1

gpml-ann-01-002 22 December 1989

CDRL sequence no. A002
SCIENTIFIC & TECHNICAL REPORTS SUMMARY
Final Report

Procuring Activity Designated Order No.:
KE-9-C4236-01-000

Name of Contractor:
General Purpose Machines Laboratory

Contract No.:
DASG60-89-C-0078

Effective Date of Contract:
5/22/89

Expiration Date of Contract:
11/22/89

Reporting Period:
5/22/89 - 11/22/89

Principal Investigator
Jurn Sun Leung, 714/856-3327

Project Scientist:
Yel-Chiang Wu, 714/856-3327

Title:
AN INTELLIGENT JOB DISPATCHER FOR COMPUTER SYSTEMS

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the policies, either expressed or implied, of the Government.

SPONSORED BY
SDIO Innovative Science and Technology Office

MANAGED BY
U.S. Army Strategic Defense Command

AN INTELLIGENT JOB DISPATCHER FOR COMPUTER SYSTEMS
FINAL REPORT

ABSTRACT

The Intelligent Job Dispatcher has two networks, the LMN-net and the information gathering network.

The main result of this effort is the architecture of the LMN-net, which optimizes the assignment of N jobs to M processors in a window of L time steps. Input to this network are information on job loads, processor capacities, and inter-job dependence. Output of this network are assignment matrices. This network is a modified Hopfield network in terms of neuron connections, but its energy function contains no adjustable free parameter. The Two-Phased Optimization algorithm exploits the method of gradient descend in the "high temperature" phase. When energy fluctuation is detected as a consequence of the combined effect of smallness of network temperature and vertex hopping network state transition, a "quenched" phase is initiated. In this phase, the natural "hopping-increase" in energy is exploited in the Free Mode to rescue the system from being trapped in infeasible local minima. The Free Mode alternated with the Constrained Mode which confines the system to approach a local minimum through feasible vertex points only. Simulation of application of the Two-Phased algorithm to the LMN-net for (L,M,N)=(5,3,10) and (5,6,20) are presented in this report.

The purpose of the information gathering network is to assimilate the information of the input and output correlation of the LMN-net and finally replace and generalize its work in quasi-stable situations. The information gathering network achieves this task through pattern association and has a counterpropagation architecture with four Kohonen networks at the front end and a Grossberg layer on the backside. Both the coding scheme for communicating with the input/output neurons and the size and update procedure of this network need to be determined through further experiment.

CONTENT

1. Introduction

The design of the Intelligent Job Dispatcher (IJD) is motivated by the requirement for future embedded computer systems on SDI assets to optimize job assignment among distributed processors and to do so efficiently. The IJD has a two loop structure, with one loop performing the optimization and the other gathering the optimization information and ultimately capturing and reproducing this information through associative memory [Fig 1]. We envision the optimization process to be carried out by a modified Hopfield network, and the information gathering done through a counterpropagation Kohonen-Grossberg network.

Most of the research effort is concentrated on the optimization loop. The LMN-net is designed for the optimization of assigning N jobs to M processors within a time window of L time steps. This network, in contrast with other "conventional" Hopfield network, contains no arbitrary adjustable free parameters. A two-phase update algorithm to propagate the network to a feasible solution that satisfied all constraints is worked out and tested through simulations. This algorithm is robust and insensitive to the initial conditions.

The information gathering loop will be performing a pattern association task. Both input and output of the neural network in this loop are a collection of matrices which are best coded as multidimensional patterns.

2. Optimization Loop

2.1 LMN-Net Design

The purpose of the LMN-net is to produce L scheduling matrices, one for each time step within the optimization time window. There are M processors, characterized by the capacity vector C={c(j),j=1,2,..,M}, and N jobs, characterized by the job load vector W={w(k),k=1,2,...,N}. Job dependence is described by matrix D={d(i,j);i,j=1,2,...,N}. d(i,j)=1 implies that job j cannot start before completion of job i. N is variable since new job may be added to the task and old jobs get completed. The number of time steps, L, is dynamically determined either by the longest dependence tree or the time required to process the total load, given the total capacity of the system, whichever is larger. At any instant, the optimization network has n=(L x M x N) neurons [Fig 2].

The energy function has four terms. They reflect, respectively, the constraint for balanced utility of the heterogeneous processors, the requirement to assign sufficient computing resource to each job, the constraints of inter-job dependence, and the requirement that the assignment matrices are genuine permutation matrices.

2.2 Problems with the Conventional Hopfield Network

The paper of Hopfield and Tank [1] started a stampede to apply neural networks to combinatorial optimization problems and the celebrated Travelling Salesman Problem was duplicated repeatedly in laboratories with computer systems of even very moderate capacity. Recently, there is a general disillusion with the conventional Hopfield network, mainly as a result of difficulties in applying such network to nontrivial problem. In retrospect, it is surprising that any but the most restricted form of Hopfield network should converge at all.

For nontrivial problems, the landscape of the energy function over the solution space is non-convex, is highly complex with a plentitude of local minima and maxima. Being trapped in a local minimum on the march along a gradient descent trajectory is only one of the perils of such algorithms. In practice we can accept suboptimal solution that satisfies all constraints and need not always seek the global minimum. The conventional updating scheme, however, would rarely converge for discretized problems even when the trial solution is indeed trapped in an energy valley, in contrast to problems where solutions in the continuum are sought.

The solution space is the direct product of n copies of the closed interval [0,1] of the real line, n being the total number of neurons/spins. Let us distinguish between the vertex points whose coordinates are all 0's or 1's, and the interior points which do not satisfy this condition. Consider the update scheme, $s(i;k+1) = 0.5\ (\tanh(\alpha\, s(i;k) - \frac{1}{T}\frac{\partial H}{\partial s}) + 1.0)$; here s(i;k) is the ith neuron variable at the kth update step, H is the equivalent of a Hamiltonian of the neural network if we consider it as a spin glass system, and α and T are relaxed gradually to 1.0 and 0.0 respectively. In this way it is guaranteed that only vertex points are reached at the later stage of the updating process. Most Hopfield like update strategies start the updating process at a "high T" stage where the neuron variables are not polarized, the state of the system is at an interior point and the trajectory of the network traverses a quasi-continuous path. As "cooling" proceeds, the smallness of T swings the neuron variables back and forth according to the sign of the local energy gradient and the dynamic system leaps from one vertex point to another.

As was shown in a paper by Jeffrey and Posner [2], the dicretized update along the gradient descent actually cause the total energy to increase if

$$T < \frac{1}{2}\left|\delta^2 H\right| \qquad (1)$$

and

$$\delta^2 H < 0 \qquad (2)$$

Here

$$\delta^2 H = \frac{\partial^2 H}{\partial s(i;k)^2} \qquad (3)$$

Since T is very small by construction, the first inequality is going to be satisfied by at most neurons s(i;k). This means that the simple minded Hopfield update scheme has a built-in instability.

A multitude of free parameters are introduced in most of the published Hopfield networks efforts. These represent attempts to deal with the difficulties by massaging the energy function hoping to create energy landscapes favorable to searching from some ,rescribed initial conditions. Sometimes these parameters are disguised as Lagrangian multiplier;, but are nevertheless essentially arbitrary fudge factors. For any nontrivial problems this is a form of hopelessly undirected "human preprocessing" which is more a test of luck and patience than analysis. These free parameters were so mystifyingly free and "ad hoc"!

A successful application of neural network to combinatorial optimization depend on constructing network architectures that satisfy the following criteria:

a. The architecture is derived solely from the optimization problem at hand without recourse to extraneous free parameters.

b. The network behavior is robust and not sensitive to choice of initial conditions.

c. The updating algorithm converges autonomously to physical solutions that satisfy all constraints.

d. The success of the algorithm does not depend on assumption of convexity of the energy function.

The two-phase optimization algorithm described in the next section will satisfy all these criteria.

2.3 The Two-Phased Optimization Algorithm

The Two-Phased optimization algorithm is designed to meet the criteria set forth in Section 2.2. The optimization procedure is carried out in two phases. The neuron variables are initialized randomly. An iteration consists of a complete sweep over all the neurons and the order of update within each sweep is randomized.

In the "High Temperature" phase, the inequality relation in (1) is not satisfied andx neuron variables are propagated in small steps along the sigmoid output function while the total energy decreases monotonically. As the network temperature decreases, there is a point where a neuron variable update involves large jumps on the sigmoid so that almost all changes in neuron variables are polarity changes (spin flips) between 0 and 1. Energy fluctuation sets in as occasions occur where inequality (1) holds. The algorithm then switches to the "Quenched Phase" when such energy fluctuation is detected.

The Quenched Phase propagation algorithm operates in two alternating modes. Basically, in the "Constrained" mode, only those updates that violates either condition (1) or (2) are selected. The nominal update equation is not changed, but when a spin flip is called for based on the evaluation of the partial derivatives, it is not automatically effected but is subjected to an energy test. If that spin flip results in an increase in energy, it is revoked and the algorithm proceeds to the next neuron on the random update list. In this way the system is constraint to progress to an energy minimum.

If total energy remains unchanged after several sweeps of constrained update, the trial solution (assignment matrix) is subjected to the constraint tests. These tests contain three parts: whether no processor is assigned more than once in each time step, whether the resource assigned to each job equal or exceed its load, and whether the assignment matrices are true permutation matrices. If all three tests are passed, the algorithm terminated. If however any of these three tests fails, the system enters a period of "Free" mode allowing both energy-increasing spin flips. This will allow the energy of the system to rise as it leaves the basin of the current attractor. After a convenient number of sweeps in the free mode, the algorithm switches back to the constrained mode.

Thus, the free and constrained modes are exercised alternatively as the system visits and leaves the basins of different attractors until a feasible solution is found. Some simulation results of the two-phase algorithm is presented in SEction 2.4.

2.4 Simulation Results

Figures 3 and 4 show evolution histories of the energy function of the LMN-net for two cases: (L,M,N)=(5,3,10) and (L,M,N)=(5,6,20). In both cases the High Temperature Phase terminates at about 600 iterations when monotonic decrease of the energy function is interrupted by fluctuations. In the Quenched phase, the effect of alternating Constrained and Free modes of updating is clearly seen. The system visits many local infeasible attractors before arriving at a feasible solution. Note that the suboptimal solution is not necessarily lower in energy than the "stop-over" attractors. For the 10 job case in Figure 3, the lowest energy point in the entire iteration history is the attractor around 900 iterations, but this is not a feasible solution because not all constraints are satisfied. Similarly, for the 20 job case in Figure 4, the low energy state around 1500 iterations is rejected.

The job load and processor capacity vectors for these two cases are shown in Tables 1 and 2. Figures 5 and 6 exhibit the assignment matrices obtained for the 10 job and 20 job cases respectively. They are permutation matrices, as required by the constraints.

A vital question is the scalability of this kind of algorithm, since the relevancy to SDI programs is determined by how well it deals with really large job/processor mixes. One expects that as the number of jobs increases, the dependence conditions also multiply. The best way to obtained some quantitative result is to investigate the efficiency on a set of realistic tasks. Another approach is to generate the job dependence matrix randomly. A limited suite of job/processor mixes are tested and the terminating iteration numbers are shown in Table 3. With this limited set of data, we tentatively conclude that the complexity of this algorithm grows approximately as 1.3 power of the number of neurons in the network, i.e., for n neurons,

$$\text{Time-Complexity}(n) \approx n^{1.3} .$$

3. Information Gathering Network

Input to the information gathering network is the object I={C,W,D}, a combination of two integer vectors and a binary matrix. As has already been defined above, C is the M-vector of processor capacities, W is the N-vector of job load, an D is the binary dependence matrix. Output of this network is the set of binary assignment matrices. We envision this network to be an extended version of the Counterpropagation network of Hecht-Nielson. The front end are four Kohonen networks and the backside is a Grossberg layer [Figure 7].

The training phase of the information gathering network is an autoassociative process. The inputs {C,W,D,Q} - Q being the set of assignment matrices - are applied to the four Kohonen nets, and the output from the Grossberg layer is compared with the input to derive weight adjustments.

The information network requires further extensive investigation. The correlation between D and Q is very difficult to capture. There is a "non-nearest-neighbor" nature in D, since changing a single dependence relation would cause the assignment matrix to change drastically.

4. Direction for Future Research

Because of lack of time, our development of the two loops of the IJD is not balanced. Major portion of the research effort is concentrated on understanding the LMN-net and constructing a workable algorithm for the optimization task.

The Two-Phased Optimization Algorithm is robust, insensitive to initial conditions, and only converge to feasible results by construction. Furthermore, the LMN-net avoids the bane of most Hopfield network, namely the requirement of adjustable (or, more aptly, impossible to adjust!) parameters for convergence. The LMN-net is free of free parameters. The next question on the optimization loop is the matter of computation efficacy. By its nature, the update of the neuron variables must be carried out sequentially in each sweep, else a totally different dynamics can be expected from the neuron/spin system. However, there are ingredients of parallelism that can be exploited in the updating of each neuron, e.g. the computation of the "exclusion" term for the permutation matrix constraint, and the evaluation of partial derivative of the Hamiltonian. Hardware processors can be designed to exploit these parallelism.

The information network need to be pursue beyond the conceptual stage. While both the input and output to this network appears to be complex and formidable, forms of coding could be found to deal with both the dependence matrix and the assignment matrices, because of the sparse nature of both. The efficiency of this network, or lack of it, can only be answer by further experimentation.

References

[1] J.J.Hopfield and D.W.Tank, "Neural Computation of Decision in Optimization Problems", Bio. Cybern., 52, 141, 1985

[2] W.Jeffrey and R.Rosner, "Optimization Algorithms: Simulated Annealing and Neural Network Processing, The Astrophysical Journal, 310, 473, 1986.

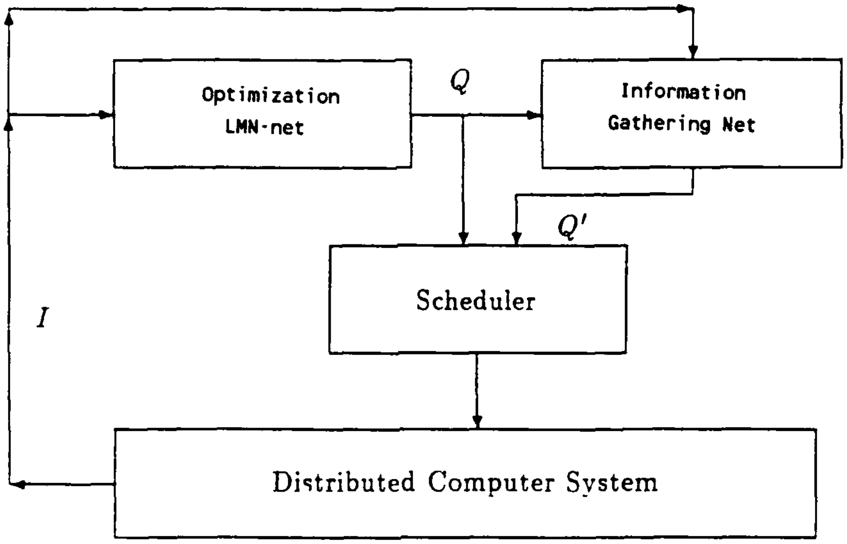

Figure 1: The Intelligent Job Dispatcher Block Diagram

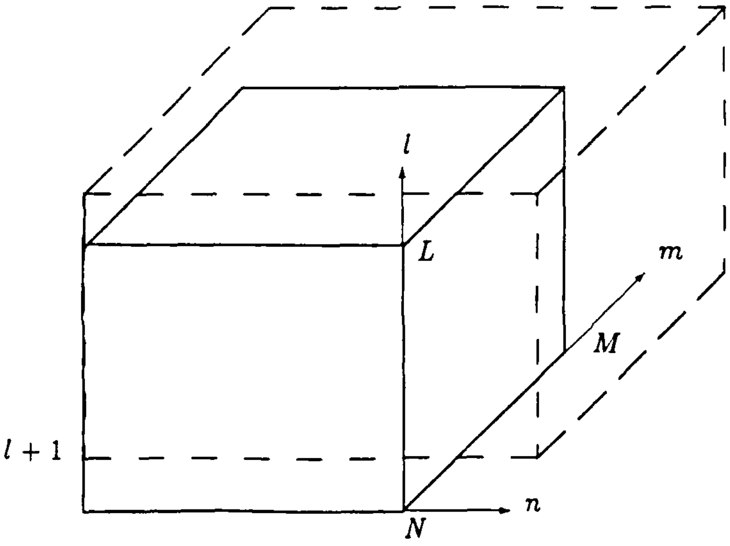

Figure 2. : Evolution of the LMN-net Optimization Volume

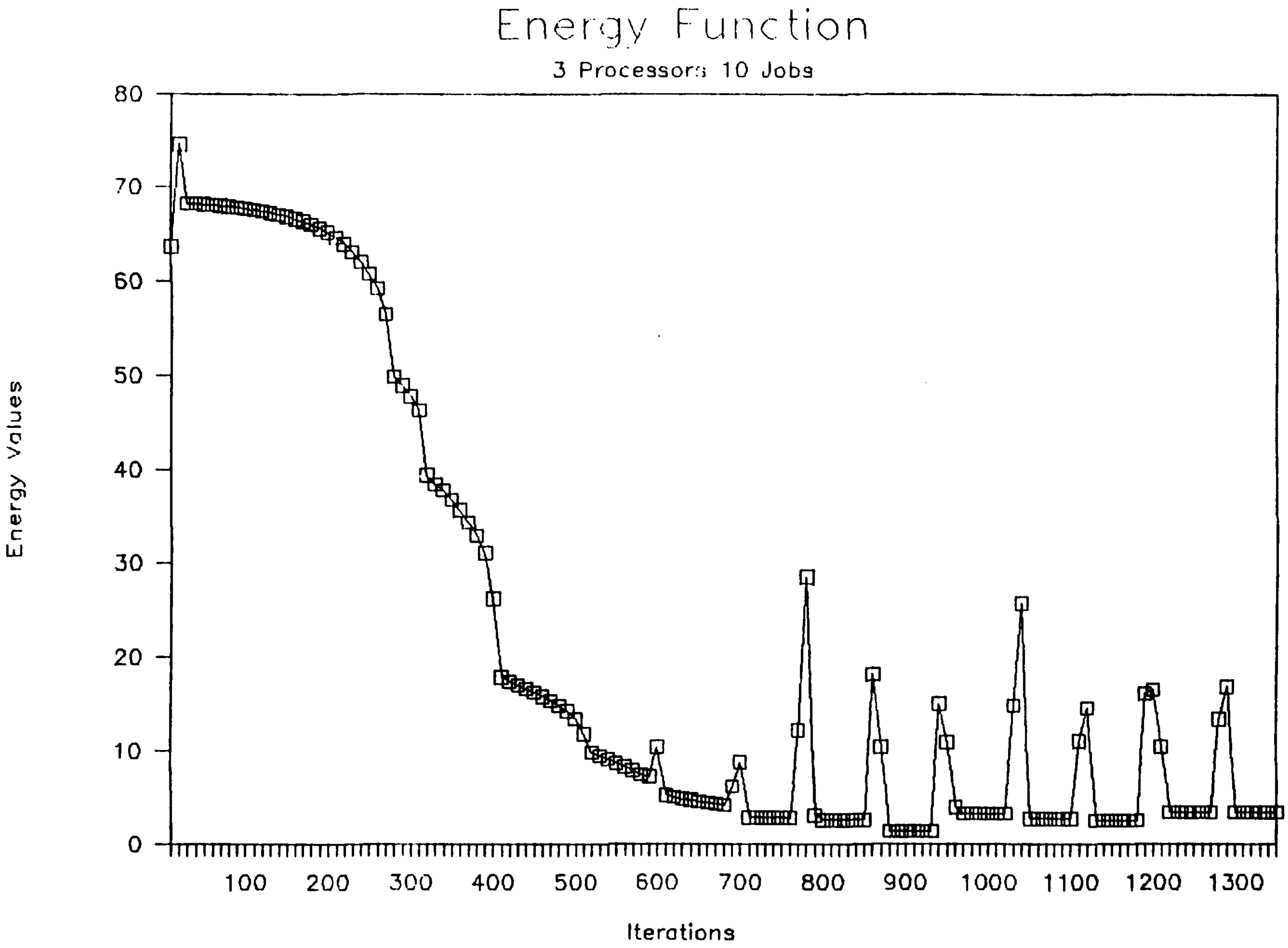

Figure 3: LMN-Net Energy Evolution
(L,M,N) = (5,3,10)

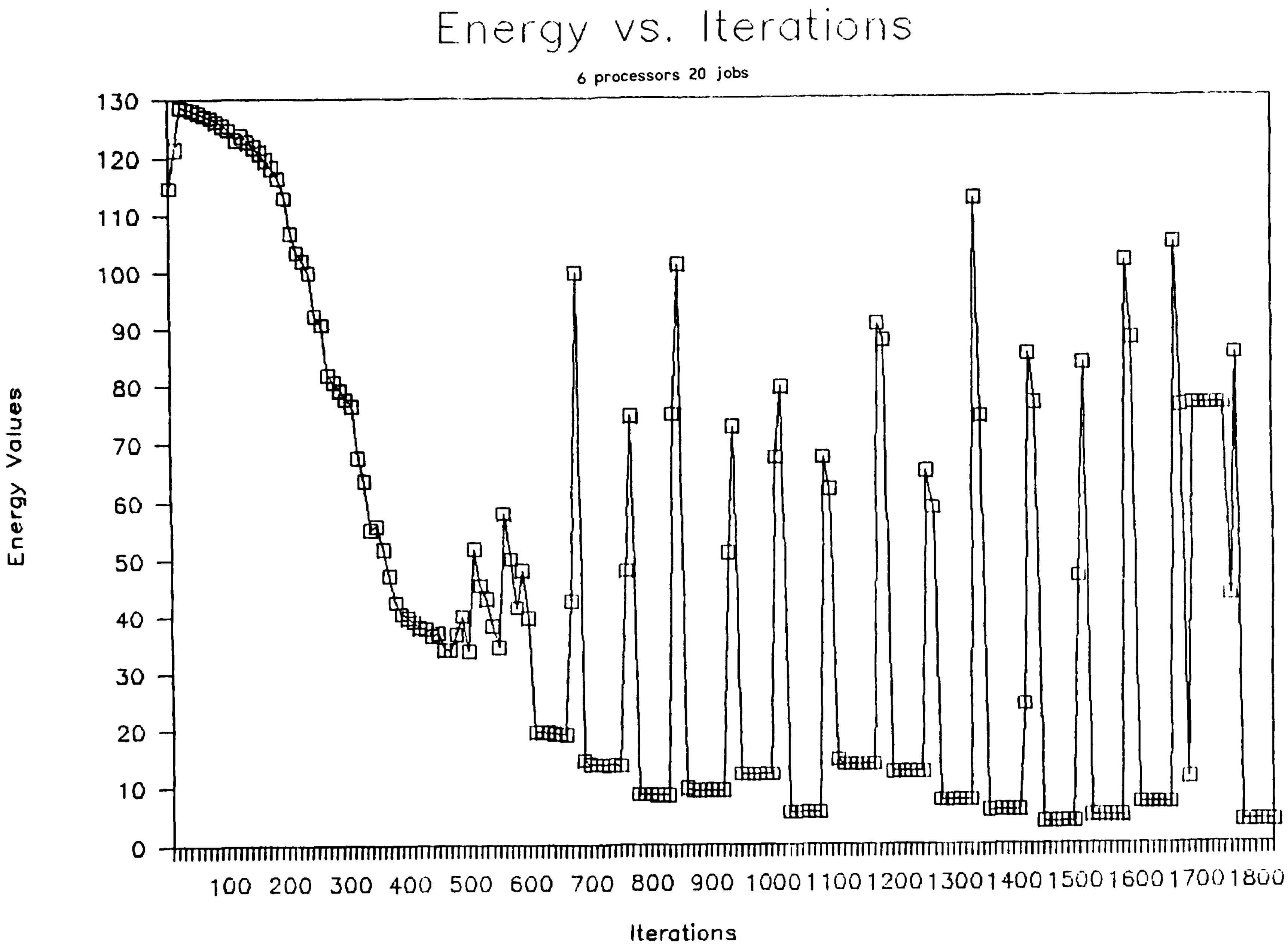

Figure 4: LMN-Net Energy Evolution
(L,M,N) = (5,6,20)

i = 1

$$\begin{array}{ll|cccccccccc} j & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ j & 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

i = 2

$$\begin{array}{ll|cccccccccc} j & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}$$

i = 3

$$\begin{array}{ll|cccccccccc} j & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ j & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ j & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

i = 4

$$\begin{array}{ll|cccccccccc} j & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ j & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

i = 5

$$\begin{array}{ll|cccccccccc} j & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ j & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 5. Assignment Matrix
(L,M,N)=(5,3,10)

i	time step
j	processor
column	job

```
i = 1
j 1  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 2  0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
j 3  0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 4  0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
j 5  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
j 6  0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
i = 2
j 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
j 2  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
j 3  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 4  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
j 5  0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 6  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
i = 3
j 1  0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 2  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
j 3  0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 4  0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
j 5  0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
j 6  0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
i = 4
j 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
j 2  0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 3  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
j 4  0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
j 5  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 6  0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
i = 5
j 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 2  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 3  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 4  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
j 5  0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
j 6  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 6. Assignment Matrix
(L,M,N)=(5,6,20)

i   time step
j   processor
column   job

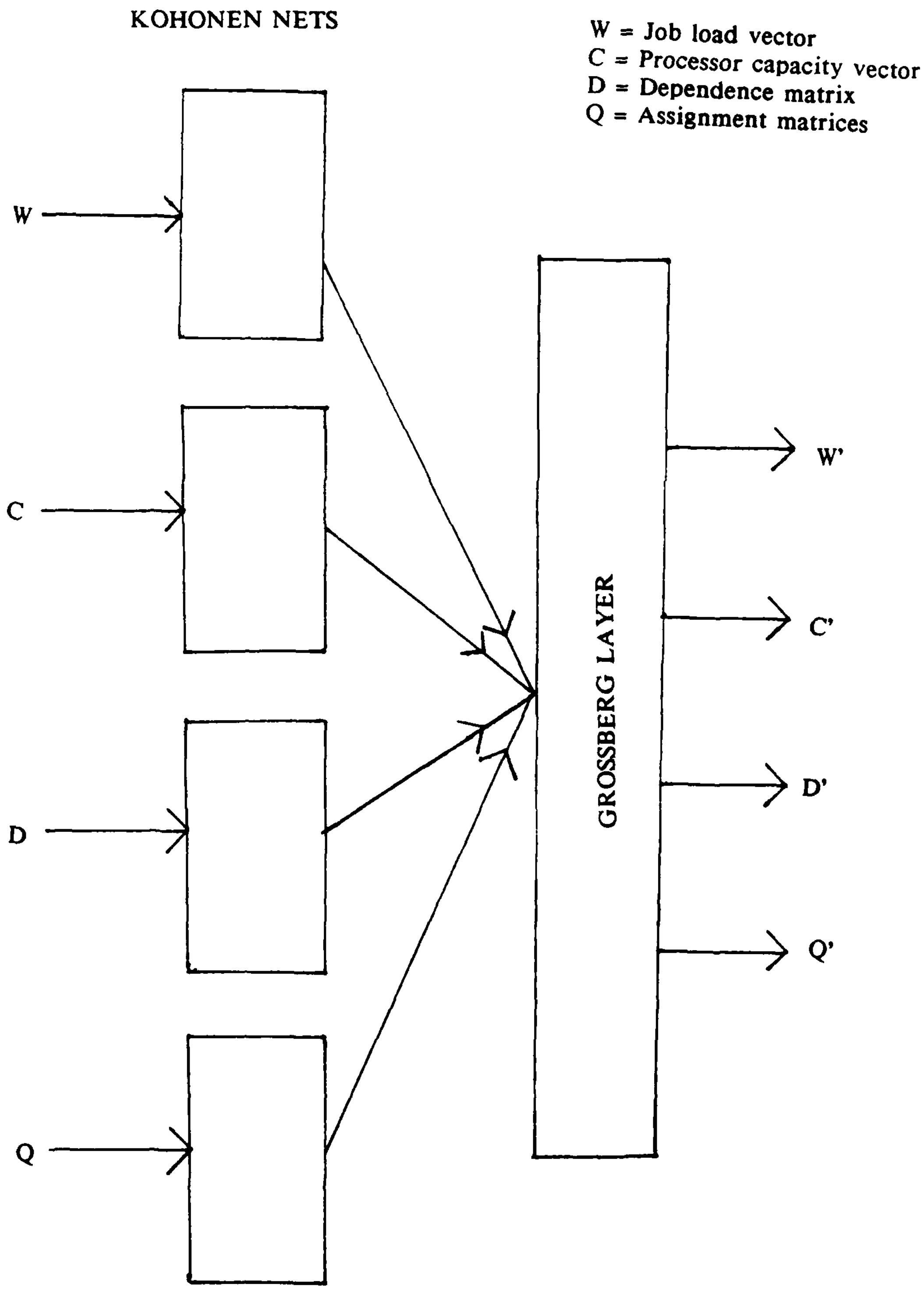

Figure 7. Information Gathering Network

TABLE 1. Job Loads & Processor Capacities
(L,M,N)=(5,3,10)

W(job loads)	1	1	2	3	1	1	1	1	1	1
C(processor capabilitie)	1	1	2							

TABLE 2. Job Loads & Processor Capabilities
(L,M,N)=(5,6,20)

W(job loads)	1	3	2	3	1	3	2	1	1	3	1	3	2	1	1	3	2	3	1	3
C(processor capabilities)	1	2	3	1	3	1														

TABLE 3. Number of Sweeps to Converge vs Neuron Number

n	(L,M,N)	Nsweep
150	(5,3,10)	1129
150	(5,3,10)	1460
150	(5,3,10)	1356
150	(5,3,10)	1437
150	(5,3,10)	1284
600	(5,3,20)	1835
600	(5,3,20)	2450