} N_i \qquad N_i = \Sigma_{i=1}^{n} \frac{q_i}{1 - q_i}$$

3 G-networks with signals

In [21] an important extension of the previous model is shown to have a product form solution. This is the extension which allows us to model important effects such as traffic re-routing in computer networks or load balancing in distributed systems. Here, an arriving

negative customer which is now denoted a signal, may either destroy a customer, or it may move a positive customer to another queue instead of destroying it. These movements are Markovian, with transition matrix is denoted as $Q = (Q_{i,j})$ where $Q_{i,j}$ is the probability that a negative customer arriving to queue i displaces a positive customer residing in that queue towards queue j. The following theorem establishes the solution of their distribution and is proved in [21].

Theorem 2 *Consider a G-network with signals. If the system of non-linear equations*

$$q_i = \frac{\Lambda_i + \sum_{j=1}^{n} r_j q_j P^+_{j,i} + \sum_{m=1}^{n} (\lambda_m + \sum_{j=1}^{n} r_j q_j P^-_{j,m}) Q_{m,i} q_m}{r_i + \lambda_i + \sum_{j=1}^{n} r_j q_j P^-_{j,i}} \tag{2}$$

has a positive solution such that for any queue i, $q_i < 1$ holds, then the stationary distribution $P(\vec{x})$ of the network exists and has the standard product form of Theorem 1.1 in which the value of q_i given here is substituted.

4 G-networks with signals and batch removals

The preceding work has been generalized in [22] to allow the destruction of a batch of positive customers, in addition to the motions of customers with the use of signals. In this extension, a signal arriving to an empty queue will have no effect and will just disappear. However if a signal arrives to queue i which is non-empty, then one of the following two events occur:

- The arriving signal triggers the instantaneous passage of a customer from queue i to some other queue j with probability $Q_{i,j}$.
- With probability $D(i) = 1 - \Sigma_j Q_{i,j}$, it forces a batch of customers of random size to leave the network. Let the length of queue i be k_i at the instant of arrival of the trigger; if $k_i \geq B_i$, then its length is reduced by B_i (batch size) and if $k_i < B_i$, the queue length becomes zero. The batch removal size distribution at queue i is given by: $P[B_i = m] = \pi_{im}, m \geq 1$.

Thus, a signal acts as an external trigger that instantaneously moves a customer from one queue to another or a batch to the outside world.

Theorem 3 *Define the function $f_i(x) = \frac{1 - \Sigma_{m=1}^{\infty} \pi_{i,m} x^m}{1-x}$. If the system of non-linear equations:*

$$q_i = \frac{\Lambda_i + \sum_{j=1}^{n} r_j q_j P^+_{j,i} + \sum_{m=1}^{n} (\lambda_m + \sum_{j=1}^{n} r_j q_j P^-_{j,m}) Q_{m,i} q_m}{r_i + (\lambda_i + \sum_{j=1}^{n} r_j q_j P^-_{j,i}) D_i f_i(q_i)} \tag{3}$$

has a solution with $q_i < 1$, then the G-network with signals and batch removal has the standard product form solution in steady-state.

Clearly, if the batch is of size 1 and the matrix Q is a null matrix, we obtain the simplest G-network with positive and negative customers described in Section 2, while if the batch is of size 1 then we will simply have the model of Section 3. Note also that the destruction of

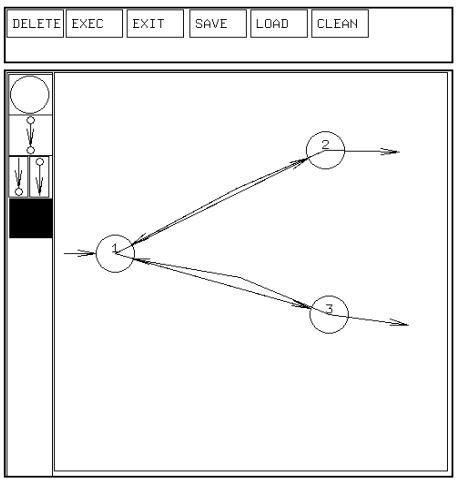

Figure 1: The solver graphical interface

all customers in the queue by the arrival of a negative customer is also very easy to achieve. Note that a batch of infinite size at queue i may simply be represented by:

$$\pi_{i,m} = 0 \quad \text{for } m < \infty, \quad \Sigma_{m=0}^{\infty} \pi_{i,m} = 1,$$

which will give:

$$f_i(x) = \frac{1}{1-x}$$

Thus to analyze this case it will suffice to introduce this particular form of $f_i(x)$ in the formulae for the q_i.

5 The solver tool

The solver tool for G-networks which we have implemented is composed of a user interface and of a numerical solver. To the best of our knowledge, this is the first tool developed for solving queueing network models which use positive and negative customers, although there exist already various tools for solving standard queueing networks: QNAP2 [43, 48], PAWS [45], RESQ [47], SIMAN V [46], etc. Let us first briefly describe its user interface.

The tool is written in C and runs on any Unix workstation provided with X-windows (the version X11R5 or a later version) and allows the user to draw on the screen the G-network that is to be solved. Once it is drawn, parameters must be entered for each arc and each queue, including the routing probabilities for each type of customer and the service rates. It is called with the command: *solver*. No arguments are necessary. A single window (see Figure 1) will appear on the screen. This window has three areas:

- Main Menu (at the top)
- Icons (on the left)

- The window itself which is a drawing area

We now describe how to use each of these areas. Clicking a mouse button simply consists in pressing the button and immediately releasing it. To drag with a mouse button, move the cursor over an object, then press a mouse button and hold it down. Move the mouse by continuing to hold down the mouse button and complete the drag by releasing the mouse button.

The Main Menu is localized at the top of the window. It has the following options which are selected by clicking the left button of the mouse. There are seven possible operations. Six operations only are displayed in the main menu. The option CREATION is the default and corresponds to the case where no operation is actually in inverse video.

- CREATION : This option allows the user to create a new G-network. This is the default option of the tool.
- LOAD : It is used to load on the screen a G-network which has been previously created. A small window appears on the screen to allow the user to give the name of the file to be loaded.
- SAVE : Once created, the G-network and its corresponding parameters are saved in a file. A dialog box is displayed to ask for a file name.
- DELETE : In the process of the construction of a G-network, the user may want to delete an object. Deleting a queue will automatically delete the arrows connecting it with other queues.
- CLEAN : This option allows the user to clear what has already been drawn and to start the drawing of a new G-network.
- EXEC : This option will run the program to solve the network. The corresponding results or errors are reported in a file called "results".
- EXIT : Its purpose is to exit the solver environment.

The Icons area is placed vertically on the left of the window; it indicates the current selected icon. There are currently four possible icons :

- node : corresponds to a service station and its associated queue; it is represented by a circle.
- connect line : corresponds to a link between two nodes; it is represented by an arrow with one circle at both end points.
- entry line : corresponds to an external arrival to a node; it is represented by an arrow with a circle at its end point.
- out line : corresponds to an external departure from a node; it is represented by an arrow with a circle at its beginning point.

The Drawing Area contains the result of mouse operations as a function of the currently selected operation in the main menu.

- CREATE When the creation mode is active, the user simply has to select an icon with the left mouse button and then click it in the drawing area at the position where the user wants the object to be drawn. To draw an arc from one node to another, drag from one node to another with the left mouse button.

- DELETE When the DELETE operation is selected in the main menu, clicking with the left mouse button on an object in the drawing area will automatically delete it.

When the middle mouse button is clicked on an object, a dialog box is displayed. It contains edit fields for the parameters corresponding to the type of the clicked object:

- Parameters of node i:
  Mu : Service rate of node i (r_i in the formulas).

- Parameters of arcs reaching node i:
  LAMBDA : Arrival rate of positive external customers (Λ_i in the formulas),
  lambda : Arrival rate of negative external customers (λ_i in the formulas).

- Parameters of arcs leaving node i:
  d : Probability for a customer to leave node i ($d(i)$ in the formulas).

- Parameters of an arc connecting nodes i and j:
  P^+ : corresponds to $P^+_{i,j}$ in the formulas,
  P^- : corresponds to $P^-_{i,j}$ in the formulas,
  Q : corresponds to $Q_{i,j}$ in the formulas.

Since this is the first version of this tool, additional features will be implemented in the future. Batch removals cannot be specified so that all the destructions are equivalent to the destruction of a batch of size 1. The size of the G-network is limited by the size of the window screen since there is no scrolling implemented. Thus the G-network can only contain as many queues as can been drawn on one screen. Drawing arrows is not very flexible; an arrow is formed by only one straight line. However, when there are two arrows between two queues in opposite directions, the second arrow is automatically represented by two lines.

Let us briefly discuss some numerical considerations. In [15, 21, 22] the existence of the solution of the nonlinear customer flow equations was established using Brouwer's fixed-point theorem. This is valid for stable and unstable systems, as long as the numerical procedure for computing the probabilities q_i makes sure that their value can never exceed 1. Once the q_i's are numerically obtained, all the other parameters of interest, including the joint queue length probabilities in steady-state, follow directly. Heuristic methods for finding a solution for G-networks have also been proposed in [8]. The following algorithm is used by the solver to compute a solution for the system of non-linear equations for the q_i's of the G-network:

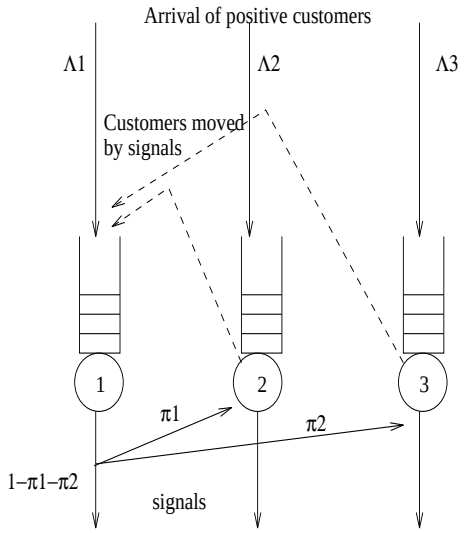

Figure 2: System with load balancing

1. Initialize the error ε which is the iteration's stopping criterion based on the square of the difference of two successive computations of q_i:

$$E^k = \sum_{i=1}^{n} (q_i^k - {q_i}^{k-1})^2 \tag{4}$$

2. Set $k = 0$. For $i = 1, ..., n$, initialize q_i^0 with random values in the interval $[0, 1]$ using a uniform distribution.

3. Compute the new value of q_i^k using the appropriate non-linear equation ((1), (2) or (3) according to which specific model is considered). If $q_i^k > 1$ set $q_i^k = 1$.

4. Compare the value of q_i^k at iteration k, to its value at the previous iteration q_i^{k-1} and go to Step 3 if $E^k = \sum_{i=1}^{n} (q_i^k - q_i^{k-1})^2 > \varepsilon$.

In [26] the convergence of this iteration to the steady-state probabilities q_i of the model is proved, for any set of initial probabilities q_i^0 in $[0, 1]$.

6 An example: Performance evaluation of receiver initiated load balancing

Dynamic load balancing occurs explicitly (by design) or implicitly (as a result of spontaneous system actions) in distributed systems. It has a profound effect on the performance of a distributed system for obvious reasons: a poorly balanced system may concentrate all the work at a few processors and leave other processing units idle, as a result of which system performance may be very bad. A well balanced system will fully exploit parallelism and therefore provide the best performance that the workload can expect. Static load balancing

may be carried out for a given workload on a given system prior to execution, while dynamic load balancing is carried out continuously during execution as a function of the observed load throughout the system. Because of its importance, there has been much research on the subject, and it is still a matter of ongoing investigation. Here we will not survey the abundant literature on load balancing but simply present a specific example using G-networks. In this example we consider receiver initiated load balancing, where any processing unit is allowed to request work from other units whenever it considers that it can assume more work. In order to simplify the discussion we consider a distributed system with three processors where the goal is to balance the processors load. The discussion and analysis presented here can be extended to a system containing an arbitrary number of processing units.

The system is depicted in Figure 2. The three processors are assumed to be connected by a perfect communication network which transfers jobs in zero time and in a perfectly reliable manner. Similarly, messages used to control the system are also assumed to transit instantaneously through the network. We wish to point out that it is quite easy to model the effect of an imperfect network with substantial transit times for messages and jobs using a very similar approach. Each processing unit receives a stream of jobs from the outside world. However in addition, it may request jobs from other processing units. Receiver initiated load balancing proceeds in the following manner. As soon as processing unit i finishes a job, it may request that a job be transferred to it from one of the other processing units, say j. If it does so, then j will transfer a job to i if it has one in its queue. This request is represented by a signal which leaves processing unit i to go to j as soon as i finishes work. This is modeled by a probability $\pi_{i,j}$ for a signal to be sent to processor j by i when at the end of a service epoch. Clearly, even if a processing unit is unaware of the actual load at other units, the fact that it will have a tendency to request work of others whenever it finishes work will have a tendency to move more work to those processors which are serving jobs at a higher rate, while slow or relatively inactive processors will become less active.

In this model no ordinary negative customers arrive or exist in the system, and therefore $\vec{\lambda} = \vec{0}$. We use the previous notation so that the external arrivals of jobs to processor i are represented by a Poisson process of rate Λ_i, and it serves jobs at rate r_i. In the numerical example given below we assume that the receiver initiated load balancing requests are carried out *only* by processor 1 so that $\pi_{1,j} = \pi$ for $j = 2, 3$. We consider that queues 2 and 3 are identical; they have the same arrival rate Λ, the same service rate r, and the same probability π of receiving a signal from 1. The transition matrices of the corresponding G-network model are then:

$$P^+ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad P^- = \begin{pmatrix} 0 & \pi & \pi \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad Q = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Using the G-network solver presented in Section 5, we computed the values of the loads q_i with the following parameters: $\Lambda_1 = 3.0$ and $\Lambda = 1.6$, $r_1 = 10.0$, $r = r_2$, r_3 varies in the range $[1, 10]$, while π varies in $[0.1, 0.5]$. Equal loads were obtained for values of r in the

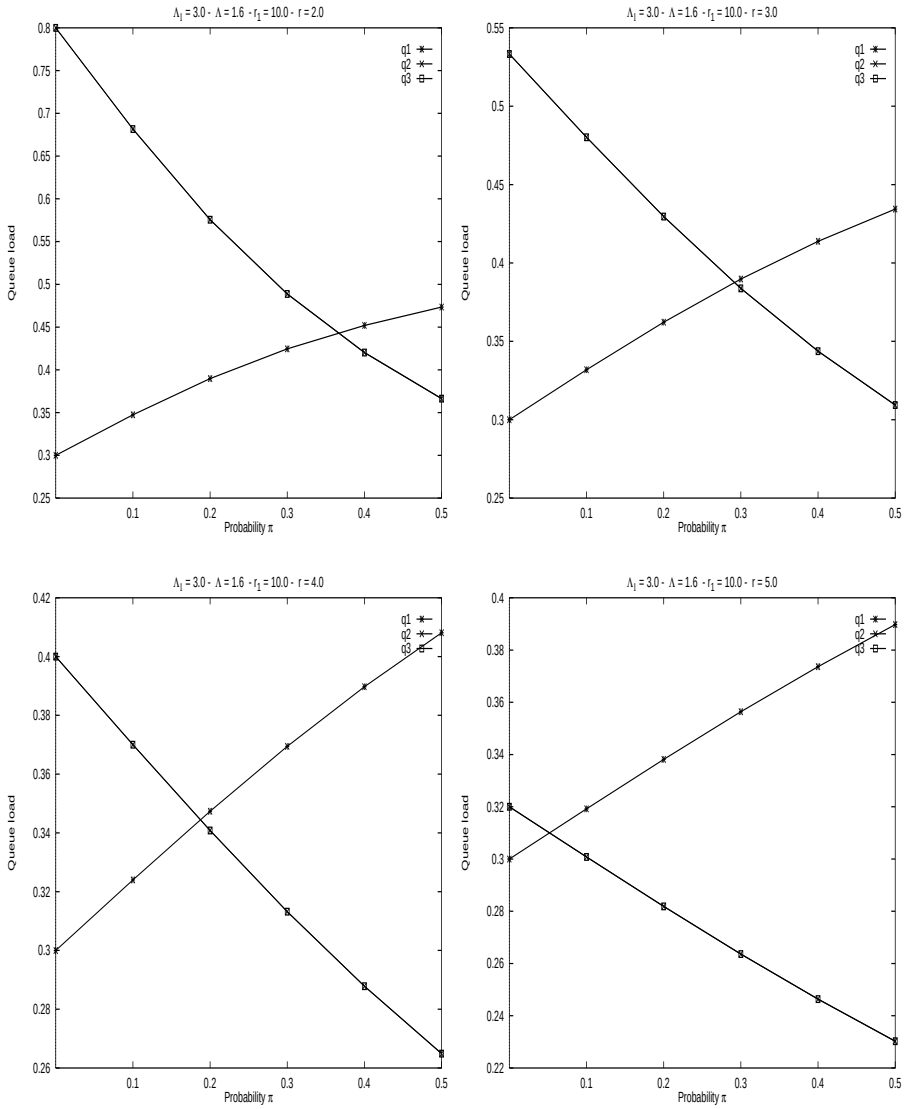

Figure 3: Cases with balanced loads at each queue

interval $[2, 5]$.

In Figure 3, we show the variation of the q_i's versus π for the four values of r. On Figure 4, we give the average sojourn time of jobs in the system as a function of r for the case with no load balancing, i.e. with $\pi = 0$, and for π in $[0.1, 0.5]$. As expected, when the service rate of processor 1 is much higher than that of 2 and 3, load balancing is very useful and results in much smaller sojourn times than without load balancing. However, beyond the intuitive relationship betwen the different parameter values and the results observed, this analysis allows us to quantify the conditions under which load balancing would make sense, and the order of magnitude of the performance improvement that can be expected.

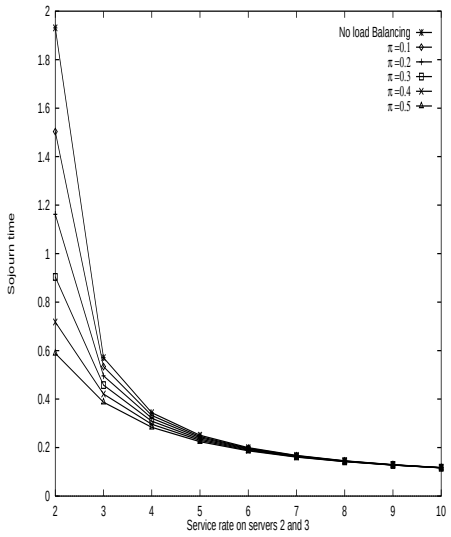

Figure 4: Average sojourn time in the system

7 Conclusions

The purpose of this paper has been to summarize the main theoretical results concerning G-networks which have been obtained in the context of the QMIPS project, to describe a convenient and portable software tool which we have developed for specifying and then solving the model in steady state, and to describe its use via a simple example.

Due to the fact that this formalism can be used in many applications, a tool such as the one we have described facilitates the task of modeling practical applications. We expect that a variety of new applications will be addressed using G-networks, and that these will lead to further generalizations and extensions.

References

[1] Aguilar J., *Comparison between the Random Neural Network Model and other Optimization Combinatorial Methods for the Large Acyclic Graph Partitioning Problem*, Proc. 7th Int. Symp. on Computer and Information Sciences (ISCIS), Antalya, Turkey (1992).

[2] Aguilar J., *L'allocation de tâches, l'équilibrage de charge et l'optimisation combinatoire*, PhD Thesis, University of Paris V, France (1995).

[3] Atalay V., *Réseaux de neurones aléatoires et textures d'images*, PhD Thesis, University of Paris V, France (Nov. 1993).

[4] Atalay V., Gelenbe E., Yalabik N., *Image Texture Generation with the Random Neural Network Model*, Int. Conf. on Artificial Neural Networks (ICANN-91), Helsinki (Kohonen T. ed.), Elsevier, (1991).

[5] Baskett F., Chandy K.M., R.R. Muntz, Palacios F.G., *Open, Closed and Mixed Networks of Queues with Different Classes of Customers*, Journal of ACM, 22(2):248-260 (Apr. 1975).

[6] Boucherie R., *Product-form in queueing networks*, PhD thesis, Vrije Universiteit, North-Holland (May 1992).

[7] Boucherie R., Van Dijk N., *Local Balance in queueing networks with negative customers*, Research memorandum 1992-1, Free University of Amsterdam, North-Holland (1992).

[8] Fourneau J.M., *Computing the Steady-state Distribution of Networks with Positive and Negative Customers*, LRI Report, $13^{th} XS$ IMACS World Congress on Computation and Applied Mathematics, Dublin, Ireland (1991).

[9] Fourneau J-M., Gelenbe E., *G-Networks with Multiple Classes of Signals*, Proceedings ORSA Computer Science Technical Committee Conference, Williamsburgh, VA, USA, Jan. 8-10, Pergamon Press (1992).

[10] Fourneau J.M, Gelenbe E., *Multiclass G-Networks*, ORSA Conference: Computer Science and operation Research : new developments in their interface, Williamsburg, USA (Jan. 1992).

[11] Fourneau J-M., Kloul L., Quessette F. *Multiple Class G-Networks with Jumps back to Zero*, Proc. of the 3d Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'95), Durham, NC, USA, pp. 28-32 (Jan. 1995).

[12] Fourneau J-M., Verchère D. *G-Networks with Triggered Batch State Dependent Movement*, Proc. of the 3d Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'95), Durham, NC, USA, pp. 33-37 (Jan. 1995).

[13] Gelenbe E., *Random Neural Networks with Negative and Positive Signals and Product Form Solution*, Neural Computation, 1(4):502-510 (1989).

[14] Gelenbe E., *Réseaux Stochastiques avec Clients Négatifs et Positifs et Réseaux Neuronaux*, Comptes-Rendus Académie des Sciences, 309, Série II, Paris, France, pp.979-982 (1989).

[15] Gelenbe E., *Réseaux Neuronaux Aléatoires Stables*, Comptes-Rendus Académie des Sciences, Paris, France, pp.310-313 (1990).

[16] Gelenbe E., *Stability of the Random Neural Network Model*, Neural Computation, 2(2):239-247, (1990).

[17] Gelenbe E., *Product Form Queueing Networks with Positive and negative customers*, Journal of Applied Probability, 28:656-663 (1991).

[18] Gelenbe E., *Theory of the Random Neural Network Model*, Neural Networks: Advances and Applications (E. Gelenbe ed.), North-Holland (1991).

[19] Gelenbe E., *G-Nets and Learning Recurrent Random Networks*, Proc. Int. Conf. on Artificial Neural Networks, Brighton, England (1992).

[20] Gelenbe E., *G-Networks: A Unifying Model for Neural Nets and Queueing Networks*, Proc. of the 1st Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'93), San Diego, CA, USA, Simulation Series, 25(1), pp. 3-8 (1993).

[21] Gelenbe E., *G-Networks with Triggered Customer Movement*, Journal of Applied Probability, Vol. 30, pp. 742-748 (1993).

[22] Gelenbe E., *G-Networks with Signals and Batch Removals*, Probability in Engineering and Informational Sciences, Cambridge University Press, England, Vol. 7, pp. 335-342 (1993).

[23] Gelenbe E., *Learning in the Recurrent Random Neural Network*, Neural Computation, 5(5):154-164, (1993).

[24] Gelenbe E., *G-networks and Minimum Cost Functions*, Proc. of the 3d Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'95), pp. 135-141, Durham, NC, USA (1995).

[25] Gelenbe E., Batty F., *Application of the Random Neural Network Model to the Minimum Graph Covering*, Neural Networks: Advances and Applications 2 (E. Gelenbe ed.) North-Holland, (1992).

[26] Gelenbe, E., Jin W. *Convergence of the numerical iteration for the steady-state distribution of G-networks*, submitted for publication, May 1995.

[27] Gelenbe E., Glynn P., Sigman K., *Queues with Negative Arrivals*, Journal of Applied Probability, Vol. 28, pp. 245-250 (1991).

[28] Gelenbe E., Stafylopatis A., Likas A., *Associative Memory Operation of the Random Network Model*, Proc. Int. Conf. on Artificial Neural Networks, Amsterdam, North-Holland (1991).

[29] Gelenbe E., Schassberger P., *Stability of Product Form G-Networks*, Probability in Engineering and Informational Sciences, Cambridge University Press, England, Vol. 6, pp. 271-276 (1992).

[30] Gelenbe E., Koubi V., Pekergin F., *Dynamical Random Neural Network Approach to the Traveling Salesman Problem*, Proc. Conf. on Systems, Man and Cybernetics, Illinois, USA (1993).

[31] Gelenbe, E., Boxma, O.W., Fourneau, J.M., Harrison, P.G., Hernandez, M., Pitel, E. *G-networks*, Hot Topics Session, ACM-SIGMETRICS/ IFIP WG 7.3 Symposium on System Performance Evaluation, Ottawa, May 1995.

[32] Harrison P.G., Pitel E., *Sojourn Times in Single Server Queues with Negative Customers*, Journal of Applied Probability, Vol. 30, pp. 943-963 (1993).

[33] Harrison P.G., Pitel E., *The M/G/1 Queue with Negative Customers*, Proceedings of the 4th QMIPS (Queueing Modeling in Parallel Systems) Workshop, London, England, pp. 185-213 (Apr. 1994).

[34] Harrison P.G., Pitel E., *Response Time Distributions in Tandem G-networks*, to appear in Journal of Applied Probability (Mar. 1995).

[35] Henderson W., *Queueing networks with negative customers and negative queue lengths*, Journal of Applied Probability, Vol. 30, pp. 931-942 (1993).

[36] Henderson W., Northcote B.S., Taylor P.G., *Modeling using queueing networks with signals*, ITC 13 (1991).

[37] Henderson W., Northcote B.S., Taylor P.G., *Geometric equilibrium distributions for queues with interactive batch departures*, Annals of Operations Research, Vol. 48, pp. 493-511 (1994).

[38] Henderson W., Northcote B.S., Taylor P.G., *Networks of customer queues and resource queues*, ITC 14 (1994).

[39] Hernández M., *Virus Transmission in a Computer Network*, Research Report LAMIFA, Université d'Amiens, France, to appear (1995).

[40] Hernández M., Aguilar J., *A Simulator for Task Assignment in a Distributed System Subject to Failures*, Proc. of the 4th QMIPS (Queueing Modeling in Parallel Systems) Workshop, London, England, pp. 89-106 (Apr. 1994).

[41] Hernández M., Fourneau J-M., *Modeling Defective Parts in a Flow System using G-Networks*, Proc. Workshop on Performability Modeling of Computer and Communication Systems, Le Mont St-Michel, France (Jun. 1993).

[42] Hubert C., *Supervised Learning and Retrieval of Simple Images with the Random Neural Network*, Proc. 7th Int. Symp. on Computer and Information Sciences (ISCIS), Antalya, Turkey (1992).

[43] Merle D., Potier D., Véran M., *A Tool for Computer Systems Performance Analysis*, Performance of Computer Installations, Ed. Ferrari D., Amsterdam, North-Holland, pp. 195-213 (1978).

[44] Mokhtari M., *Réseau Neuronal Aléatoire: Applications à l'apprentissage et à la reconnaissance d'images*, PhD Thesis, Univ. Paris V, France (Jan. 1994).

[45] *PAWS: A User Guide*, Information Research Ass., Austin, TX, USA (1983).

[46] Pegden C.D., *Introduction to Siman*, Systems Modeling Corp., State College, PA, USA (1984).

[47] Sauer C.H., MacNair E.A., *Simulation of Computer Communication Systems*, Englewood Cliffs, Prentice-Hall, USA (1983).

[48] Véran M., Potier D., *QNAP2: A Portable Environment for Queueing Systems Modeling*, Int. Conf. on Modeling Techniques and Tools for Performance Analysis, North-Holland, pp. 25-63 (1984).