--
--

This page is intentionally left blank.

COMPUTATIONAL PROPERTIES OF GENERALIZED HOPFIELD NETWORKS APPLIED TO NONLINEAR OPTIMIZATION

Athanasios G. Tsirukis
and
Gintaras V. Reklaitis

School of Chemical Engineering

Manoel F. Tenorio

School of Electrical Engineering

Technical Report TREE 89-69

School of Electrical Engineering

Purdue University

ABSTRACT

A nonlinear neural framework, called the Generalized Hopfield Network, is proposed, which is able to solve in a parallel distributed manner systems of nonlinear equations. The method is applied to the general nonlinear optimization problem. We demonstrate GHNs implementing the three most important optimization algorithms, namely the Augmented Lagrangian, Generalized Reduced Gradient and Successive Quadratic Programming methods.

The study results in a dynamic view of the optimization problem and offers a straightforward model for the parallelization of the optimization computations, thus significantly extending the practical limits of problems that can be formulated as an optimization problem and which can gain from the introduction of nonlinearities in their structure (eg. pattern recognition, supervised learning, design of content-addressable memories).

1. Introduction

The ability of networks of highly interconnected nonlinear processors (neurons) to solve complicated optimization problems was demonstrated in a series of papers by Hopfield and Tank, (Hopfield, 1984), (Tank *et al.*, 1986). Problems that can be formulated and solved on such neural circuits include signal decoding, pattern recognition, linear programs, the Traveling Salesman Problem and other decision problems whose objective function can be conveniently expressed as a quadratic function of the system's independent variables. The dynamics of such networks, generated by the analog response, high interconnectivity and the existence of feedback connections, produce a path through the space of independent variables that tends to minimize the objective function value. Eventually, a stable steady-state configuration is reached, which corresponds to a local minimum of the objective function.

Since each optimization or, in general, nonlinear equation solving problem can be considered as a transition from an initial state to an optimal one, we will try to extend the original model, so that it can handle general nonlinear optimization problems. Specifically in this paper we will:

(i) propose a systematic procedure to transform a nonlinear optimization problem (both unconstrained and constrained) into a dynamic model

(ii) investigate the necessary structure of a network of simple nonlinear analog processors which are able to implement the dynamic model, and

(iii) propose a highly distributed computational model for solving nonlinear optimization problems in a series of parallel processors, able to implement all the existing important solution approaches (Cauchy's and Newton's method, Augmented Lagrangian, Generalized Reduced Gradient, Successive Quadratic Programming).

2. Literature Review

The bulk of the research effort on the application of neural networks in optimization has been concentrated on the solution of combinatorially complex decision problems. It was soon realized, (Bruck *et. al*, 1987), that the quality of the neural networks solution depends on the quality of the underlying algorithm and that combinatorial problems (eg. Traveling Salesman Problem) can not be solved with guaranteed quality, getting trapped in locally optimal solutions. Jeffrey and Rossner, (Jeffrey *et al.*, 1986), extended Hopfield's technique to the nonlinear unconstrained optimization problem, using Cauchy dynamics. Kennedy and Chua, (Kennedy *et al.*, 1988), presented an analog implementation of a network solving a nonlinear optimization problem. The underlying optimization algorithm is a simple transformation method, (Reklaitis *et al.*, 1983), which was proved relatively inefficient for large nonlinear optimization problems.

3. Background

3.1 Linear Hopfield Networks

The computation in a Hopfield network is done by a collection of highly interconnected simple nonlinear processors (amplifiers) also called neurons. Each processing element, i, is characterized by the activation level, u_i, which is a function of the cumulative input that the processor receives from the external environment and the other members of the network, through their weighted interconnections. The activation level of i is transmitted to the other processors, after passing through a filter that converts u_i to a 0 or 1 discrete value, V_i. The time behavior of the system is described by the following model:

$$C_i(\frac{du_i}{dt}) = \sum_j T_{ij} V_j - \frac{u_i}{R_i} + I_i \qquad (3.1.1)$$

$$u_i = s_i^{-1}(V_i) \qquad (3.1.2)$$

where, T_{ij} is the strength of the connection from processor j to i, I_i denotes the fixed external input, and the parameters R_i, C_i control the time behavior of the processor.

The function $V_i = s_i(u_i)$ describes the aforementioned filter. Ideally, it should be a hard threshold function. This is not realizable because of the discontinuity introduced at u = 0. A continuously differentiable s-shaped function (eg. $tanh(\lambda u)$, $1 / [1 + e^{-\lambda u}]$) is used to approximate the threshold behavior. In the limit $\lambda \rightarrow \infty$ the S - shaped function reproduces exactly the hard threshold function (Figure 1).

The right hand side of equation (3.1.2) describes the inputs to neuron j, which are linearly combined, after being weighted by a corresponding interconnection strength. We will refer to these architectures as linear Hopfield networks (LHN). Hopfield showed, (Hopfield, 1984), that the underlying Lyapunov function of the network, for symmetric interconnection strengths T_{ij} is given by:

$$\begin{aligned} E_0 &= E_1 + E_2 \\ E_1 &= -\frac{1}{2}\sum_i \sum_j T_{ij} V_1 V_j + \sum_i I_i V_i \qquad (3.1.3) \\ E_2 &= \sum_i (\frac{1}{R_i}) \int_0^{V_i} s_i^{-1}(V) dV \end{aligned}$$

The 'energy' function E_0 is composed of two terms. E_1 is a quadratic form of the activation levels. E_2 is produced by the S - shaped approximation to the filter equation. In the high gain limit, $\lambda \rightarrow \infty$, and for symmetric T matrix, E_2 becomes negligible and does not affect the topology of E_0, which becomes identical to E_1. When λ is finite, E_2 dominates because the function $s_i\,(V_i)$ becomes unbounded for extreme values of V_i. It is proven that if the system is assigned an initial value, it will relax to a steady state ($t \rightarrow \infty$) corresponding to a local minimum of the system's energy function E_0. For a symmetric T, the local minima of E_0 lie on the corners of the hypercube (feasible region) $-1 \leq V_i \leq 1$, where some V_i will be -1 and others +1. These observations turn the linear Hopfield network to a very useful discrete optimization tool.

3.2 The Nonlinear Optimization Problem

In the general nonlinear optimization problem we are looking for the values of the independent variables x_i, i = 1,2,...,N, which minimize a certain multivariable objective function so that certain conditions (constraints) are satisfied at the optimum. The problem can be expressed as:

$$
\begin{aligned}
&\textit{minimize} \quad f\,(x_1, x_2, ..., x_n) \\
&\textit{subject to} \\
&\qquad h_i\,(x_1, x_2, ..., x_n) = 0 \qquad\qquad i = 1,2,...,K, \quad K < N \\
&\qquad a_j \leq g_j\,(x_1, x_2, ..., x_n) \leq b_j \qquad j = 1,2,...,M \\
&\qquad x_k^L \leq x_k \leq x_k^U \qquad\qquad\qquad k = 1,2,...,N
\end{aligned}
\tag{3.2.1}
$$

where, f is the objective function, h_i are the equality constraints and g_j are the inequality constraints. x_k^L and x_k^U are respectively the lower and upper bounds of the independent variables.

The features of the nonlinear optimization problem require more expressive power that this offered by the LHN architecture:

(i) - The energy (objective) function f, to be minimized, can take any nonlinear form, as opposed to the E_0's quadratic form in (3.1.3).

(ii) - The feasible region, implicitly defined by equations h_i, g_j, can have any shape, as opposed to the original hypercube geometry ($-1 \leq V_i \leq 1$).

(iii) - The optimum can lie anywhere in the feasible region.

An extension to the original network structure is necessary. The processors must be allowed to interact in a general nonlinear fashion, contrary to the linear input structure in (3.1.1).

3.3 Necessary Optimality Conditions

In the following analysis, we will concentrate on the equality constrained optimization problem (with bold characters denoting vectors):

$$\textit{minimize} \quad f(\boldsymbol{x}) \tag{3.3.1}$$

$$\textit{subject to}$$

$$\boldsymbol{h}(\boldsymbol{x}) = 0$$

Each inequality constraint can take an equivalent equality form by introducing an additional *slack* variable, (Reklaitis *et al.*, 1983):

$$g(\boldsymbol{x}) \geq 0 \quad \rightarrow \quad g(\boldsymbol{x}) - x_{N+1} = 0\,, \quad x_{N+1} \geq 0$$

Assuming that functions f, h_i are differentiable, the necessary conditions for a point $\boldsymbol{x}^*$ to be the solution of problem (3.2.1) are:

$$\boldsymbol{\nabla} f - \boldsymbol{v}^T \boldsymbol{\nabla h} \geq \boldsymbol{0} \tag{3.3.2}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \tag{3.3.3}$$

Equations (3.3.2) and (3.3.3) are called *Lagrangian conditions.* The parameters $\boldsymbol{v}$ are the *Lagrangian multipliers.* It is customary to use a unified notation for both of the above equations by introducing the *Lagrangian function:*

$$L = f - \boldsymbol{v}^T \boldsymbol{h} \tag{3.3.4}$$

Then, conditions (3.3.2) and (3.3.3) are simply:

$$\boldsymbol{\nabla}_x L = \boldsymbol{0} \qquad (N \textit{ equations}) \tag{3.3.75}$$

$$\boldsymbol{\nabla}_v L = \boldsymbol{0} \qquad (K \textit{ equations}) \tag{3.3.6}$$

These are (N+K) equations with (N+K) unknowns, $\boldsymbol{x}$ and $\boldsymbol{v}$. In order to solve the problem (3.3.75) - (3.3.6) the nonlinear optimization algorithms rely on the following theorem:

Theorem 3.3.1: If $\boldsymbol{x}^*$ is the solution of the nonlinear optimization problem, then $(\boldsymbol{x}^*, \boldsymbol{v}^*)$ is a saddle-point of the Lagrangian function satisfying:

$$L(\boldsymbol{x}^*, \boldsymbol{v}) \leq L(\boldsymbol{x}^*, \boldsymbol{v}^{*)}) \leq L(\boldsymbol{x}, \boldsymbol{v}^{*)}) \qquad \text{for } \textit{all } \boldsymbol{x} \text{ and } \boldsymbol{v} \tag{3.3.7}$$

Therefore, in the neighborhood of the optimum $\boldsymbol{x}^*$ minimizes L and $\boldsymbol{v}^*$ maximizes L.

The existing nonlinear optimizers try to approach the solution of the nonlinear set of equations (3.3.2) and (3.3.3) through an iterative procedure, so that the Lagrangian function is minimized by the $\boldsymbol{x}$ variables and maximized by the $\boldsymbol{v}$ variables. Clearly, problems (3.3.75) and (3.3.6) are equivalent to locating the extremums of an unconstrained function f:

$$\nabla f = \mathbf{0} \tag{3.3.8}$$

Two methods have been extensively used for the solution of (3.3.8):

(i) - **Cauchy's method:** It is the famous steepest descent algorithm, which tracks the direction of the largest change in the value of the objective function:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \varepsilon \nabla f \ , \varepsilon = \pm 1 \tag{3.3.9}$$

The optimization problem (3.3.8) can be viewed as a dynamically changing system that progresses from an initial state to a final one (optimum). Equation (3.3.9) suggests the following "equation of motion" for the system:

$$\frac{d\boldsymbol{x}}{dt} = \varepsilon \nabla f \quad ; \qquad \boldsymbol{x}(0) = \boldsymbol{x}_o \tag{3.3.10}$$

Evidently, the steady states of the initial value problem (3.3.10) are identical to the roots of equation (3.3.8). These steady states correspond to the extremums of the original function f:

$$\frac{df}{dt} = \sum_i \frac{\partial f}{\partial x_i} \frac{dx_i}{dt} = \nabla \boldsymbol{f}^T \frac{d\boldsymbol{x}}{dt} \tag{3.3.11}$$

Combining (3.3.10) and (3.3.11) :

$$\frac{df}{dt} = \varepsilon \, \|\nabla f\|^2 \tag{3.3.12}$$

If $\varepsilon = -1$ the value of f monotonically decreases with time and the steady state corresponds to a local minimum.

(ii) - **Newton's method:** If second-order information about the Lagrangian function is available, a more rapid convergence is produced using Newton's approximation:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}_{(k)} + \varepsilon \, (\nabla^2 f)^{-1} \, \nabla f \ , \varepsilon = \pm 1 \tag{3.3.13}$$

with corresponding "equation of motion":

$$\frac{d\boldsymbol{x}}{dt} = \varepsilon \, (\nabla^2 f)^{-1} \, \boldsymbol{\nabla} f \tag{3.3.14}$$

Newton's method is applicable only if $\nabla^2 f$ exists and is non-singular. Under these conditions the steady states of (3.3.14) and the roots of (3.3.8) are identical. The time-behavior of the algorithm is:

$$\frac{df}{dt} = \sum_i \frac{\partial f}{\partial x_i} \frac{dx_i}{dt} = \boldsymbol{\nabla} f^T \frac{d\boldsymbol{x}}{dt} = \varepsilon \, \boldsymbol{\nabla} f^T \, (\nabla^2 f)^{-1} \, \boldsymbol{\nabla} f \tag{3.3..75}$$

$$Q = \varepsilon \, \boldsymbol{\nabla} f^T \, (\nabla^2 f)^{-1} \, \boldsymbol{\nabla} f \tag{3.3.16}$$

The sign of (3.3..75) is determined by the quadratic form Q. If Q is either positive or negative definite, the behavior of f as a function of time can be controlled through ε. If Q is indefinite, the Levenberg - Marquardt approach can be adopted, (Reklaitis *et al.*, 1983):

$$\frac{d\boldsymbol{x}}{dt} = \varepsilon \, (\nabla^2 f + \lambda \, I)^{-1} \, \boldsymbol{\nabla} f \tag{3.3.17}$$

Large values of the adjustable parameter λ turn the corresponding quadratic form into positive definite. As it is proved in Appendix I, any method that combines the above mentioned solution procedures and the results of Theorem 3.3.1 is stable.

In the next section we will combine the formulation of the problem and the dynamic solution methods in generalized Hopfield architectures, capable to implement all the important optimization algorithms (Augmented Lagrangian, Generalized Reduced Gradient, Successive Quadratic Programming) and their variants.

4. Generalized Hopfield Networks and Optimization

4.1 Unconstrained Optimization

The unconstrained optimization problem can be expressed as follows:

$$\textit{minimize} \quad f(\boldsymbol{x})$$

Applying the dynamic variation of Cauchy's method:

$$\frac{d\boldsymbol{x}}{dt} = -\nabla f \quad \rightarrow \quad \frac{dx_i}{dt} = -\frac{\partial f}{\partial x_i} \qquad i = 1,2,...,N \tag{4.1.1}$$

The corresponding Hopfield network solving the unconstrained minimization problem consists of N processors, each one representing an independent variable. Equations (4.1.1) describe the time behavior of each processor. It is clear that the input structure of each processor must be arbitrarily nonlinear, dictated by (4.1.1), as opposed to the original linear Hopfield model. We will refer to the networks of these processors as nonlinear or generalized Hopfield networks (GHN).

Example

In order to demonstrate the power of GHNs in unconstrained optimization the following problem was solved using Cauchy dynamics (Reklaitis *et al.*, 1983):

$$minimize \quad f(\boldsymbol{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 11)^2$$

As shown in Figure 2 the above function, known as Himmelblau function, possesses four local minima. Two neurons are needed, representing the problem's independent variables. The nonlinear input structure of each neuron is dictated by equation (4.1.1). Steepest descent is a local optimization algorithm. The system's steady state is determined by the choice of initial values. This is shown clearly in Figure 2, where two different initial points produce convergence to different steady states. Euler's explicit integration method was employed to calculate the trajectories shown in Figure 2.

4.2 Transformation Methods - Augmented Lagrangian

One of the first attempts to solve the general nonlinear optimization problem involved the incorporation of the constraints into the objective function. In these *transformation methods,* the constraint optimization problem is reduced to the unconstrained minimization of the lumped function:

$$P(\boldsymbol{x},R) = f(\boldsymbol{x}) + R\,\Omega(h_{i,j}) \tag{4.2.1}$$

If the constraints are violated, a large penalty (function Ω) is added to the objective function. If the value of the adjustable parameter R is appropriately updated after each iteration, the unconstrained minimum of (4.2.1) will be identical to the optimum of the original problem. It was soon realized that the transformation methods are not very efficient, because of numerical difficulties implicitly embedded in their structure, (Reklaitis *et al.*, 1983). The *Augmented Lagrangian Method* is specifically designed to avoid these problems. The transformed unconstrained objective function becomes:

$$P(\boldsymbol{x},\boldsymbol{\sigma},\boldsymbol{\tau}) = f(\boldsymbol{x}) + R \sum_j \{ <g_j(\boldsymbol{x}) + \sigma_j>^2 - \sigma_j^2 \}$$
$$+ R \sum_i \{ [h_i(\boldsymbol{x}) + \tau_i]^2 - \tau_i^2 \} \qquad (4.2.2)$$

where R is a predetermined constant. Function (4.2.2) is an approximation of the Lagrangian function (3.3.4). Variables $\boldsymbol{\sigma}$, $\boldsymbol{\tau}$ are iteratively updated by the equations:

$$\sigma_j^{(t+1)} = <g_j(\boldsymbol{x}) + \sigma_j^{(t)}>$$
$$\tau_i^{(t+1)} = h_i(\boldsymbol{x}) + \tau_i^{(t)} \qquad (4.2.3)$$

The bracket operator (<.>) is defined as :

$$<a> = \begin{cases} a & \text{if } a \le 0 \\ 0 & \text{if } a > 0 \end{cases}$$

When the process converges, $\boldsymbol{x}$ is the solution of the original problem and σ_j, τ_i the corresponding inequality - equality Lagrange multipliers. Equations (4.2.3) suggest that the dynamical behavior of a network implementing the Augmented Lagrangian optimization method is:

$$\frac{d\boldsymbol{x}}{dt} = -\nabla_x P = -\nabla f - 2R <\boldsymbol{g} + \boldsymbol{\sigma}>^T \nabla \boldsymbol{g} - 2R [\boldsymbol{h} + \boldsymbol{\tau}]^T \nabla \boldsymbol{h}$$
$$\frac{d\boldsymbol{\sigma}}{dt} = +\nabla_\sigma P = 2R <\boldsymbol{g} + \boldsymbol{\sigma}> - 2R\, \boldsymbol{\sigma} \qquad (4.2.4)$$
$$\frac{d\boldsymbol{\tau}}{dt} = +\nabla_\tau P = 2R\, \boldsymbol{h}$$

where $\nabla \boldsymbol{g}$ and $\nabla \boldsymbol{h}$ are matrices, eg. $\nabla \boldsymbol{h} = [\nabla h_1, ..., \nabla h_k]$.

In the corresponding GHN (N+K+M) neurons, representing the independent variables and the equality - inequality Lagrange multipliers, are necessary. The connectivity among the neurons (processors) is dictated by equation (4.2.4).

4.3 Generalized Reduced Gradient

In the Generalized Reduced Gradient (GRG) method, the independent variables $\boldsymbol{x}$ form two disjoint subsets:

– the set of K *basic variables* (dependent variables), $\hat{x}_i$

– the set of N-K *non-basic variables* (independent variables), $\bar{x}_j$

Then, equations (3.3.2) and (3.3.3) can be rewritten as follows:

$$\nabla \bar{f} - \boldsymbol{v}^T \nabla \bar{\boldsymbol{h}} = \boldsymbol{0} \quad (N-K \ equations) \tag{4.3.1}$$

$$\nabla \hat{f} - \boldsymbol{v}^T \nabla \hat{\boldsymbol{h}} = \boldsymbol{0} \quad (K \ equations) \tag{4.3.2}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \quad (K \ equations) \tag{4.3.3}$$

where $\nabla \bar{f}$ is a (N-K) dimension vector containing the partial derivatives of f with respect to the non-basic variables and $\boldsymbol{v}^T = [\, v_1 \, , \, . \, . \, . \, , \, v_k \,]^T$

Equations (4.3.2) can be explicitly solved for $\boldsymbol{v}^T$:

$$\boldsymbol{v} = \nabla \hat{f}^T (\nabla \hat{\boldsymbol{h}})^{-1} \tag{4.3.4}$$

Basic variables are chosen so that $\nabla \hat{\boldsymbol{h}}$ is non-singular. Equation (4.3.4) is substituted in (4.3.1), so that equations (4.3.1)-(4.3.2) are transformed to

$$\nabla \tilde{f} = \nabla \bar{f} - \nabla \hat{f} \, (\nabla \hat{\boldsymbol{h}})^{-1} \, \nabla \bar{\boldsymbol{h}} = 0 \tag{4.3.75}$$

$$\boldsymbol{h}(\boldsymbol{x}) = 0 \tag{4.3.6}$$

where $\nabla \tilde{f}$ is a modified reduced gradient for the independent variables $\bar{x}_j$ influenced by the shape of the constraints. The values of the K dependent variables $\hat{x}_i$ are computed from the system of K nonlinear equations (4.3.3). In GRG, equations (4.3.75) and (4.3.6) are solved using Cauchy's and Newton's method, respectively.

The time behavior of a dynamic optimizer that uses GRG method is represented by the following system of equations:

$$\frac{d\bar{\boldsymbol{x}}}{dt} = -\nabla \tilde{f} = -\nabla \bar{f} + \nabla \hat{f} \, (\nabla \hat{\boldsymbol{h}})^{-1} \, \nabla \bar{\boldsymbol{h}} \tag{4.3.7}$$

$$\boldsymbol{h}(\boldsymbol{x}) = 0 \quad \left(\rightarrow \frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{h} \, (\nabla \hat{\boldsymbol{h}})^{-1} \right) \tag{4.3.8}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0$$

System (4.3.7)-(4.3.8) is a differential - algebraic system, with an inherent sequential character: for each small step towards lower objective values, produced by (4.3.7), the system of nonlinear constraints should be solved, by relaxing equations (4.3.8) to a steady-state. The procedure is repeated until both equations reach a steady state. The above problem can be solved using a GHN of N-K+K=N nonlinear processors, the connectivity of which is dictated by (4.3.7) - (4.3.8). GRG uses K less processors than the Augmented Lagrangian method, but spends more effort in the computation of the reduced gradient.

4.4 Successive Quadratic Programming

In the SQP strategy, Newton's method is employed in the computation of both the independent variables, $\boldsymbol{x}$, and the Lagrange multipliers, $\boldsymbol{v}$. The state equations of a dynamic SQP optimizer are:

$$\frac{d\boldsymbol{z}}{dt} = \varepsilon \left[\nabla^2 L \right]^{-1} (\nabla L) = \varepsilon \; H \; (\nabla L) \tag{4.4.1}$$

$$\boldsymbol{z}(0) = \boldsymbol{z}_0$$

where $\boldsymbol{z}$ is the augmented set of independent variables, and L is the Lagrangian function, defined as follows:

$$\boldsymbol{z} = [\boldsymbol{x} \, ; \boldsymbol{v}]$$

$$L = f - \boldsymbol{v}^T \boldsymbol{h}$$

If matrix H is nonsingular, then the steady states of equations (4.4.1) are identical to the stationary points of L. In order to attain convergence to a local optimum of the optimization problem, which is a saddle point of L, we must guarantee, through continuous manipulation of ε, that the $\boldsymbol{x}$ state equations produce a descent in L-space and the $\boldsymbol{v}$ equations produce an ascent in it.

Example

The dynamics of the three algorithms were investigated with the following nonlinear optimization problem:

$$minimize \quad f(\boldsymbol{x}) = -x_1 \, x_2^2 \, x_3^3 \, / \, 81$$

$$subject \;\; to$$

$$h_1(\boldsymbol{x}) = x_1^3 + x_2^2 + x_3 - 13 = 0$$

$$h_2(\boldsymbol{x}) = x_2^2 \, x_3^{-1/2} - 1 = 0$$

The SQP network was an adaptation of equations (4.4.1) which used the Levenberg-Marquardt method. Figure 3 shows the transient behavior of the SQP and the Augmented Lagrangian (AL) networks, starting from a feasible initial state. The behavior of the GRG algorithm is almost identical to that of the AL. Since initially, the objective-function gradients are very small, the second-order Newton dynamics of the SQP network prevail over the first-order steepest-descent dynamics of the GRG and AL networks.

A major disadvantage of the original GRG algorithm is the requirement of feasibility for both the initial and intermediately generated points. Figure 4 shows the transient behavior of the networks starting from an infeasible initial state. Again, the GRG and AL dynamics are almost identical. All three networks converged to a local optimum. Additional experiments showed that starting from an infeasible initial point, $\boldsymbol{x}_o = [\,\hat{\boldsymbol{x}}_o\,,\,\bar{\boldsymbol{x}}_o\,]$, the GRG network always converges to a local optimum, as long as there exists a solution to the system of nonlinear equations:

$$\boldsymbol{h}\,(\,\hat{\boldsymbol{x}}\,;\bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}_o\,) \;=\; \boldsymbol{0}$$

5. Optimization and Parallel Computation

A most important application of the proposed model lies in its direct translation to a parallel algorithm which can distribute the computational burden of optimization to a large (at most N+K) number of simultaneously computing digital processors. Each one of them simulates the nonlinear analog processors of a GHN, which represents either a variable or a Lagrange multiplier and is continuously updated through the integration of the state equations:

$$x_j' \;=\; x_j \;+\; \phi\,(\,\boldsymbol{x}\,,\,\boldsymbol{v}\,)$$

where $\boldsymbol{x}$, $\boldsymbol{v}$ are the most recent updates of the independent variables and the Lagrange multipliers, and ϕ depends on both the optimization algorithm and the integration method. Here are two unique features of the algorithm:

(i) – An integral of the state equations is available, namely the Lagrangian function, which was differentiated in the first place. Thus, since only a steady-state solution is desired, it is not necessary to use a method for stiff O.D.Es. Any explicit or semi-implicit integration method will suffice.

(ii) – Because of the above and the fact that the state equations are autonomous, it is possible to update each variable in each processor completely asynchronously: the $(k+1)^{th}$ update of x_j does not require the k^{th} update of all the other variables. Thus the need to synchronize the computations done by the various processors is avoided. Consequently, the algorithm is robust with respect to intercommunication and execution delays. In contrast, the conventional forms of the optimization algorithm require complete synchronization in:

- the underlying unconstrained optimizations and

- the solution of linear (SQP) and nonlinear (GRG) algebraic equations

As a result their potential for parallelization is significantly reduced. The proposed algorithm efficiently distributes the computational burden among the parallel processors.

6. Conclusions

In this paper we presented an extension to the linear Hopfield network which can solve any constrained nonlinear optimization problem, or set of nonlinear algebraic equations, simulating any existing solution algorithm. The existence of strong nonlinearities demands significantly more expressive power than that offered by the Linear Hopfield Network.

The whole study resulted in a dynamic view of the optimization problem, which significantly contributed in gaining some insight to the problem and the existing solution methods and offered a straightforward model for parallelizing the optimization computations.

References

Bauer, F. and J.A. Nohel, *The Qualitative Theory of Ordinary Differential Equations: An Introduction (1969)*, W.A. Benjamin, Inc.

Bruck, J. and J. Goodman (1988). *On the Power of Neural Networks for Solving Hard Problems.* Neural Information Processing Systems, D.Z. Anderson (ed.), American Institute of Physics, New York, NY, 137-143.

Hopfield J.J. (1984), *Neurons with Graded Response have Collective Computational Properties like those of Two-state Neurons,* Proc. Natl. Acad. Sci. USA, vol. 81, 3088-3092.

Jeffrey, W. and R. Rosner (1986), *Neural Network Processing as a Tool for Function Optimization,* Neural Networks for Computing. J.S. Denker (ed.), American Institute of Physics, New York, NY, 241-246.

Kennedy, M.P. and L.O. Chua (1988), *Neural Networks for Nonlinear Programming,* IEEE Transactions on Circuits and Systems, vol. .75, no..75, pp..7554.7562.

Reklaitis, G.V., A. Ravindran and K.M. Ragsdell (1983), *Engineering Optimization: Methods and Applications,* Wiley - Interscience.

Tank, D.W. and J.J. Hopfield (1986), *Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit,* IEEE Transactions on circuits and systems, CAS-33, no..75.

Appendix I - Extension of Lyapunov's Stability Theorem

We consider the autonomous system of n nonlinear O.D.Es:

$$\frac{dx_i}{dt} = f_i(x_1, ..., x_n) \qquad i = 1,2,...,n \tag{I.1}$$

where f_i and $\frac{\partial f_i}{\partial x_j}$, $j = 1,2,...,n$ are continuous in a region D of the n-dimensional $\boldsymbol{x}$ - space. We will assume that D contains the origin $\mathbf{0}$, which is a critical point such that:

$$f_i(\mathbf{0}) = 0 \qquad i = 1,2,...,n \tag{I.2}$$

The above assumption is not restrictive. Any critical point of $\boldsymbol{f}$ can become the origin with the appropriate translation of the coordinate axes.

Lemma I.1. *Suppose that $V(\boldsymbol{x})$ is a scalar function of the n-dimensional vector, $\boldsymbol{x}$, with the following properties:*

(i) V is continuous with continuous first partial derivatives inside a region Ω of the domain of $\boldsymbol{x}$, containing the origin, $\boldsymbol{x} = \mathbf{0}$.

(ii) V has a stationary point (minimum, maximum or saddle point) at the origin.

Then,

(i) There exists a neighborhood D around $\mathbf{0}$ inside which no other stationary points of V exists ($D = D_1 \times D_2 \times \cdots \times D_n$).

(ii) If $\boldsymbol{c} \in D$ then the single-variable scalar functions (intersections) defined by the equations:

$$V_k(\boldsymbol{c}\ ; x_1 = c_1, ..., x_k, ..., x_n = c_n) \qquad k = 1,2,...,n \tag{I.3}$$

also have, in the neighborhood D_k, a single stationary point, $\xi_k(\boldsymbol{c})$.

(iii) The points $\boldsymbol{x}_k^* = (c_1, c_2, ..., c_{k-1}, \xi_k, c_{k+1}, ..., c_n)$, for every $c_1 \in D_1$, $c_2 \in D_2$, $c_{k-1} \in D_{k-1}$, $c_{k+1} \in D_{k+1}$, ..., $c_n \in D_n$. form a curve, called a stationary point curve, C_k with the following properties:

(α) C_k is a continuous curve with no cycles.

(β) There exists one and only one point, identical to the origin, on which all the curves C_k , $k = 1, ..., n$ meet.

Proof: We will prove each proposition separately:

(i) If there did not exist such a neighborhood, then $\mathbf{0}$ would not be a stationary point, or V would be constant.

(ii) The functions $V_k(\boldsymbol{c}\ ; x_k)$ are continuous and continuously differentiable, otherwise V would not possess the same properties. If no stationary points existed for the functions V_k inside D, then V_k would be either constant, thus making V constant, or discontinuous, also making V discontinuous. No more than one stationary point can exist for each V_k. Otherwise, there would exist either a discontinuity in V, or multiple stationary points of V in D, not allowed by (i).

(iii - α) The continuity of V and property (ii) do not allow for a discontinuity of curve C_k, inside the region D. The existence of a single stationary point ξ_k for each V_k and $\boldsymbol{c}$ inside D precludes the existence of cycles in the curve C_k.

(iii - β) It is clear that all curves C_k, $k = 1, 2, \ldots, n$, are met on the origin, which is a stationary point of V. Any other meeting point would also have all partial derivatives equal to zero, therefore it would be another stationary point inside D, thus contradicting with proposition (i).

Theorem I.1. *Suppose that for the nonlinear system (I.1) one can identify a continuous scalar function $V(\boldsymbol{x})$ with the following properties:*

(i) V is continuous with continuous first derivatives.

(ii) $\mathbf{0}$ is a saddle point of V.

(iii) If we partition the set of independent variables x_i into two subsets:

$$\bar{X} = \{x_k \mid V_k(\mathbf{0}; x_k) \text{ have a minimum at } x_k = 0\}$$

$$\hat{X} = \{x_k \mid V_k(\mathbf{0}; x_k) \text{ have a maximum at } x_k = 0\}$$

the following relations are true, at least for a neighborhood Ω around the origin:

$$sgn\left(\frac{\partial V}{\partial x_k}\right) = -sgn(f_k), \qquad \forall x_k \in \bar{X}$$

$$sgn\left(\frac{\partial V}{\partial x_k}\right) = sgn(f_k), \qquad \forall x_k \in \hat{X}$$

where $sgn(.)$ function returns the sign of the argument.

(iv) $|\bar{X}| + |\hat{X}| = n$ (meaning that there is no V_k with an inflection point).

Then, the origin $\boldsymbol{x} = \mathbf{0}$ *is a stable critical point of (I.1).*

Proof: Lemma I.1 guarantees the existence of the stationary point curves C_k. We will prove that there exists a function W, the dual of V, which is positive definite and for which the origin $\boldsymbol{x} = \mathbf{0}$ is the sole minimum. The Energy function W is constructed as follows:
(i) The points $\boldsymbol{x}_0$ that belong to curves C_k are given zero energy:

$$W(\boldsymbol{x}_0) = 0$$

(ii) For each other point $\boldsymbol{x}_0 = (x_{01}, ..., x_{0n}) \in D$ we define a 'distance' measure from each curve C_k, defined as:

$$d_k(\boldsymbol{x}_0) = \| \boldsymbol{x}_0 - \boldsymbol{x}_k^* \|$$

where $\boldsymbol{x}_k^*$ is the stationary point of $V_k(\boldsymbol{c}\ ; x_k)$, with $c_j = x_{0j},\ j \neq k$. and $\|.\|$ is any vector norm of the n-dimensional real vector space. Note that for each $\boldsymbol{x}_0$ there is only one $\boldsymbol{x}_k^*$ in D. Then,

$$W(\boldsymbol{x}_0) = \sum_{k=1}^{n} \alpha_k d_k(\boldsymbol{x}_0)$$

where α_k are positive constants. It is clear from Lemma I.1 that W function has only one zero, namely at the origin. Condition (iii) implies that if V_k has a minimum at $x_k = 0$ then equation

$$\frac{dx_k}{dt} = f_k$$

drives the system towards it, given that $x_j,\ j \neq k$ are kept constant:

$$sgn\left(\frac{dV}{dt}\right) = sgn\left(\frac{\partial V}{\partial x_k}\frac{dx_k}{dt}\right) = -1$$

Hence it tries to minimize $d_k(\boldsymbol{x})$ and consequently $W(\boldsymbol{x})$. Therefore:

$$\frac{dW}{dt} \leq 0$$

The Energy function W and the system (I.1) make the preconditions of the original Lyapunov's theorem true. Therefore, the origin $\mathbf{0}$ is asymptotically stable.

Notes:

(i) Lyapunov's stability theorem, (Bauer *et al.*, 1969), is concerned with energy functions having a minimum (maximum) and nonlinear O.D.Es that try to minimize (maximize) it. Clearly, Theorem I.1 is an extension to the original theorem that also covers energy functions having a certain kind of saddle points.

(ii) It can be easily proved that if the stationary points of some V_k around the origin are inflection points, then system (I.1) is unstable.

(iii) It is the dual function W that monotonically decreases in time. The value of V, which has a saddle point at the origin, can either decrease, increase or approach the stable value $V(\mathbf{0})$ in an oscillatory way.

Theorem I.2. *The system of nonlinear O.D.E.s, produced by the application of Cauchy's dynamic method to the general optimization problem, as expressed by equations (3.3.2') and (3.3.3'), has some stable points that coincide with the roots of the above system of equations.*

Proof: As stated in Theorem 3.3.1, in the neighborhood of a solution of the system of equations (3.3.2') - (3.3.3'), $\boldsymbol{x}$ variables minimize the Lagrangian function whereas $\boldsymbol{v}$ maximize it. Therefore, according to Theorem I.1, any dynamical method that treats $\boldsymbol{x}$ as minimizing variables and $\boldsymbol{v}$ as maximizing variables asymptotically converges to the solution of the system above. The above result is also true for the application of Newton's method, as long as the quadratic form Q in equation (3.3.12) is positive definite.

Theorem I.3. *The GRG optimization method is stable.*

Proof: GRG attempts to eliminate the basic variables $\hat{\boldsymbol{x}}$ by implicitly solving equation (4.3.6) and solves the resulting unconstrained optimization problem (4.3.75). Therefore, equations (4.3.7) are stable, with corresponding Lyapunoff function the modified, reduced objective function, $\tilde{f}$.

If $\nabla\hat{h}$ is nonsingular, then the system (4.3.8) has as steady-states the solutions of (4.3.6). Moreover, system (4.3.8) is stable, with corresponding Lyapunov function:

$$L = \boldsymbol{h}^T \boldsymbol{h} = \sum_{i=1}^{K} h_i^2$$

Note: Equations (4.3.7) and (4.3.8) are integrated in a sequential manner. For each new set of $\bar{\boldsymbol{x}}$ - values, produced by the steepest descent in the $\tilde{f}$ space, the corresponding $\hat{\boldsymbol{x}}$ (dependent) values must be computed by relaxing (4.3.8) to an intermediate steady-state, corresponding to the solution of (4.3.6).

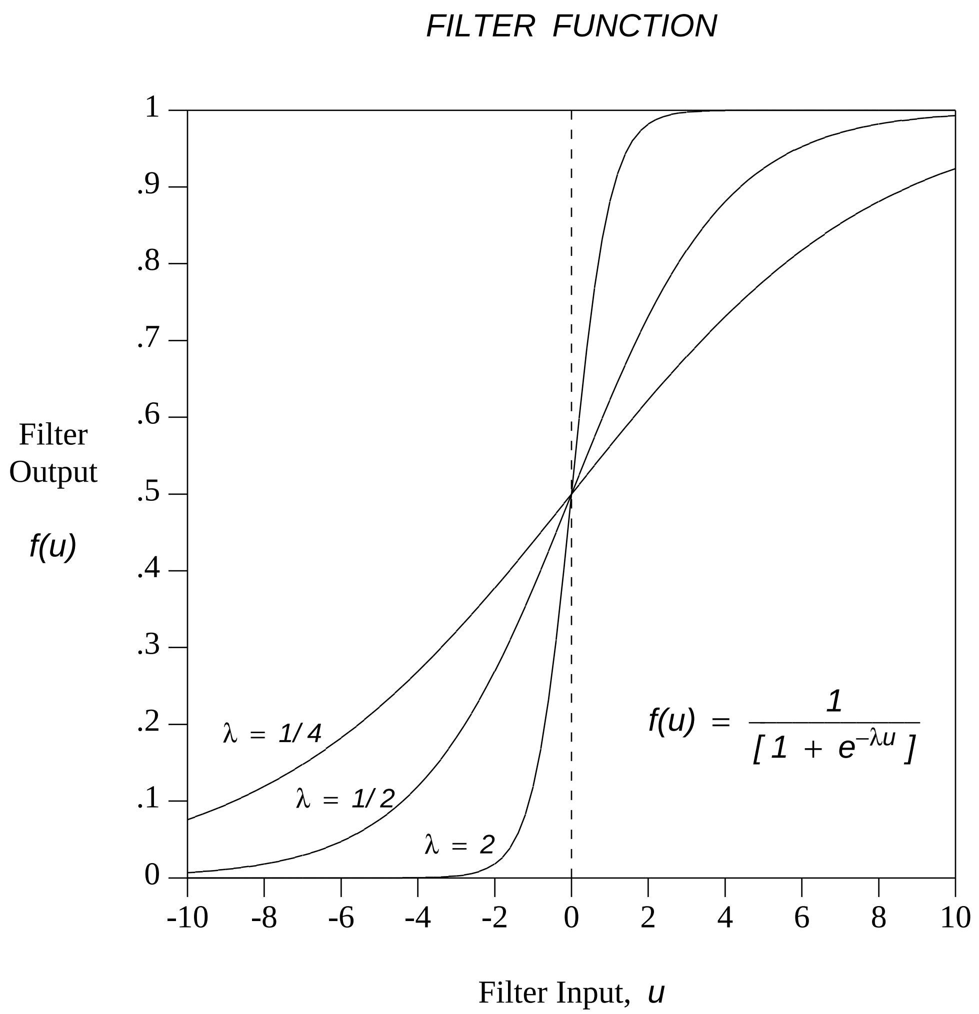

Figure 1. Filter Function and Influence of λ.

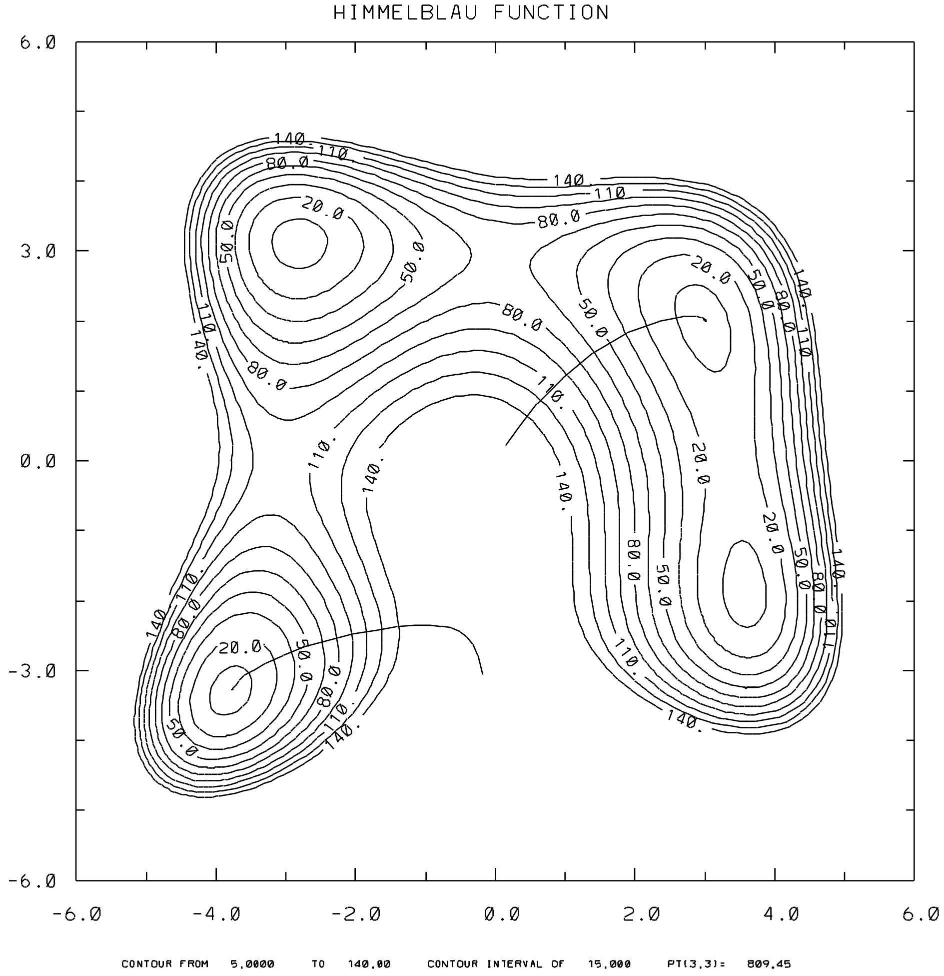

Figure 2. Convergence to Local Optima

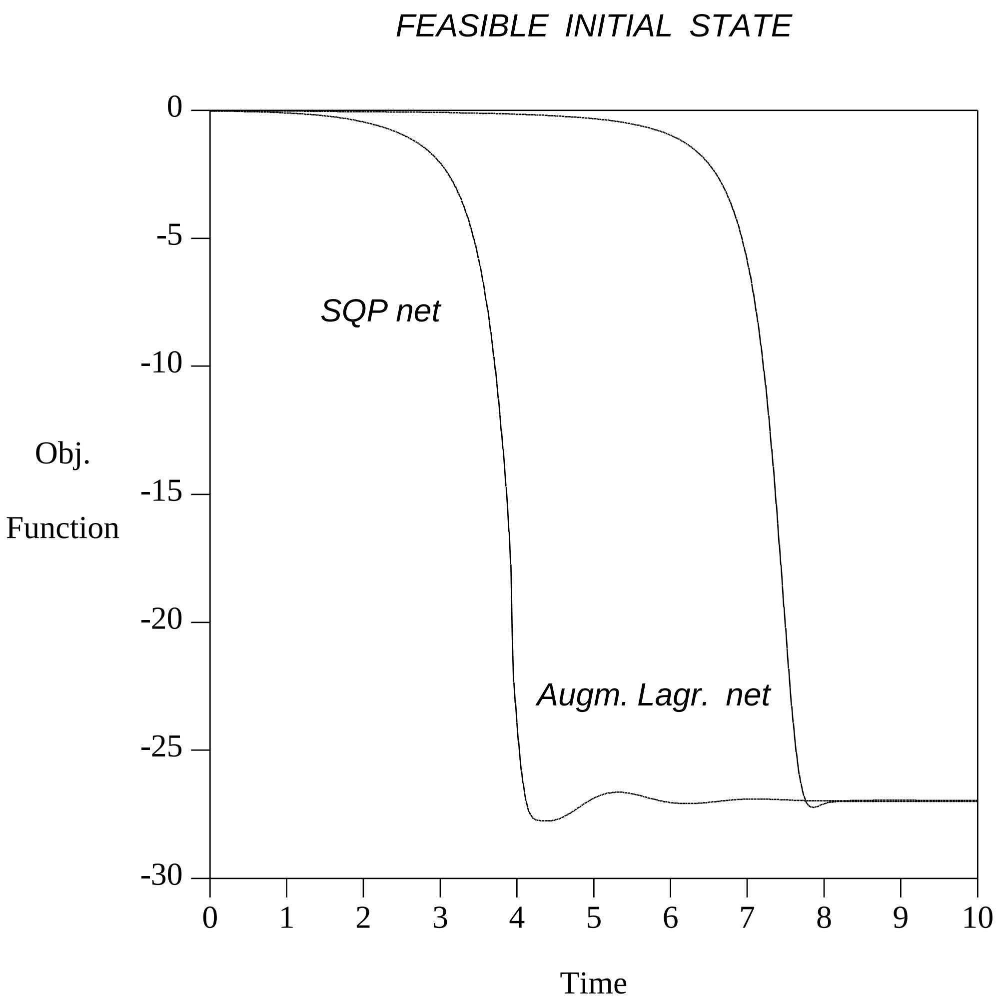

Figure 3. Network Dynamics and Feasible Initial State.

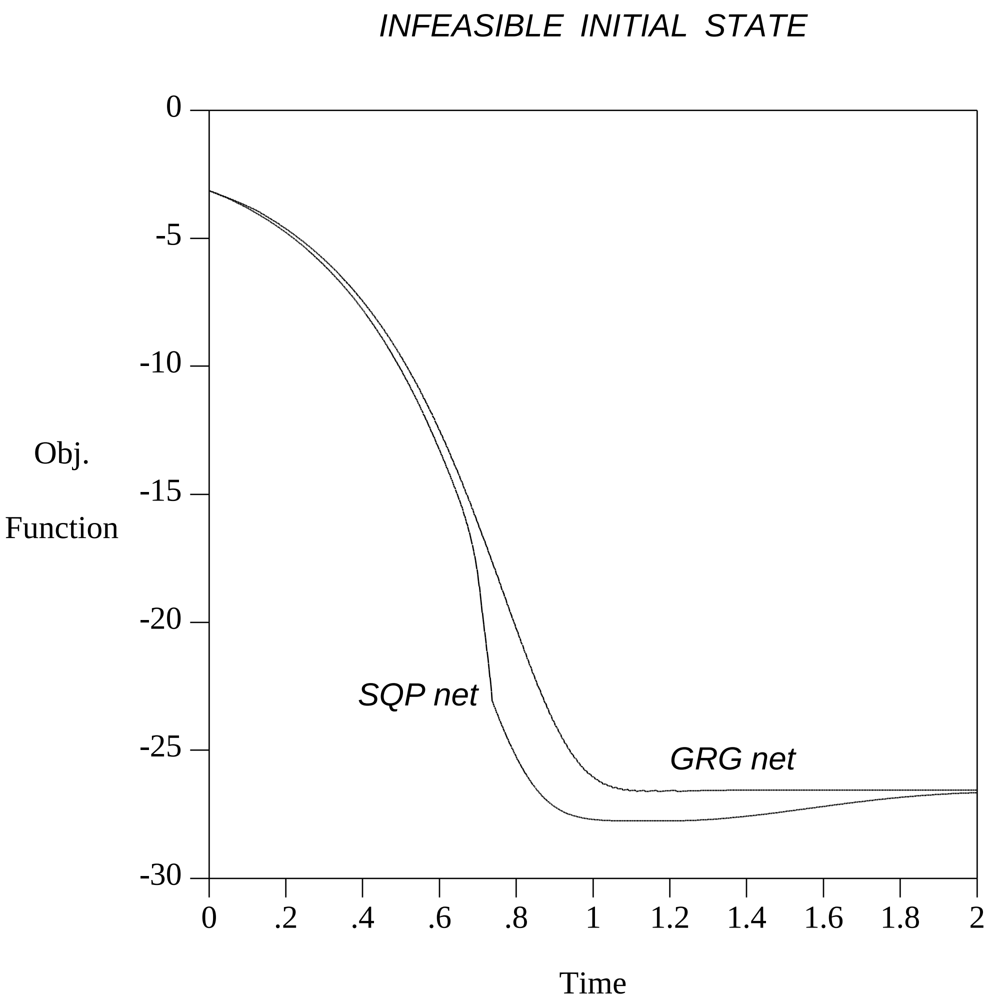

Figure 4. Network Dynamics and Infeasible Initial State.