~~CBSAR-89/~~33

Engineering Physics and ~~Mathematics~~ Division

CONF-900116--3

DE89 015975

A NEURAL NETWORK FOR BOUNDED LINEAR PROGRAMMING*

J. C.Culioli
V. Protopopescu
C. Britton
N. Ericson

Submitted to International Joint Conference on Neural Networks, January 15–19, 1990, Washington, DC

DISCLAIMER

* Research supported in part by the U.S. Air Force Wright Aeronautical Laboratory under DOE Interagency Agreement, DOE-1570-1579-A1, and U.S. Department of Energy, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

A Neural Network for Explicitly Bounded Linear Programming

Jean-Christophe Culioli and Vladimir Protopopescu
Engineering Physics and Mathematics Division
Charles L. Britton, Jr., and Milton N. Ericson
Instrumentation and Control Division
Oak Ridge National Laboratory, Oak Ridge, TN 37831-6364

Abstract: The purpose of this paper is to describe a neural network implementation of an algorithm recently designed at ORNL [1] to solve the Transportation and the Assignment Problems, and, more generally, any explicitly bounded linear program.

1. Introduction. In a companion paper [2], we study two general neural network models for Linear Programming. Here, we introduce a different model. This model applies only to explicitly bounded linear problems, i. e. problems for which a priori bounds are available for the optimization variables. In particular, it is well suited to the Transportation and the Assignment Problems (TP and AP). Due to the very structure of the explicitly bounded linear problems, the architecture complexity of the network is much simpler. For example, a $K \times L$ AP will require only $K + L$ neurons and $K \times L$ connections, instead of $K \times L + K + L$ neurons and $(K \times L)(K + L)$ connections for the Primal-Dual model studied in [2] or for the circuit proposed by Hopfield and Tank in [3]. The neural network proposed here could also be used to solve the Analog Decoding problem proposed in [4]. In the next Section, we will briefly describe the TP and AP models. In Section 3, we present a network that implements a solution of a slightly more general problem, via a parameterization of the primal variables with respect to the dual variables. We also discuss a possible implementation. Some simulations results are given in Section 4.

2. Two Examples of Explicitly Bounded Linear Problems. The TP can be formulated in the following way: we have to ship some goods from k different sources with stocks S_i, $i = 1, 2, ...K$, to L destinations with associated demands D_j, $j = 1, 2, ..., L$. A transportation cost $c_{ij}x_{ij}$ is associated with the shipment of the (positive) quantity x_{ij} from the source S_i to the destination D_j. One assumes that there is no loss during the process, i.e. for every source i, $\sum_j x_{ij} = S_i$, and also that the demand is met for every destination, that is $\sum_i x_{ij} = D_j$. This defines a "balanced" TP: $\sum_{i,j} x_{ij} = \sum_j D_j = \sum_i S_i$. It is possible, at the price of adding shadow sources or destinations, to transform any unbalanced problem into a balanced one. The problem of minimizing the cost of the transportation leads to the linear program:

$$(1) \qquad \min \sum_{i,j} c_{ij}x_{ij}, \quad subject\ to\ x_{ij} \geq 0, \ \sum_j x_{ij} = S_i, \ \sum_i x_{ij} = D_j$$

The AP has the same mathematical formulation as the TP, except that each stock S_i is equal to 1 and each demand D_j is also equal to 1. A typical application is to assign K jobs to L machines, with operation costs $[c_{ij}]$. As noted before, one can assume, without loss of generality that $K = L$. The AP can be viewed as a combinatorial (0,1)-programming problem. However, it has been shown [5] that it can be expressed as a continuous linear program with $0 \leq x_{ij} \leq 1$. There exist several algorithms of complexity $O(K^3)$ dedicated to solving both problems (see for example [6,7]). From their above formulation, one can notice that explicit bounds on the optimization variables are available ($0 \leq x_{ij} \leq \min\{S_i, D_j\} =: X_{ij}$). We now introduce a generic problem that is somewhat more general that the TP and the AP, but preserves their fundamental properties. We seek for the solution of

$$(2) \qquad \min < c, x > \quad subject\ to \quad Ax = b, \quad x \geq 0,$$

where c and x are vectors in R^n, b is a vector in R^m, and A is an $m \times n$ matrix, with $m \leq n$. The brackets $< ., . >$ denote the scalar product in R^n. We assume that the problem (2) has a bounded solution x^* and, for the purpose of the forthcoming derivations, that the rank of A is m. We also assume that the entries of A and the entries of b are positive. This implies that one can compute an explicit bound X^{max} for the

variable x. One choice is X^{max} with entries $X_i^{max} = \frac{\min_j b_j}{\max_j \{a_{ji} \mid a_{ji} \neq 0\}}$. Note that if $X_i^{max} = 0$, this implies immediately that $x_i^* = 0$, and we can remore this variable from the original problem. We thus assume in the following that all components of X^{max} are strictly positive.
The dual problem associated with (2) is [8]

$$\max < b, p > \quad subject\ to \quad A^T p \leq c, \tag{3}$$

where A^T denotes the transpose of matrix A and the scalar product in R^m is also denoted by $< ., . >$. By definition, a vector x such that $Ax = b, \quad x \geq 0$ or a vector p such that $A^T \hat{p} \leq c$ will be called *admissible*.
The fundamental result of duality is (for a proof, see [8]):
Proposition 1. *If $\hat{x}$ is admissible for (2) and $\hat{p}$ is admissible for (3) then the duality gap $\delta :=< c, \hat{x} > - < b, \hat{p} >$ is positive. If $\delta = 0$, then $\hat{x}$ is a solution of (2), and $\hat{p}$ is a solution of (3).*

3. The Parameterized Neural Network Model. To solve problem (2), we propose to parameterize the primal variables x in the following way

$$x = X^{max} \bullet g_\lambda(c - A^T p), \quad g_\lambda(y) := \frac{2}{1 + e^{\lambda y}}, \tag{4}$$

where the function g_λ is applied componentwise on the vector $c - A^T p$, and the operation "$\bullet$" denotes the Kronecker product of vectors, that is $(y_1, y_2, ..., y_n) \bullet (z_1, z_2, ..., z_n) := (y_1 z_1, y_2 z_2, ..., y_n z_n)$
With this parameterization, we wish to solve in p the equation $Ax(p) = b$ which now writes $AX^{max} \bullet g_\lambda(c - A^T p) = b$. To do that, we consider the variables p as input-states of neurons with output states x (the implementation will be clarified in the next Section) and assume the following dynamics:

$$\frac{dp}{dt} = -(Ax(p) - b). \tag{5}$$

The sigmoid function g_λ has two useful properties that we shall take advantage of in the future derivations. Namely,
$(i) \forall \lambda > 0, \ \forall y, \quad g'_\lambda(y) = -\frac{\lambda}{2} g_\lambda(y) g_\lambda(-y) \ < 0,$
$(ii) \forall \lambda > 0, \forall y \geq 0, \quad y\, g_\lambda(y) \leq \frac{2}{\lambda e}.$
We now address the convergence of the network.
Proposition 2. *If p in system (4-5) is bounded, then the network converges to stable states $(\bar{x}, \bar{p})$ which are admissible solutions of (2) and (3).*

Proof: We introduce the Lyapunov functional $E = \frac{1}{2}||Ax - b||^2$. The functional E is bounded below (it is positive) and above, because x given by (4) is bounded. We study the variation of E along the trajectories of (5). We have $\frac{dE}{dt} =< A\frac{dx}{dt}, Ax - b >=< \frac{dx}{dt}, A^T(Ax - b) >= - < \frac{dx}{dt}, A^T \frac{dp}{dt} >$. From (4), we get $\frac{dx}{dt} = \frac{\lambda}{2} X^{max} \bullet A^T \frac{dp}{dt} \bullet g(c - A^T p) \bullet g(A^T p - c)$, which leads to

$$\frac{dE}{dt} = -\frac{\lambda}{2} < X^{max} \bullet g_\lambda(c - A^T p) \bullet g_\lambda(A^T p - c) \bullet A^T \frac{dp}{dt}, A^T \frac{dp}{dt} > .$$

We can rewrite this as $\frac{dE}{dt} = -\frac{\lambda}{2} < DA^T \frac{dp}{dt}, A^T \frac{dp}{dt} > \leq 0$ with $D(t) := diag(X^{max} \bullet g_\lambda(c - A^T p) \bullet g_\lambda(A^T p - c))$, a strictly positive diagonal matrix. Thus E is decreasing along the trajectories of (5). If p is bounded, then each entry of the vector $g_\lambda(c - A^T p) \bullet g_\lambda(A^T p - c)$ is bounded below by a strictly positive constant. The matrix $\tilde{A}(t) := ADA^T$ is, at any time t, an $m \times m$ positive definite matrix, with $||\tilde{A}(t)|| \geq a > 0$, which implies $\frac{dE}{dt} \leq -\lambda a E$. Then E converges to 0 and the trajectories of p and x converge to fixed points $\bar{p}$ and $\bar{x}$ such that $A\bar{x} = b, \quad \bar{x} = X^{max} \bullet g_\lambda(c - A^T \bar{p})$. Also, by construction of X^{max}, and due to the factor 2 in the definition of g_λ, the satisfaction of the constraint $A(X^{max} \bullet g_\lambda(c - A^T \bar{p})) = b$ implies that $c - A^T \bar{p}$ is positive. ∎

Remarks.
1. Although we had to assume a "boundedness hypothesis" for p in the theoretical derivation, it appears that in practice, this condition is not very restrictive. It is always possible to limit the iterates of p in a ball of radius R in a computer implementation (which, for R large enough does not perturb the system), but it proved unnecessary.
2. The assumption $rank(A) = m$, needed for the proof of convergence (strict positivity of $\tilde{A}(t)$), can also be relaxed in the numerical tests. In particular, in the case of the TP or the AP, $m = K + L$ but the rank of A is $K + L - 1$. This fact did not seem to alter the simulations either.

Now that we have obtained admissible solutions for (2) and (3), we need to evaluate their associated duality gap δ.

Proposition 3. *The duality gap associated with $\bar{x}$ and $\bar{p}$ is positive and bounded above by $\frac{M}{\lambda}$, where M is a positive constant which depends on the data A and b.*
Proof: The duality gap is positive since $\bar{x}$ and $\bar{p}$ are admissible. We have

$$\delta =< c, \bar{x} > - < b, \bar{p} >=< c - A^T\bar{p}, \bar{x} >=< c - A^T\bar{p}, X^{max} \bullet g_\lambda(c - A^T\bar{p}) > .$$

By using property (ii) and the Cauchy-Schwarz inequality, we conclude $0 \leq \delta \leq \frac{2\sqrt{n}||X^{max}||}{\lambda e}$. ∎

In order to do some comparison with "standard implementations", and address the complexity of implementation, we define the vector $z := c - A^T p$. With this notation, we can write

$$x = g_\lambda(z), \quad \frac{dz}{dt} = A^T(Ax - b). \tag{6}$$

System (6) defines a neural network comprising n neurons with input states z, output states x, activation functions g_λ and thresholds $-A^T b$. Neurons (z_{i_1}, x_{i_1}) and (z_{i_2}, x_{i_2}) are connected with a connection strength equal to $-\sum_j A_{ji_1}A_{ji_2}$. In conclusion, the only difference between the network (z, x) and Hopfield and Tank's network (u, V) is the absence of a time constant τ. However, it is not necessary to consider so many neurons and connections. The system (4-5) is naturally expressed as

$$\frac{dp}{dt} = -(A(X^{max} \bullet g_\lambda(c - A^T p)) - b), \tag{7}$$

for which we are led to an implementation whith only m neurons. In the case of the TP or the AP, the implementation is even more simplified. In both cases, $n = K \times L >> m = K + L$. Also, due to the matricial structure of these problems, we can denote the vector x by x_{ij} (with $x_{ij}^{max} = 1$), and associate dual variables p_i and q_j to the rows and columns of the constraints equations ($p_i \leftrightarrow \sum_{j=1}^{L} x_{ij} = 1$ and $q_j \leftrightarrow \sum_{i=1}^{K} x_{ij} = 1$) respectively. With this notation, the system (7) reduces to

$$\frac{dp_i}{dt} = -(\sum_{j=1}^{j=l} g_\lambda(-c_{ij} + p_i + q_j) - 1), \quad \frac{dq_j}{dt} = -(\sum_{i=1}^{i=k} g_\lambda(-c_{ij} + p_i + q_j) - 1). \tag{8}$$

The corresponding neurons p_i and q_j are somewhat different from the "standard neurons" of Hopfield and Tank. They include some "feedback" and a whole vector of internal thresholds (a row or a column of the cost matrix $[c_{ij}]$) in the activation function. Their interconnection is however very simple: each neuron p_i is connected to itself and to all the neurons q_j. The same is true for the neurons q_j. The connection strength are equal to -1 and the external thresholds are all equal to -1.

4. Numerical Application. We report the test of the numerical simulation of (8) for Assignment Problems with $K = L$ rangint from 10 to 100. Note that this corresponds to $10^2 \leq n \leq 10^4$ and $20 \leq m \leq 200$, i. e. fairly large problems. The entries of the matrix $[c_{ij}]$ were generated using a uniform distribution law on $[0, 1]$. The simulation of equation (8) was performed with a step size $\epsilon = \frac{1}{\lambda}$. The

parameter λ was taken equal to 10^3. We had the following results (compare the duality gap with an optimal cost of approximately 1):

$K = L$	10	20	30	40	50	75	100
δ	10^{-7}	$2\,10^{-4}$	10^{-3}	$2\,10^{-3}$	$7\,10^{-3}$	$2\,10^{-2}$	$3\,10^{-2}$
# iterations	240	170	100	90	110	70	60

One notices that the number of iterations needed to reach a stable state is not increasing with the number of variables. On the other hand, the duality gap deteriorates with the dimensions, as predicted by the above analysis. We have also simulated the network with $\lambda = 10^4$ (and $\epsilon = 10^{-4}$). We noticed the following improvement of the duality gap:

$K = L$	30	50	75	100
δ	10^{-5}	10^{-4}	10^{-4}	$2\,10^{-4}$
# iterations	820	960	490	460

5. Conclusions. We have presented a neural network model for solving explicitly bounded linear programs. Its low architectural complexity is due to the parameterization of the primal variables (x) with respect to the dual variables (p) which applies very well to matricial problems like the TP and the AP. The theoretical work presented here and many computer simulations seem to prove its applicability. We are now in the process of designing an analog circuit implementation.

Acknowledgements

This research was partially sponsored by the U.S. Air Force Wright Aeronautical Laboratory under DOE Interagency Agreement, DOE-1570-1579-A1, by the Office of Basic Energy Sciences and the Exploratory Studies Program of Oak Ridge National Laboratory, U. S. Department of Energy, under contract # DE-AC05-84OR21400 with Martin Marietta Energy Systems, by DARPA under contract # 1868-A037-A1, and by an ORNL Postgraduate Research Appointment administered by Oak Ridge Associated Universities.

References

[1] J.-C. Culioli and V. Protopopescu, "An Algorithm for Linear Programming That is Easy to Implement", Applied Mathematics Letters, Vol. 2, N. 2, pp.125-129, 1989.

[2] J.-C. Culioli, V. Protopopescu, C. Britton, and N. Ericson, " Neural Networks Models for Linear Programming", this Conference Proceedings.

[3] J. J. Hopfield , and D. W. Tank, "Simple "Neural" Optimization Networks : an A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Trans. on Circuits and Systems, Vol. CAS-33, N. 5, May 1986.

[4] J.-C. Platt and J. J. Hopfield, "Analog decoding Using Neural Networks", Proc. of the AIP Conference, Snowbird, UT 1986.

[5] J. von Neumann, "A Certain Zero-Sum Two-Person Game Equivalent to the Optimal Assignment Problem", in H. J. Kuhn and A. W. Tucker (eds), Contribution to the Theory of Games, Vol. 2, Annals of Mathematics Study N. 28, Princeton University Press, 1953, pp. 12-15.

[6] N. Christophides, Graph Theory. An Algorithmic Approach., Academic Press, London, 1975.

[7] F. Bourgeois and J.-C. LaSalle,"An Extension of the Munkres Algorithm for the Assignment Problems to Rectangular Matrices", Communications of the ACM, algorithm 415, December 1971, Vol. 14.

[8] G. B. Dantzig, Linear Programming and Extensions, Princeton University Press, 1964.

[9] J.-C. Culioli and V. Protopopescu, "Bifurcating Optimization Algorithms and their Possible Application", ORNL/TM-10976, Nov. 1988.