
LINEAR PROGRAMMING AND RECURRENT ASSOCIATIVE MEMORIES

Robert Kalaba[1]
Moon Kim[2]
James E. Moore II[3]*

A B S T R A C T

Many optimization procedures presume the availability of an initial approximation in the neighborhood of a local or global optimum. Unfortunately, finding a set of good starting conditions is itself a nontrivial proposition. We describe a procedure for identifying approximate solutions to constrained optimization problems. Recurrent neural network structures are interpreted in the context of associative memories. Associative memory matrices are trained to map the inputs of closely related transportation linear programs to optimal solution vectors. The procedure performs well when training cases are selected according to a simple rule, identifying good heuristic solutions for representative test cases. Modest infeasibilities may exist in these estimated solutions, but the basic variables associated with true optimums are usually apparent. In the great majority of cases, rounding identifies the true optimum.

Key words: linear programming, associative memories, neural networks, optimization, transportation

DRAFT: Not for citation. Prepared for the International Joint Conference on Neural Networks. To be revised and submitted to **General Systems**.

School of Urban and Regional Planning
Von Kleinsmid Center - 351
University of Southern California
University Park
Los Angeles, CA 90089-0042

June, 1990

*author to whom correspondence should be addressed

----

1 Professor of Electrical Engineering, Biomedical Engineering, and Economics, University of Southern California.
2 Ph.D. Candidate, School of Urban and Regional Planning, University of Southern California.
3 Assistant Professor of Urban and Regional Planning, Civil Engineering, and Industrial and Systems Engineering, University of Southern California.
The authors gratefully acknowledge substantive assistance and comments from Peter Gordon, Associate Dean of the School of Urban and Regional Planning and Professor of Economics, University of Southern California. The usual caveat applies.

1. Introduction

Operations research techniques have been applied to numerous resource allocation problems in urban planning. Much of the impetus for these developments was provided by Dorfman, Samuelson and Solow's[1] path-breaking text on linear programming and economic efficiency. This work was especially important in the context of land use and transportation planning, because these efforts involve organizing and managing the market for urban land. The connection between linear programming and general equilibrium economics provided urban planners with a new view of the field's objectives.

The influence of operations research on spatial resource allocation problems can be traced to at least three seminal papers. Beckmann, McGuire and Winsten[2] formulated a nonlinear program that identified competitive equilibrium flows for a transportation network with congestive links. Koopmans and Beckmann[3] formulated a quadratric assignment problem with the objective of minimizing the location and interaction costs for a set of discrete activities located on a network. And finally, Hakimi[4] considered the problem of locating one or more discrete facilities on a network to minimize either the sum of distances or the maximum distance between facilities and network nodes. Much important work has followed these efforts, and operations research continues to lend substantive theoretical insights to urban planning and regional science activities.

Because operations research is such an extremely valuable planning tool, improved heuristic and optimal solution algorithms have considerable bearing on urban planning research. Our current efforts focus on fundamental OR formulations germane to urban form, structure, and activities. Many of these formulations are discrete or otherwise nonlinear. Nonlinear optimization techniques consistently require good starting conditions, usually in the form of good feasible solutions. Gradient search and other numerical analysis procedures perform well in the neighborhood of a local optimum, but how does an investigator locate the right neighborhood? Even in the convenient case of a linear program, finding an initial feasible solution has routinely proved as difficult as finding the program's global optimum. Identifying good, infeasible starting conditions for linear programming problems is an important area of research, because an effective crash procedure can significantly reduce the computational burden of locating an initial feasible solution.

Our work is motivated by the desire to find an inexpensive procedure for identifying good starting conditions, feasible or otherwise, for constrained optimization problems relevant to the urban environment. The experiment described below involves estimating the optimal solutions for a population of transportation linear programs. We believe that this approach has considerable merit, and are working to extend this approach to more complicated urban programming problems.

2. Network Problems

For four decades linear programming has held a pre-eminent position in economics and operations research. This is primarily due to the utility of the simplex method, a solution algorithm developed by George Dantzig[5]. Among the basic linear programming formulations, the transportation problem occupies a prominent position. This problem requires the determination of an optimal shipping schedule, X_{ij}, from source i to sink j such that the cost function

$$\sum_i \sum_j \; c_{ij} \bullet X_{ij} \qquad \textbf{(1)}$$

is minimized, subject to the constraints

$$\sum_j X_{ij} \leq a_i, \qquad i = 1,2, \ldots, m, \tag{2}$$

$$\sum_i X_{ij} \geq b_j, \qquad j = 1, 2, \ldots, n, \tag{3}$$

$$X_{ij} \geq 0, \qquad i = 1,2, \ldots, m;\ j = 1, 2, \ldots, n. \tag{4}$$

For consistency, it is further assumed that

$$\sum_i a_i = \sum_j b_j, \tag{5}$$

which assures that supplies are exactly adequate to meet demands. A significant aspect of this problem is to study the sensitivity of the optimal shipping schedules to variations in the configuration of supplies and demands. We show, experimentally, that recent developments in one of the simpler areas of artificial neural networks hold promise in this context.

Hopfield and Tank[6] used a recurrent neural network to find good feasible solutions to the Traveling Salesman Problem. Their exercise in supervised learning is insightful, but Hopfield and Tank are logically silent on the question of how representative training cases should be generated if this involves solving NP-complete optimization problems. Unlike the problem of finding an optimal tour, transportation linear programs can be solved in polynomial time. Consequently, solutions can be generated inexpensively, and supervised learning procedures are simple to employ.

We employ Kohonen's[7] concept of an associative memory matrix to map transportation linear programs to optimal solutions. As Figure 1 indicates, associative memories address the pair-associate problem. Does there exist an associative memory **M** that will always map a finite set of arbitrarily selected stimulus vectors to the corresponding set of response vectors? Associative memory matrices are simple neural structures, but the notion of stimulus and response remains crucial. For each of K training cases, let the stimulus vector $\mathbf{s_k}$ of dimension px1, and the response vector $\mathbf{r_k}$ of dimension qx1, be specified. Our objective is to determine an associative memory matrix $\mathbf{M^*}$ of dimension pxq such that $\mathbf{M^* \bullet s_k}$ will as nearly as possible equal $\mathbf{r_k}$ for k = 1, 2, …, K. Following

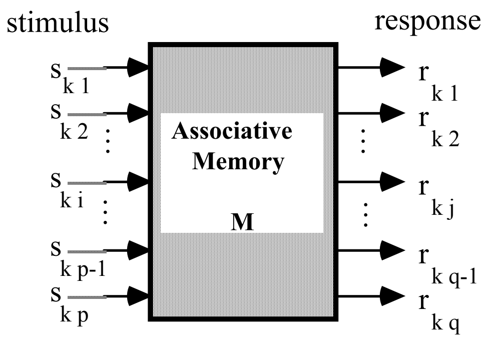

Figure 1: An Ideal Associative Memory

Kohonen[7], we take this to mean that if we form the stimulus matrix **S** of dimension pxK, whose kth column is $\mathbf{s_k}$, and the response matrix **R** of dimension qxK, whose kth column is $\mathbf{r_k}$, then the matrix **M*** is to be determined by minimizing the L_2 norm of the difference matrix **R - M•S**. That is,

$$\mathbf{M}^* = \min_{\mathbf{M}} \|\mathbf{R} - \mathbf{M}\bullet\mathbf{S}\|^2. \tag{6}$$

Obviously, minimizing the L_2 norm minimizes mean square error. The solution to this problem[8] is

$$\mathbf{M}^* = \mathbf{R}\bullet\mathbf{S}^+ \tag{7}$$

where $\mathbf{S}^+$, dimension Kxp, is the Moore-Penrose generalized inverse of the rectangular matrix **S**. Codes for calculating this psuedo inverse are available in standard software packages such as SPEAKEASY and MATLAB. Codes that use approximations to $\mathbf{S}^+$ to reduce numerical instability are currently under investigation at the University of Southern California.

Pure network problems are particularly appropriate candidates for this methodology. They are unimodular, and representative formulations can be restricted to integer inputs and outputs with no loss of generality. The coefficients in network constraints form a sparse matrix of ones and zeroes that is fixed for a given network configuration. Most urban transportation planning problems involve determining network performance in response to changes in the demand for service, possibly as part of a larger search for optimal supply alternatives. Even if link costs vary with link flows (which they do in an urban context), or origin-destination flows are not substitutable, *network optimization problems are essentially vector mapping problems if the physical configuration of the network is fixed*. A vector of link cost parameters and/or origin-destination requirements defines the formulation, and the solution is a vector of link or path flows.

In the application presented below, the right hand side of a transportation linear program is designated the stimulus vector, with the associated response vector being the optimal shipping schedule and objective function value. Finding a suitable set of training cases specifies the matrices **R** and **S**. Each training case involves the solution of a transportation problem, for which SPEAKEASY contains a conventional algorithm.

In view of the actual nonlinear relation between the transportation problem's supplies and demands and the corresponding optimal shipping schedule, it is not clear that any memory matrix **M*** would adequately map stimuli into responses in this case. Furthermore, if the product $\mathbf{M}^*\bullet\mathbf{s_k}$ is formed for a stimulus that is not in the training set, we may well wonder if the product yields a useful approximation to the optimal solution. Previous work[9] indicates that this simple technique works surprisingly well. We present some further numerical results, and an extension that incorporates recurrent structure into the associative memory approach.

2.1 A Representative Network

We tested the utility of this procedure on a small transportation linear program, minimizing

$$c_{11}\bullet X_{11} + c_{12}\bullet X_{12} + c_{13}\bullet X_{13} + c_{21}\bullet X_{21} + c_{22}\bullet X_{22} + c_{23}\bullet X_{23} \tag{8}$$

subject to

$$X_{11} + \quad X_{12} \quad X_{13} + \qquad\qquad \leq \quad a_1, \tag{9}$$

$$X_{21} + \quad X_{22} + \quad X_{23} \leq a_2, \tag{10}$$

$$X_{11} + \qquad X_{21} \geq b_1, \tag{11}$$

$$X_{13} + \qquad X_{23} \geq b_3, \tag{13}$$

$$X_{11}, \quad X_{12}, \quad X_{23}, \quad X_{21}, \quad X_{22}, \quad X_{23} \geq 0. \tag{14}$$

The graph for this formulation appears in Figure 2. Even for such a small network, large numbers of training cases can be generated by permuting coefficients in the objective function and/or right hand side. In this case, we set

$$\text{total supply} = \textstyle\sum_i a_i = \sum_j b_j = \text{total demand} = 6, \tag{15}$$

and fixed **c** the vector of objective function coefficients at

$$c_{11} = 1, \tag{16}$$

$$c_{12} = 2, \tag{17}$$

$$c_{13} = 3, \tag{18}$$

$$c_{21} = 4, \tag{19}$$

$$c_{22} = 5, \tag{20}$$

$$c_{23} = 6. \tag{21}$$

As noted above, a stimulus vector is defined to be the right hand side of the constraint set

$$\mathbf{s_k} = \begin{vmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \\ b_3 \end{vmatrix}, \tag{22}$$

and the associated response vector is taken to be the optimal set of transportation flows, plus the optimal value of the objective function

$$\mathbf{r_k} = \begin{vmatrix} X_{11} \\ X_{12} \\ X_{23} \end{vmatrix} \tag{23}$$

$$\begin{vmatrix} X_{21} \\ X_{22} \\ X_{23} \\ Z \end{vmatrix}.$$

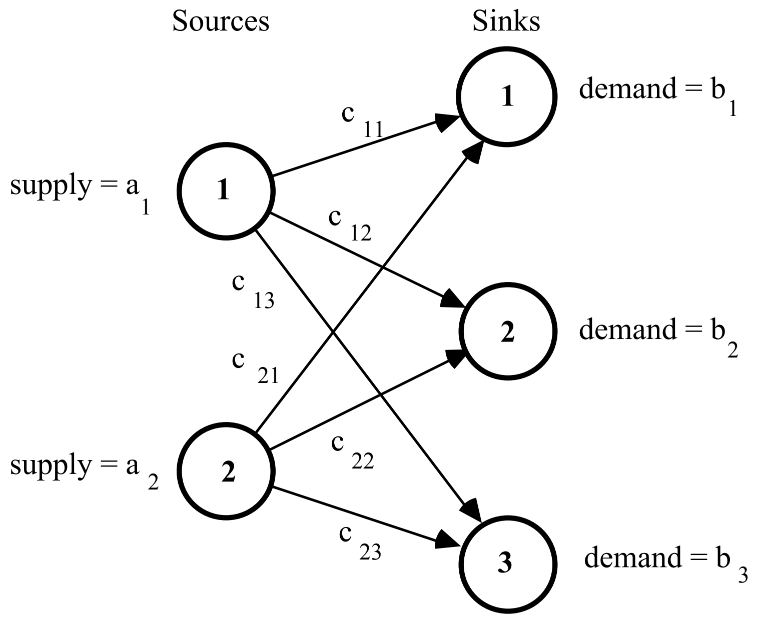

Figure 2: A Simple Transportation Network

Permuting feasible integer values for the right hand side generates a population of 196 stimulus-response pairs.

Any n-subset of these paired stimulus and response vectors defines the respective columns of the training matrices $\mathbf{S}_{5xn}$ and $\mathbf{R}_{7xn}$. The objective is to use this training information to define an associative memory matrix $\mathbf{M}^*_{7x5}$ that produces good estimates of the response (solution) vectors associated with new stimulus (formulation) vectors

$$\mathbf{M}^* \bullet \mathbf{s}_k = \mathbf{r}^*_k. \qquad \textbf{(24)}$$

2.2 Selecting a Training Set

How to select a representative training set is not clear. Kohonen[7] suggests that stimulus vectors be selected in a way that maximizes their mutual orthogonality. We used a random sampling procedure to identify a class of training vectors that yields good results. 20 random samples of sizes 5, 10, 20, and 30 were used to compute a total of 80 matrices **M***. In each case, the associative memory matrix was used to estimate the response vectors for the stimuli not included in the training sample. Three results are apparent. First, the root mean square (RMS) error measure computed for each such test set attenuates rapidly as the number of training cases used to determine **M*** increases. Second, there is an obvious lower bound on these RMS values. And third, certain classes of training vectors are consistently associated with lower RMS measures than others. The use of RMS as an error measure is arbitrary. It is a conventional measure that is easier to compute than the mean absolute deviation (MAD), and which permits **M*** to be interpreted as the best linear unbiased estimator (BLUE) in the special case of a linear statistical relationship between stochastic vectors $\mathbf{s}_k$ and $\mathbf{r}_k$.

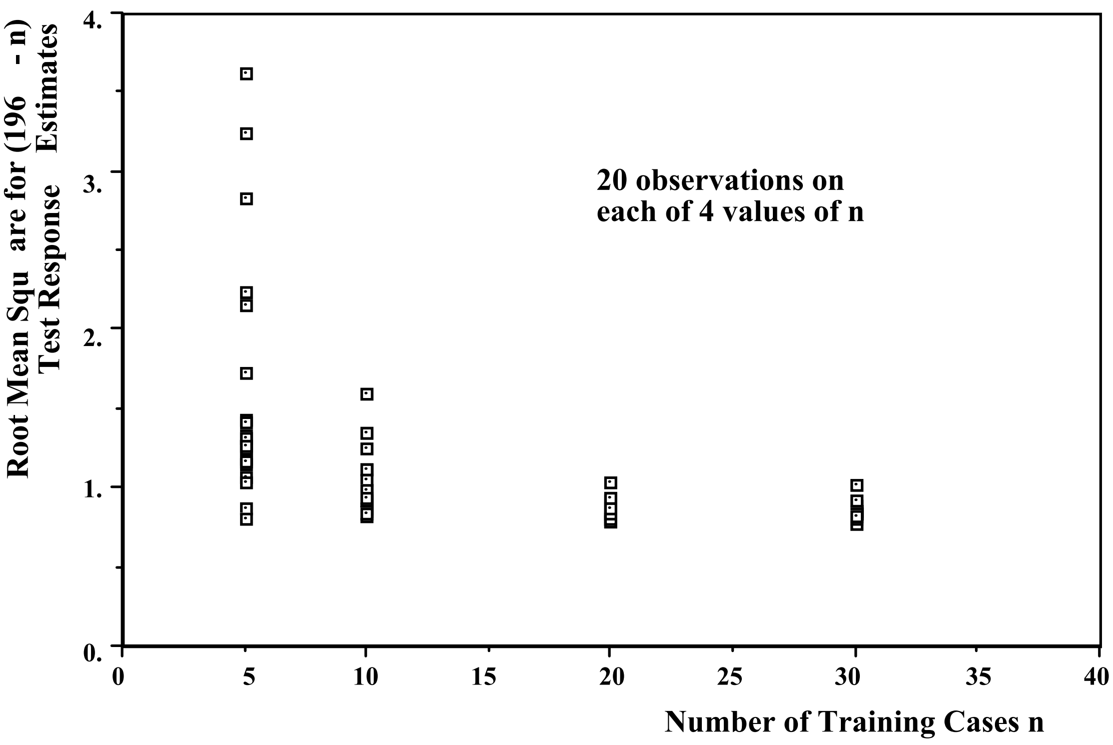

Figure 3: Root Mean Square Error of Estimation Vs. Number of Training Cases: Four Samples of Size Twenty.

These results imply that there is an efficient dimension for the training matrices **S** and **R**. More importantly, training sets within which each vector element assumes a representative range of values consistently provide test estimates that have lower RMS values. Based on these criteria, we selected a representative training set of 23 stimulus-response pairs. The corresponding matrices **S** and **R** are presented in the Appendix. These imply the following associative memory matrix

$$\mathbf{M}^* = \begin{vmatrix} 0.60000 & 3.60000 & 0.40000 & 1.40000 & 2.40000 \end{vmatrix} \qquad \textbf{(25)}$$

$$\begin{vmatrix} 0.24063 & -0.09341 & 0.34614 & -0.06979 & -0.12913 \\ 0.30062 & -0.10079 & -0.05326 & 0.35638 & -0.10329 \\ 0.25875 & -0.00580 & -0.09288 & -0.08659 & 0.43241 \\ -0.04064 & 0.29341 & 0.45386 & -0.13021 & -0.07087 \\ -0.10062 & 0.30079 & -0.14674 & 0.44362 & -0.09671 \\ -0.05875 & 0.20580 & -0.10712 & -0.11341 & 0.36759 \end{vmatrix}.$$

3. Numerical Results: The Nonrecurrent Case

Forming the products $\mathbf{M}^*\bullet\mathbf{s}_k$ for the 173 formulation vectors $\mathbf{s}_k$ not included in the training set produces test estimates $\mathbf{r}^*_k$ for the corresponding solution vectors $\mathbf{r}_k$. The average RMS measure for these test estimates was 0.7231, versus an average RMS of 1.0049 for the estimates of the 23 training responses, implying that the training set is indeed representative of the test population. A scatterplot of the nonrecurrent training estimates versus optimal link flows is shown in Figure 4. If the nonrecurrent associative memory was returning perfect estimates of the optimal link flows, the data points on this graph would all fall on the 45 degree line. The estimates covary with the true optimal flows, but weakly: The variance in the optimal link flows accounts for 64.4 percent of the variance observed in the training estimates.

If negative numbers are always rounded to zero, then rounding the estimated elements of the training and test response vectors to the nearest integer value produces one of four outcomes:

- the optimal basis and solution are correctly identified;
- the optimal basis is correctly identified but some numerical values are incorrect;
- the optimal basis is not correctly identified because some nonbasic variables have been rounded to positive values, but no basic variables have been rounded to zero; and
- the optimal basis is not correctly identified because some nonbasic variables have been rounded to positive values, and <u>some basic variables have been rounded to zero</u> .

In either of the middle circumstances, the optimal solution can be obtained by solving a new, simpler linear program in which all of the nonbasic variables are constrained to zero. The last outcome is much more problematic, because variables in the optimal basis have been incorrectly identified as nonbasic. Fortunately, the best and worst case results were, respectively, the most and least frequent of the outcomes we obtained. The relative frequencies with which the associative memory matrix $\mathbf{M}^*$ generates these outcomes in these four categories are summarized in Table 1.

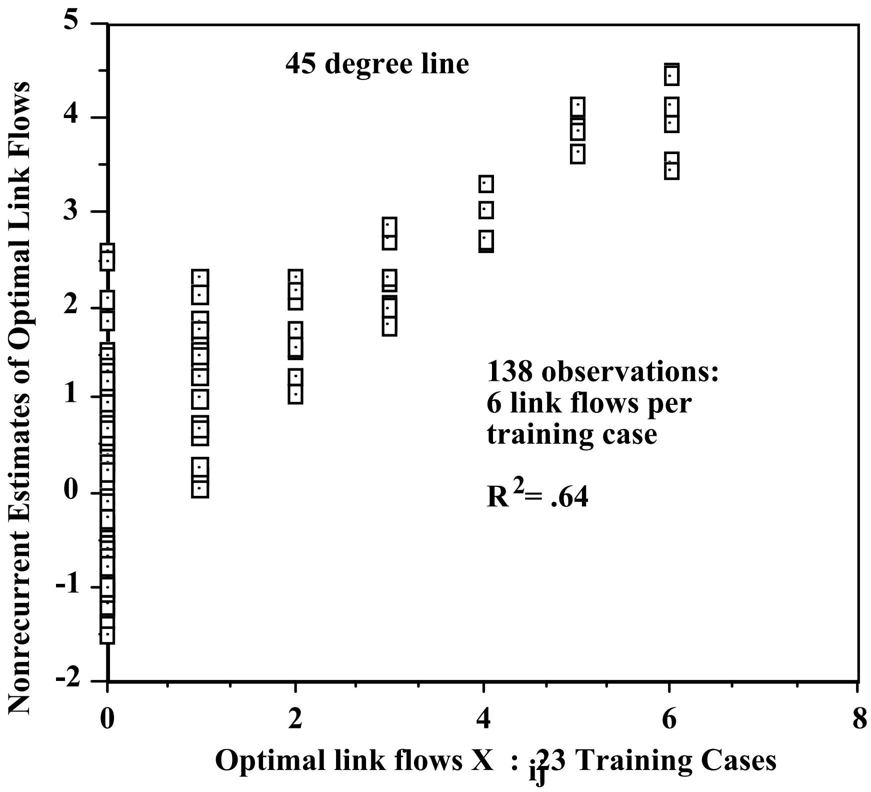

Figure 4: Nonrecurrent Estimates of Optimal Link Flows Vs. Optimal Link Flows: Twenty-Three Training Cases

Table 1: Rounding Results: Estimated Solutions $\mathbf{M^* \bullet s_k} = \mathbf{r_k}^*$ vs. True Solutions $\mathbf{r_k}$

Rounding Identifies the Optimal Solution	Rounding Identifies the Optimal Basis	Rounding Produces Positive Nonbasic Variables	Rounding Produces Zero-Valued Basic Variables
23 Training Vectors			
0	0	20	3
173 Test Vectors			
35	11	97	30

4. Extending the Stimulus Vectors

This simple procedure performs surprisingly well, but there are avenues for improvement. The most obvious is to extend the dimension of the stimulus vector to include quadratic or other nonlinear combinations of the stimulus elements[10]. This extension improved the estimates obtained for a previous exercises in optimization[9] and signal processing[11]. In this case, however, extending the stimulus vectors to include all possible quadratic combinations of the original stimulus elements provides only marginal improvement. Extending the dimension of $\mathbf{s}_k$ increases the dimension of $\mathbf{M}^*$. Clearly, a larger associative memory matrix can encode more information, and thus we would never expect extending either the stimulus vectors to cause an increase in the errors associated with test response estimates. However, there is a trade-off between the computation cost of identifying $\mathbf{M}^*$ and the benefits from using polynomial versions of the stimulus vectors. Our experiences indicate that, as the systems being modeled become more complex, the trade-off favors shorter stimulus vectors.

It is tempting to pursue a more sophisticated artificial neural network approach, such as a Rumelhart[12, 13] net in which internal layers of hidden units with thresholding operations are used to provide a representation for the nonlinear relations connecting stimulus and response vectors. One difficulty with this latter approach is that it is not possible to know in advance just how many hidden units and layers will suffice for the problem at hand. Moreover, obtaining the connective strengths for a Rumelhart network is a nonlinear optimization problem for which convergence is not always assured[14].

4.1 Nonlinear Recurrence

There is a simpler alternative available. Rummelhart networks are feed forward networks, and thus nonrecurrent[15]. Such nonrecurrent networks are unconditionally stable, because outputs do not feed back to inputs. In contrast, recurrent networks are characterized by such feed backs. This makes recurrent networks dynamic, and endows them special utility as associative memories. Because a trajectory of network outputs is available from a single input, recurrent networks have considerable potential for data compression and pattern recognition[7, 15]. For example, the Bidirectional Associative Memory[16] is a recurrent network structure that recognizes patterns by using estimated outputs to refine inputs, and then feeding back the refined inputs to improve the estimated outputs. Unfortunately, recurrent networks may also be unstable. Outputs might never converge. A sufficient but not a necessary condition for recurrent network stability has been defined by Cohen and Grossberg[17].

We attempted to further reduce test case RMS by interpreting recurrent neural network structure in the context of Kohonen's associative memory matrix. This extension is summarized by the flowchart in Figure 5. An initial associative memory matrix $\mathbf{M}^*$ is obtained as before. Training stimulus vectors are subsequently extended by appending nonlinear transformations f(•) of the estimated training response vectors, and a recurrent associative memory matrix $\mathbf{M}^{**}$ is computed based on the extended training stimulus matrix $\mathbf{S}$. Test stimuli $\mathbf{s}_k$ are initialized by appending nonlinear transformations of the estimated response vectors $\mathbf{r}^*_k = \mathbf{M}^* \bullet \mathbf{s}_k$. The definition of the test stimulus vectors $\mathbf{s}_k$ is updated to the extended version, and new test response estimates $\mathbf{r}^{**}_k$ are computed as the product of the recurrent memory matrix $\mathbf{M}^{**}$ and the extended test stimuli $\mathbf{s}_k$. These response estimates $\mathbf{r}^{**}_k$ are improved versions of the response information used to extend $\mathbf{s}_k$. If each of the response components used to extend the vectors $\mathbf{s}_k$ is updated to the improved version, further improvement results each time the operation $\mathbf{M}^{**} \bullet \mathbf{s}_k$ is performed. This updating procedure continues until improvements are sufficiently small.

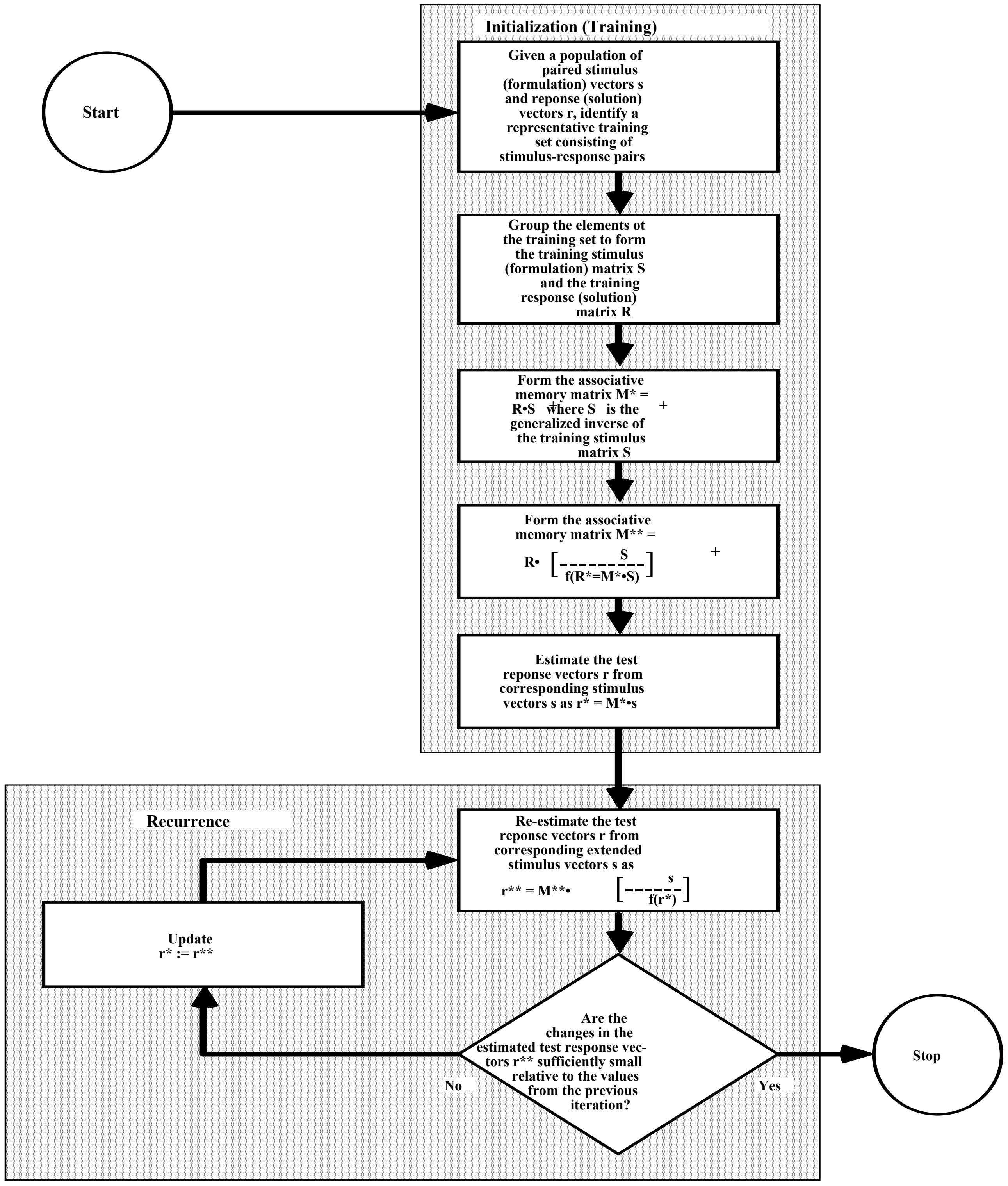
Initialization (Training)
Start
Given a population of paired stimulus (formulation) vectors s and reponse (solution) vectors r, identify a representative training set consisting of stimulus-response pairs
Group the elements ot the training set to form the training stimulus (formulation) matrix S and the training response (solution) matrix R
Form the associative memory matrix M* = R•S+ where S+ is the generalized inverse of the training stimulus matrix S
Form the associative memory matrix M** = R• [S / f(R*=M*•S)]+
Estimate the test reponse vectors r from corresponding stimulus vectors s as r* = M*•s
Recurrence
Re-estimate the test reponse vectors r from corresponding extended stimulus vectors s as r** = M**• [s / f(r*)]
Update r* := r**
Are the changes in the estimated test response vectors r** sufficiently small relative to the values from the previous iteration?
No
Yes
Stop

Figure 5: Algorithm for Computing a Recurrent Associative Memory Matrix.

4.2 Nonlinear Examples

The elements of the 23 training response vectors were subjected to three nonlinear transformations:

- quadratic expansion: $X_{11}^2, X_{11}•X_{12}, \ldots, X_{23}•X_{22}, X_{23}^2$;
- exponential: $\exp(X_{ij})$; and
- hyperbolic tangent: $\tanh(X_{ij}) = [\exp(X_{ij}) - \exp(-X_{ij})] / [\exp(X_{ij}) + \exp(-X_{ij})]$.

The first two transformations are simple, obvious candidates for experimentation. The hyperbolic tangent is often used as an activation function in neural networks. In each case, the training stimulus vectors are extended by appending the transformed response vectors. These separate modifications of the matrix **S** produce three recurrent associative memory matrices $\mathbf{M}^{**}_{\text{quadratic}}$, $\mathbf{M}^{**}_{\text{exponential}}$, and $\mathbf{M}^{**}_{\text{hyperbolic tangent}}$, of dimensions 7x26, 7x11, and 7x11, respectively.

In all three cases, a recurrent extension of the stimulus vector substantially improves the performance of the associative memory matrix. Only two iterations were required in each case, with the second iteration providing virtually no improvement. The hyperbolic tangent transformation defines the best case. Scatterplots of the estimated optimal flows obtained at iteration 1 versus the true optimal flows appear in Figure 6. These indicate considerable improvement relative to the

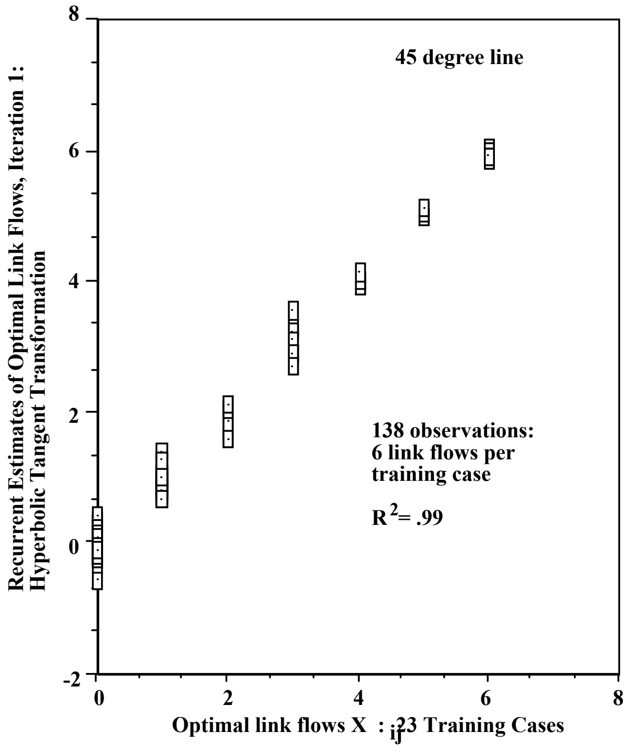

Figure 6: Recurrent Estimates of Optimal Link Flows Obtained at Iteration 1 Vs. Optimal Link Flows: Twenty-Three Training Cases, Hyperbolic Tangent Transformation

nonrecurrent case. At iteration 1, the variance in the optimal link flows accounts for 99.0 percent of the variance observed in the training estimates. There is very little unexplained variance left for iteration 2 to attack.

Rounding results are summarized in Table 2. Prior to rounding, the best case (hyperbolic tangent) mean square error measure for the terms X_{ij} is reduced by 99.46 % relative to the nonrecurrent case, dropping from 3.278 to 0.018. The optimal solution is identified by rounding in 78.61 % of the hyperbolic tangent test cases, versus only 20.23 % of the nonrecurrent test cases. One or more basic variables is rounded to zero in only 5.78 % of the hyperbolic tangent test cases, versus 17.34 % of the nonrecurrent test cases.

Table 2: Rounding Results: Estimated Solutions $\mathbf{M}^{**}\bullet\mathbf{s}_k = \mathbf{r}_k^{**}$ vs. True Solutions $\mathbf{r}_k$

Rounding Identifies the Optimal Solution	Rounding Identifies the Optimal Basis	Rounding Produces Positive Nonbasic Variables	Rounding Produces Zero-Valued Basic Variables
Quadratic			
23 Training Vectors			
21	0	2	0
173 Test Vectors			
107	17	39	10
Exponential			
23 Training Vectors			
14	2	7	0
173 Test Vectors			
88	8	64	13
Hyperbolic Tangent			
23 Training Vectors			
22	1	0	0
173 Test Vectors			
136	10	17	10

It is surprising that the hyperbolic tangent transformation identifies optimal test solutions and bases more frequently than the quadratic transformation, because $\mathbf{M}^{**}_{\text{quadratic}}$ has more than twice the elements of $\mathbf{M}^{**}_{\text{exponential}}$ or $\mathbf{M}^{**}_{\text{hyperbolic tangent}}$. Extending the stimulus vectors by appending linear transformations of the estimated response vectors provides some additional context for these results, bu produced no improvements relative to the nonrecurrent case. Consequently, we conclude that these results include more improvement than can be explained by the increased dimensionality of the matrices $\mathbf{M}^{**}$. The nonlinear transformation of the estimated response vector is making a useful contribution that extends beyond mere dimensionality.

5. Conclusions and Extensions

This approach requires much additional exploration, including the generation of more general results. It is surprising that such a simple procedure performs so well. Executing the transportation simplex algorithm is computationally less intensive than the general simplex procedure used to solve more difficult linear programs. For one of these more difficult problems, a recurrent associative memory matrix might logically be used generate a good infeasible or feasible solution, substantially reducing the number of iterations required to complete phases 1 or 2 of the simplex algorithm. We have not combinatorialized a comparison between the transportation simplex algorithm and the matrix operation $\mathbf{M}^{**}\bullet\mathbf{s}_k$, but conclude on inspection that the computational savings is considerable. The crucial point is that the recurrent associative memory matrix $\mathbf{M}^{**}$ successfully encodes the nonlinear mapping of linear programming inputs to linear programming outputs.

This research effort raises more questions than it answers. For example:

- Should information about the dual formulation be included in the stimulus and response vectors? Because linear programs have simple duals, we expect that training cases defined by manipulating coefficients in the objective function would perform as well as training cases defined by manipulating the right hand side.
- Can noisy stimulus vectors be used? If noise is present in the test stimuli, should it appear in training vectors?
- Can equally good approximate solutions be generated for more complicated linear programs?
- Can different objective functions be accommodated in the same training set?
- Can the information in training cases be reliably extrapolated? How accurate is the response vector estimated for a new stimulus vector if that stimulus includes values outside the range of the training cases?
- Can good heuristic solutions to nonlinear programming problems be generated? Probably so. Neural networks have been used to approximate the solutions to nonlinear optimization problems. For example, what would be the quality of estimates produced if congestion accrued on this transportation network, and link costs were increasing functions of link flows?

We fully expected that a more sophisticated neural network procedure would be needed to obtain estimates as promising as the outputs provided by the recurrent associative memory matrix.

However, the transportation network used in this exercise is very simple. A larger-scale or more complicated formulation might well be better handled by a Rumelhart network than by the procedure described here. Research is underway at the University of Southern California to better define the class of constrained optimization problems for which associative memory and more sophisticated neural network procedures provide a degree of utility.

REFERENCES

1. Dorfman, R., P. Samuelson, and R. Solow, *Linear Programming and Economic Analysis*, McGraw-Hill Book Co., New York, New York, 1958.

2. Beckmann, M., McGuire, C., and Winsten, C., *Studies in the Economics of Transportation*, Yale University Press, New Haven, Connecticut, 1956.

3. Koopmans, T., and Beckmann, M., *Assignment Problems and the Location of Economic Activities*, Econometrica, Vol. 25, pp. 53-76, 1957.

4. Hakimi, S., *Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph*, Operations Research, Vol. 12, pp. 450-459, 1964.

5. Dantzig, G., *Linear Programming and Extensions*, Princeton, Princeton University Press, New Jersey, 1963.

6. Hopfield, J., and Tank, D., *'Neural' Computation of Decisions in Optimization Problems*, Biological Cybernetics, Vol. 52, pp. 141-152, 1985.

7. Kohonen, T., *Self-Organization and Associative Memory*, Third edition, Springer-Verlag, New York, New York, 1989.

8. Murakami, K. and Aibara, T., *An Improvement on the Moore-Penrose Generalized Inverse Associative Memory*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 17, pp. 699-706, 1987.

9. Kalaba, R., Kim, M., and Moore, J. E., *Linear Programming and Associative Memories*, forthcoming in Applied Mathematics and Computation, 1990.

10. Poggio, T., *On Optimal Nonlinear Associative Recall*, Biological Cybernetics, Vol. 19, pp. 201-209, 1975.

11. Kalaba, R., Lichtenstein, Z., Simchnoy, T., and Tesfatsion, L., *Linear and Nonlinear Associative Memories for Parameter Estimation*, forthcoming in IEEE Transactions on Systems, Man, and Cybernetics, 1990.

12. Rummelhart, D., Hinton, G. E., and Williams, R. J., *Learning Internal Representations by Error Propagation*, Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, Massachusetts, 1986, pp. 318-362.

---

13. Rummelhart, D., Hinton, G. E., and Williams, R. J., *Learning Representations by Back-Propagating Errors*, Nature, Vol. 323, pp. 533-536, 1986.

14. Wasserman, P.D., *Combined Back Propagation/Cauchy Machine*, Proceedings of the International Neural Network Society, Pergamon Press, New York, New York, 1988.

15. Wasserman, P. D., *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, New York, New York, 1989.

16. Kosko, B., *Bidirectional Associative Memories*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, pp. 49-60, 1988.

17. Cohen, M. A., and Grossberg, S. G., *Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks,* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 3, pp. 815-826, 1983.

APPENDIX

Twenty-three Stimulus Response Pairs

-----------------------Stimulus Vectors----------------------

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a_1	0	0	0	0	0	0	1	1
a_2	6	6	6	6	6	6	5	5
b_1	0	0	0	1	2	6	1	5
b_2	0	3	6	5	4	0	0	0
b_3	6	3	0	0	0	0	5	1

	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}
a_1	1	2	3	3	3	4	5	5
a_2	5	4	3	3	3	2	1	1
b_1	6	0	1	3	3	1	0	1
b_2	0	3	3	1	2	4	0	2
b_3	0	3	2	2	1	1	6	3

	s_{17}	s_{18}	s_{19}	s_{20}	s_{21}	s_{22}	s_{23}
a_1	5	6	6	6	6	6	6
a_2	1	0	0	0	0	0	0
b_1	5	0	0	0	3	4	6
b_2	0	0	4	6	3	2	0
b_3	1	6	2	0	0	0	0

-----------------------Response Vectors-----------------------

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
X_{11}	0	0	0	0	0	0	0	0
X_{12}	0	0	0	0	0	0	0	0
X_{13}	0	0	0	0	0	0	1	1
X_{21}	0	0	0	1	0	6	1	5
X_{22}	0	3	6	5	2	0	0	0
X_{23}	6	3	0	0	4	0	4	0
Z	36	33	30	20	28	24	31	23

	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}
X_{11}	1	0	0	0	0	3	0	0
X_{12}	0	0	1	1	2	1	0	2
X_{13}	0	2	2	2	1	1	5	3
X_{21}	5	0	1	3	3	1	0	1
X_{22}	0	3	2	0	0	0	0	0
X_{23}	0	1	0	0	0	0	1	0
Z	21	27	22	20	19	18	21	17

	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}
X_{11}	4	0	0	0	3	4	6
X_{12}	0	0	4	6	3	2	0
X_{13}	1	6	2	0	0	0	0
X_{21}	1	0	0	0	0	0	0
X_{22}	0	0	0	0	0	0	0
X_{23}	0	0	0	0	0	0	0
Z	11	18	14	12	9	8	6