A Lagrangian Formulation of Neural Networks I: Theory and Analog Dynamics

Eric Mjolsness[1] and Willard L. Miranker[2]

[1]Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena CA 91109-8099

[2]Department of Computer Science
and Neuroengineering and Neuroscience Center
Yale University
New Haven CT 06520

and
Research Staff Member, Emeritus,
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

We expand the mathematical apparatus for relaxation networks, which conventionally consists of an objective function E and a dynamics given by a system of differential equations along whose trajectories E is diminished. Instead we (1) retain the objective function E, in a standard neural network form, as the measure of the network's computational *functionality;* (2) derive the dynamics from a Lagrangian function L which depends on both E and a measure of computational cost; and (3) tune the form of the Lagrangian according to a meta-objective $\mathcal{M}$ which may involve measuring cost and functionality over many runs of the network. The key new features are the Lagrangian, which specifies an objective function that depends on the neural network's state over all times (analogous to Lagrangians which play a similar fundamental role in physics), and its associated *greedy functional derivative* from which neural-net relaxation dynamics can be derived. It is the greedy variation which requires the dissipation critical to optimization with neural dynamics.

With these methods we are able to analyze the approximate optimality of Hopfield/Grossberg dynamics, the generic emergence of sub-problems involving learning and scheduling as aspects of relaxation-based neural computation, the integration of relaxation-based and feed-forward neural networks, and the control of computational *attention mechanisms* using priority queues, coarse-scale blocks of neurons, default-valued neurons, and other special-case optimization algorithms. Some of these applications are the subject of part II of this work.

In part II of this work we show that the combination of Lagrangian and meta-objective suffice to derive and provide an interpretation for so-called *clocked objective functions,* a notation useful for the algebraic formulation and design of ramified neural network applications. Clocked objectives thus generalize the original static objective function E and furnish a practical neural network specification language.

1 INTRODUCTION

Optimization is a prominent way to bring mathematical methods to bear on the design of neural networks. Often the connection is made [Hop84, Gro88, 11'1'85] by specifying the attractors of a

neural network's dynamics by means of a static objective function (or *objective*) to be optimized, provided that the optimization problem can be put in a standard neural-net form (which is not too restrictive a requirement [MG90]). In this way it has proven possible to design neural networks for applications in image processing [KMY86], combinatorial optimization [DW87], clustering [RGF90, BK93], particle tracking in accelerators [YHP91], object recognition [Tre91] and other applications. It is also customary (albeit limiting) to introduce a generic steepest-descent dynamics to optimize or '(relax" the objective, without further regard to computational constraints. The resulting equations of motion generally contain gradients of the static objective, but arc otherwise ad hoc slid not particularly suited to elaboration or refinement in response to varied computational constraints. We shall develop a more general approach, starting from basic principles, to formulating the dynamics of a relaxation-based neural network.

Here *we start from fundamental computational considerations which, wc hypothesise, constrain all dynamical systems that compute.* Specifically, the cost and functionality (efficacy) of a computation are fundamental to its design, and in general each must be traded off against the other in the course of optimizing that design. (Here the "design" is all the information which directly specifies the structure or configuration of the dynamical system that performs a computation.) In the context of neural computations, we will find measures of cost and functionality and combine them into *dynamical objective functions* from which one may derive the entire dynamics of a neural network. This dynamics includes not only the (fixed point) attractors but also the equations of motion governing convergence to an attractor, i.e. a mathematical model or specification of the network itself.

Our dynamical objective functions can be specialized in many ways that correspond to the wide variety of goals and constraints that may be imposed on a computation. We will also relate the dynamical objective functions to a so-called *Lagrangian functional.* Our Lagrangian is analogous to one which plays a similar and fundamental role in physics. A basic constraint which we impose on **our approach is that** such a dynamical objective function or Lagrangian is optimized in a special way, by means of *greedy algorithms* which don't look ahead in time. This constraint allows our algorithms to be implemented in physical hardware, and also allows us to derive nonconservative, irreversible dynamics which can lead to a desired fixed point. We will derive these algorithms by means of a novel *greedy variation* applied to the Lagrangian functional.

Generally we will accept the limited type of optimization that results, but sometimes we can do better by introducing another level of optimization: a *meta-optimization* problem in which the (analytic) form of the dynamic objective (the Lagrangian functional) is itself varied so as to optimize another objective function. This latter optimization may involve measuring cost and functionality over many runs of the network. 'l'his meta-optimization problem determines the choice of the exact algebraic form of the Lagrangian and hence of the computational dynamics for a whole class of applications. So for a meta-objective function, cost and functionality are measured over a class of computational problems rather than over a single instance of that class as would be the case for a Lagrangian functional. In practice the computational cost or analytic effort required to perform the rneta-optimization is to be amortized over many problem instances, One example of this approach will be a (meta-)optimality objective for Hopfield/Grossberg dynamics [Hop84, Gro88], for which we provide a proof that the associated Lagrangian is optimal in an approximate sense.

1.1 Cost and Functionality

Consider a physical system capable of nontrivial computation. More abstractly, consider a discrete, continuous or mixed dynamical system which computes, in the sense that it models a computational device or framework. Examples include a general-purpose computer equipped with suitable programs, a discrete data structure implemented by means of such a program, an individual silicon chip, or an animal brain. Such devices have detailed dynamics, often approximable as large sparsely coupled systems of ordinary differential equations, which have been designed (or evolved in the case of a brain) to serve some set of computational purposes at feasible cost. So we refer to these dynamical systems as *computational systems* and hypothesize very broadly that *fundamentally, a computational system is designed (or evolved) to optimize two things:* its *cost* and its *functionality. Functionality* means what the system can do, and *cost* means how cheaply or quickly it can do it.

For example, the design of silicon chips is largely constrained by the use of chip area and cycle time as the measures of cost, and the need to attain at least a minimal level of functionality to make the chip generally useful (e.g. to implement an adequate instruction set in a CPU chip); tradeoffs

between minimization of chip area and maximization of detailed functionality are frequent in the design process. For another example we refer to the implementation of abstract data structures such as priority queues, for which a functionality specification requires that a small set of operations (such as adding a prioritized element to a queue and removing the element with highest priority from the queue) must be supported, ancl cost is conventionally characterized by an asymptotic scaling rule for the time-cost of performing a worst-case mix of these operations on a very large queue.

For a relaxation-based neural net which is programmed or designed to optimize a static objective function $E(\mathbf{x})$ from an arbitrary starting point $\mathbf{x}_{\text{initial}}$, typical expressions for cost C and functionality F might be

$$C = \text{4-Volume of the Net} = \text{Space x Time} \tag{1}$$

and

$$F = E(\mathbf{x}_{\text{final}}) - E(\mathbf{x}_{\text{initial}}). \tag{2}$$

The space-time product is familiar in computer science as an important measure of cost, in which the Space term is a volumetric measure of hardware usage such as chip area (including on-chip wires) or memory usage, and the Time term is likewise a computational version of physical time such as the number of clock cycles required to complete a computation. (A specific volumetric measure of wiring cost for circuit implementations of neural nets has been proposed in [Mjo85].) As to functionality, the use of an objective function E is a common way to measure progress (hence functionality) in a wide variety of computational problems. For example, one can fit a piecewise-constant model to a 2-d image given by the data $\{d_{ij}\}$, segmenting it into roughly constant regions, with the objective function [KMY86]

$$\begin{aligned} E(f, s^h, s^v) &= \frac{A}{2}\sum_{ij}(f_{ij} - d_{ij})^2 + \frac{B}{2}\sum_{ij}(f_{i+1,j} - f_{ij})^2(1 - s^v_{ij}) \\ &\quad + \frac{B}{2}\sum_{ij}(f_{i,j+1} - f_{ij})^2(1 - s^h_{ij}) + \mu\sum_{ij}(s^h_{ij} + s^v_{ij}), \end{aligned} \tag{3}$$

where $f_{ij} \in \Re$ is a reconstructed version of the image, and $s^{h,v}_{ij} \in \{0, 1\}$ represent discrete decisions concerning the probable presence or absence of horizontal and vertical edges. f and s together constitute the vector x appearing in equation (2). This kind of objective has been used to derive functional neural networks for large-scale problems (10^5 neurons with 10^6 connections) as required for image-processing [RC91, KMY86].

1.2 Outline

We (a) introduce a three-level optimization framework, concentrating on Lagrangians (of a type relevant to computation) and their specialization to clocked objective functions (section 2); (b) apply the framework to derive analog circuits such as those modeled by the Hopfield/Grossberg dynamics for optimization (section 3); and (c) apply the framework to incorporate computational attention mechanisms (similar to saccading and foveatiou in biological vision) into various dynamical systems which are designed to solve optimization problems (section 2 of Part II).

Section 2 introduces the three-level optimization framework, beginning with the general form of a Lagrangiau suitable for use in attractor dynamics for optimization problems. The greedy functional derivative is defined and calculated for such Lagrangians (sections 2.1 and 2.2). The strategy used to design circuit-implementable Lagrangians is one of *refinement* (section 2.3), in which cost and functionality measures are first defined at a coarse temporal scale and then refined for use at finer time scales, down to the infinitesimal time scale suitable for dynamical systems that model analog circuits, The validity of the transformations required during refinement is ultimately specified by a meta-objective function which measures network per formance. One circuit-implementable form of Lagrangian is introduced in sections 2.2 and 2.3, though not completely derived until section 3.2, and it is illustrated by the concrete example of Hopfield/Grossberg dynamics for a region-segmentation neural network. A more general circuit-implementable form of Lagrangian, which allows network dynamics to be controlled by a repeating cycle of objective functions rather than a single objective function, is introduced in section 2.1 of Part 11.

where it is illustrated by an algorithm similar to line minimization. This type of Lagrangian gives rise to the practical *clocked objective function* and *clocked sum* notation of sections 2.1.2 and 2.1.3 of Part 11, whose theoretical justification requires all three levels of opt imization: the objective E, the Lagrangian L, and the meta-objective $\mathcal{M}$.

Section 3 is devoted to the study of circuit-level Lagrangians with continuous time dynamics and analog- valued neurons. Two novel possibilities for such Lagrangians are discussed in sections 3.1.1 and 3.1.2. In section 3.2 a simple meta-optimality criterion for a limited class of analog circuit Lagrangians is presented. Since this constrained meta-objective function $\mathcal{M}_\tau$ is a function of the fastest and slowest physical time scales in various circuits, it is invariant with respect to monotonic, coordinatewise reparameterizat ions (changes of variable) of the circuit.

In sections 3.2.1, 3.2.2, and 3.2,3 we prove Theorem 1, which asserts that the Lagrangian 1, corresponding to Hopfield/Grossberg dynamics yields a value of $\mathcal{M}_\tau[L]$ which is within a factor of two of the optimal value of *MT*. This means, roughly, that the worst-case time constant for this Lagrangian L is at most twice that of the optimal Lagrangian I,*, whatever that is. The proof exploits a sharp global opti mality result for Hopfield/Grossberg dynamics (Lemma 1 of section 3.2.2). Unlike *MT*, the optimized functional of Lemma 1 does depend on the coordinate system chosen. A number of limitations of Theorem 1 are discussed. The resulting Lagrangian for analog circuits can be generalized to clocked objective functions, as discussed in section 2.1.5 of Part II. Section 2.1.6 of Part 11 provides an instructive example: a clocked objective function which incorporates one or more general feed-forward neural networks (for which relatively efficient learning algorithms are available) inside a general relaxation neural network.

In section 2 of Part 11 we show how simple cost constraints can lead to a variety of computational *attention mechanisms* analogous to virtual memory protocols in present-day computers, and an associated Lagrangian or clocked objective function to control each attention mechanism. Examples of possible foci of attention include a subset of the n (out of N) neurons with nighest estimated improvement in functionality $|\Delta E|$, which may be tracked efficiently by means of a priority queue data structure (section 4.1 of Part II); a subset of course-scale blocks in a minimal partition of the neurons, scheduled by their estimated individual and pairwise contributions to $|\Delta E|$ (section 4.2 of Part 11); a set of rectangular windows in a two-cl imensional network, each of which can either "jump" or "roll" to a new location (section 4.3 of Part II); a subset of neurons in a sparsely active network inducting all neurons which don't have prescribed default values and hence do require storage space (section 4.4 of Part II); and a subset of neurons determined as the Cartesian product of several simpler foci of attention (section 4.5 of Part 11). The designs presented in section 2 of Part 11 are theoretically well-rnotivatecl but may need to be revised in the light of subsequent experimentation, which is beyond the scope of the present paper.

Finally, a brief summary of our work is given in the concluding section 4.

2 DYNAMICAL OBJECTIVE FUNCTIONS AND LAGRANGIANS

We have argued that fundamentally, a computing system is designed by trading off two competing utilities: its cost of operation and its *functionality*. We may specify a fixed allowable cost and seek to obtain maximal functionality, or we may specify a fixed functionality and seek to obtain a minimal cost, or we may seek a specified trade-off between cost and functionality. W'e may specify further dynamical *constraints* required for implementability. With Lagrange multipliers and/or penalty terms we may reduce all these cases to extremizing

$$S = AC_{\text{cost}} + BF_{\text{functionality}}, \tag{4}$$

where the system is more functional for lower values of F, and where any dynamical constraints have been absorbed into the C_{cost} term. Now the designer's problem is to find functions C and F (perhaps based on equations (I) and (2)) which depend on the trajectory of some vector of state variables x(t) over time, such that the global optimization of S can be reduced to a collection of *local decisions* about how to change the individual components of the state vector x at a given small time step from time $t - \Delta t$ to time t. (A local decision could be viewed as the choice of the value of a variable (e.g. a control variable).) These decisions must however be made by very simple

physical devices such as transistor circuits containing only a few transistors. *Such local decisions* will prove to be analogous, in a physical system, to a differential or difference equation formulation of dynamics that follows from the principle of least action for the same system.

For example, it would be advantageous if C and F were each sums (or integrals) over a collection of decisions spread out over space and time. To express this summation, let us index the components of the state vector x by an index s. Since s indexes all the variables present at a fixed time, those variables could be viewed as being cm bedded in one fixed-tirllc slice of a space-ti[nc volume, in which case s may also be viewed as indexing spatial locations in the system. So wc refer to s as the *spatial index* and t as the *temporal index*; the entire trajectory of a computation is specified by $\{x(s,t)\}$. Then the sum over decisions would be

$$S = A \sum_{\text{decisions}(s,t)} C_{s,t}(\{x(s',t')\}) + B \sum_{\text{decisions}(s,t)} F_{s,t}(\{x(s', t')\}) \tag{5}$$

where each function $C_{s,t}$ or $F_{s,t}$ may depend on only a few of its arguments $\{x(s',t')\}$ and hence on only a small part of the trajectory near (s, t). In equation (5) we may introduce a continuous time axis by replacing the temporal sums by integrals; we can do this by integrating over t and summing over s. Following the analogy with physics, S is referred to as the "action". The decomposition (5) would be a useful first step towards enforcing spatial and temporal *locality on* the dynamics of our computation, since the decomposition distributes S over a sum of terms which pertain to particular spatial and temporal locations. Unlike space, time has an intrinsic directionality, and we will also need to enforce *causality* in the optimization of S. Before seeking specific forms for $C_{s,t}$ and $F_{s,t}$, we will discuss locality and especially causality.

A pattern of *communication* is implicit in the dependence of $C_{s,t}$ and $F_{s,t}$ on $x(s'$, t'). If $C_{s,t}$ and $F_{s,t}$ were each a function only of $x_{s,t}$, rather than a functional of the entire State vector $\mathbf{x}(t')$ at many different times t', then every decision term could be optimized independently, and the associated computation would proceed without any communication. This is a trivial case, however, and generally we will have quite a bit of interaction (via specific C and F terms) between variables defined at different times and places. (For a non-trivial example see the region-segmentation Lagrangian of section 2. I .2.) The pattern of communication is defined by a communication graph whose nodes are space-time sites (s, t) and whose links record the presence or absence of functional dependencies of $C_{s,t}$ or $F_{s,t}$ on trajectory variables x defined at other space-time sites (s', t'). We want to keep this implicit pattern of communication relatively local, and we insist that it be causal.

The effect of causality on the communication pattern is twofold. (i) Causality favors the adoption of a *convention* in which interactions between variables indexed by different times are entirely incorporated in the C and F terms indexed by the later of the two times, and do not enter into the C and F terms defined at the earlier of the two times. That way, every C_t or F_t term depends only on variables indexed by times $t' \leq t$. This is called the *retarded interaction form* of S. (ii) If we introduce computational dynamics by sequential optimization, at successive time steps t' of sets of variables indexed by t', then causality denies a computation the possibility of optimizing all terms of S with respect to any one variable $x(s'$, t'). Instead, each variable $x(s',t')$ can only be varied under an objective involving those terms of S all of whose variables $x(s'',t'')$ are optimized at the same time as $x(s',t')$ or earlier. The values of all other variables (those indexed by $t'' > t'$) are as yet undetermined. Which terms of S are eligible to participate in the variation of $x(s', t')$? Any C_t or F_t term for which $t > t'$ depends on variables (such as $x(s, t)$) which have unknown values at time step t' and are not being varied at that time step. Such a term is is ineligible; so we are restricted to those terms of S indexed by time $t \leq t'$.

Note that the eligible terms of S with $t \leq t'$ arc mostly irrelevant to the optimization of $x(s'$, t'), since point (i) imples that the $t < t'$ terms do not contain the variable $x(s', t')$. 'l'his leaves only the $t = t'$ terms of S to determine $x(s', t')$.

Of course, an acausal optimizer could achieve a better value for S by being less "greedy" (increasing present $C_t + F_t$ terms to decrease future ones by a greater amount), but as argued above *causality forces our dynamics to be greedy.* In other words, the causality constraint only permits a partial or *greedy optimization* of S, and the nature of the partial optimization depends on the decomposition of S into a sum over decisions of causally constrained terms. This basic limitation to causal or *greedy dynamics* will be more or less severe depending on which of many possible decompositions of C and F over time is chosen.

We shall define the *greedy derivative* of S with respect to $x(s',t')$ as being the ordinary derivative of the sum of such eligible $(t \leq t')$ terms of S', and use that derivative to define optimality of $x(s',t')$. But this greedy derivative immediately simplifies due to the retarded interaction form off' and F:

$$\frac{\partial_G}{\partial_G x(s',t')} \sum_t (AC_t + BF_t) \equiv \frac{\partial}{\partial x(s',t')} \sum_{t \leq t'} (AC_t + BF_t) = \frac{\partial}{\partial x(s',t')} (AC_{t'} + BF_{t'}). \tag{6}$$

How can we find functions L'(x{t'}) and I"(x{t'}) that specify (via optimization of S) an entire computational task and yet break up into a sum over easily computed decisions? This is a statement of the problem of algorithm design, for which there is no general answer, but we can still invent some fairly general techniques. The cost function can be regarded as some kind of space-ti[nc volume to be minimized (e.g. circuit size times the duration of its use) and can be decomposed into a sum of space-time volumes for the many elementary decisions or state changes, at individual locations and times, that comprise the associated computation:

$$\mathrm{c} = \mathrm{Vol} = \sum_{s,t} \delta \mathrm{Vol}_{s,t}. \tag{7}$$

Also the functionality $F(\mathbf{x}\{t'\})$ is often measured by some definite objective function $E(\mathbf{x})$, such as total tour length in a traveling salesman problem [11185], and this can be decomposed over time as (cf. equation (2))

$$F(\mathbf{x}_{\text{final}}) = E(\mathbf{x}_{\text{final}}) - E(\mathbf{x}_{\text{initial}}) = \sum_t \Delta E \Big|_{t-\Delta t}^{t}. \tag{8}$$

For example, a standard form for analog neural networks' objectives E is [MG90]:

$$E(\mathbf{v}) = -\frac{1}{6} \sum_{ijk} T_{ijk} v_i v_j v_k - \frac{1}{2} \sum_{ij} T_{ij} v_i v_j - \sum_i h_i v_i + \sum_i \phi_i(v_i), \tag{9}$$

which encompasses many network designs including equation (3). Here v takes the place of x, and the indices i, j, and k take the place of s. In equation (9), v_i is the output value of neuron i; T_{ij} and T_{ijk} are **connection** weights between two and **three** neurons, respectively; h_i is a bias input to neuron i; and $\phi(v_i)$ is the potential function for neuron i and determines the transfer function g_i (e.g. a sigmoid function) through

$$v_i = g_i(u_i) \quad \text{and} \quad u_i = \phi'(v_i). \tag{lo}$$

Often equation (9) is further specialized by setting $T_{ijk} = 0$.

As a complete example of a dynamical objective function we present, in the following equation (11), a dynamical objective for the Hopfield/Grossberg dynamics of an analog circuit. This dynamical objective will be derived in sections 2.1 and 3.2, using the fact (to be established in section 2.2) that, for a continuous-time analog circuit model, a condition for the greedy optimization takes the form of a (functional) derivative $\delta/\delta\dot{v}$ (where $\dot{v}_i = dv_i/dt$). The dynamical objective is

$$S[\dot{\mathbf{v}}(t), \mathbf{v}\ \mathrm{V(t)}] = \int dt \sum_i \left(K[\dot{v}_i, v_i] + \frac{\partial E}{\partial v_i} \dot{v}_i, \right. \tag{11}$$

where $K[\dot{v}, v]$ is a cost-of-movement term to be derived in section 3 (see Theorem 1). Varying with respect to $\dot{v}_i$ and making use of the form of E given by equation (9), we will find analog neural-net equations of motion as expected:

$$\tau_H \dot{u}_i = \sum_{jk} T_{ijk} v_j v_k + \sum_j T_{ij} v_j + h_i \quad \text{and} \quad v_i = g(u_i) \ . \tag{12}$$

Here τ_H is a time constant. The dynamical objective function S of equation (11) can be recognized as an instance of (5) by identifying the neuron index i with the space index (i.e. component index) s and the time integral $\int dt$ with the temporal sum $\sum_t$; also $C_{st} \to K[\dot{v}_i\ (t), v_i(t)]$ and $F_{st} \to (\partial E[\mathbf{v}(t)]/\partial v_i)\dot{v}_i(t)$.

There is a close analogy between equation (5) and standard ideas and terminology in physics. The *action*, S, can be decomposed into the temporal sum (in physics, an integral) of a Lagrangian $L(t)$ which in turn is a spatial sum of a *Lagrangian density* $L_{s,t} = C_{s,t} + F_{s,t}$:

$$\begin{aligned} S &= \sum_t L(t) \\ &= \sum_{(s,t)} L_{s,t}(\{\mathbf{x}(t')\}) = \sum_{(s,t)} (C_{s,t} + F_{s,t}) \end{aligned} \tag{13}$$

(Note that the sum over time may become an integral when we consider time steps of infinitesimal duration, since the extra factor of At required to get an integral is just a constant that doesn't affect the solution to an optimization problem.) For our neural network design purposes the Lagrangian L is generally the most useful of these alternative notations, particularly for algebraic manipulation, because the temporal sum has the same algebraic form from one problem to the next (and hence is uninformative), but the spatial sum does not.

Extremization of such functions (or functional) provides a foundation for the study of many dynamical systems including quantum field theories. F and C might with lower confidence be identified as classical kinetic energy and potential energy terms respectively, but as we will see, many details are different. These differences prevent a literal-minded mapping of our ideas and constructs onto the formalism of physics. In particular, causality is not built into physical theories by means of the partial optimization of S, but in a completely different way that is inconvenenient for treating irreversible dynamics such as our computations; therefore neither the dynamics nor the Lagrangians of physics can be called "greedy" in the sense wc use the term.

There are a number of other ways to derive dissipative dynamics from Lagrangians, as summarized in [VJ89]. Allowing explicit time dependence, such as an overall factor of $e^{\mu t}$, in a conventional Lagrangian permits physically clamped second-order dynamics to be derived. The strategy of the approach is to start with a differential equation, derive an associated Lagrangian (this is called the inverse problem of the calculus of variations, and it may have many solutions), and use that Lagrangian to analyse or approximate the solutions of the differential equation. Our strategy and methods differ, since the Lagrangians are obtained from cost and functionality considerations and hence are known before the differential equations are known. Moreover these Lagrangians require an unconventional variational principle (the greedy variation) to produce acceptable differential equations. Nevertheless there may exist some deeper relationships between our greedy Lagrangians and previous approaches discussed in [VJ89].

2. I Cost and Functionality Terms

Equation (8) for F is particularly appropriate for a net whose dynamics is intended to converge to fixed points that encode the answer to a static optimization problem, such a-s the standard neural network form of (9). Equation (8) represents a substantial specialization from the general set of functions $F_t(\{x(s', t')\}) = \sum_s F_{s,t}(\{x(s', t')\})$ that appears in (5). For in equation (8), F_t depends on t only through its arguments and not through its subscript, so that the algebraic form of F_t is independent of time (i.e. F_t is autonomous):

$$F_t(\{x(s', t')|t' \leq t\}) = E[\mathbf{x}(t)] - E[\mathbf{x}(t - \Delta t)]. \tag{14}$$

In the simplest case of static special-purpose neural circuitry the computational cost is just a constant N, reflecting the hardware committed (neurons and connections), times the length of time it is used:

$$C = ANt_{total} \tag{15}$$

for fixed hardware, or the more general

$$C = A\int dt N(t) \tag{16}$$

if the amount of hardware devoted to the network can vary over time (a possibility we will consider in detail in section 2 of Part II. Once N is allowed to vary with time, it becomes relevant to consider the details of how much node and wire volume is required to implement dynamically a given pattern of connections.

Equations (14) and (15) go part of the way towards defining a computational system, but they arc not yet detailed enough to specify a parallel algorithm or analog circuit that optimizes E. Our main line of development will be from these equations towards an analog circuit. But first we note an alternative strategy for generating parallel algorithms which will be developed in sections '2. I and 2 of Part II.

2.1.1 **Remarks on** *Some* **Generalizations**

It is by no means necessary to specialize the expression for S in (5) all the way to the form in (14), if some other way to minimize the original action in (4) can be found. Most alternative sets of F functions would pertain only to one particular objective function E, but there are also systematic methods for deriving F_t from E in which F_t benefits from retaining an explicit time dependence. For example, F_t might take the form of $\Delta E_{\alpha(t)}$ for one of p possible objectives E_α, where the choice of objective as a function of time (given by $\alpha(t) \in \{1, 2, \ldots p\}$) is made in a cyclic fashion. Then (14) is replaced by

$$F_t(\{x(s', t')|t' \leq t\}) = \sum_\alpha \psi_\alpha(t) \Delta E_\alpha[\mathbf{x}(t), \mathbf{x}(t - \Delta t); \mathbf{x}(t^{\text{old}})], \tag{17}$$

where $\psi_\alpha(t) = 1$ if $\alpha = \alpha(t)$ and O otherwise, and where

$$\Delta E_\alpha[\mathbf{x}(t), \mathbf{x}(t - \Delta t); \mathbf{x}(t^{\text{old}})] = E_\alpha[\mathbf{x}(t); \mathbf{x}(t^{\text{old}})] - E_\alpha[\mathbf{x}(t - \Delta t); \mathbf{x}(t^{\text{old}})]. \tag{18}$$

Here we assumed that t' takes only the values $t, t - \Delta t$ and t^{old}, where t — At is the previous time step in the current α phase of the cycle and t^{old} is the final time step of the previous phase $\alpha - 1$ in the cycle. Because of its explicit dependence on a cyclic *clock signed et(t),* E_α is called a *clocked objective junction.* It must be fundamentally connected to the original objective function E if the resulting cyclic Lagrangian is to have the correct functionality, but there are several ways of making such a connection. This possibility is explored further in section 2.1 of Part II and applied extensively in section 2 of Part II.

It is troubling that there exists a wide variety of different local and causal Lagrangians (cf. (5)) each of whose dynamics will partially optimize the original dynamical objective function or action given by (4). How do we choose one over another, and what are the minimal criteria for any to be acceptable? In other words, what are the rules of the game for proposing distributed cost and functionality terms in (5)? 'l'he answers must ultimately be related to algorithmic performance in minimizing the action itself (see (4)). We begin our work on these questions in section 2.3.2.

2.1.2 **Refinement to Continuous Dynamics**

For the moment, let us assume that (14) and (15) describe an acceptable Lagrangian, which is a decomposition of (1) and (2) to finite-sized time steps, and try to further refine them to a dynamics with infinitesimal time steps, i.e. continuous time and continuous-valued (i.e. analog) variables.

A standard form for analog neural networks' objectives E is given in (9). The corresponding functionality term F may be derived with a series of three design transformations. Start with an objective function $E[\mathbf{v}]$ of continuous variables $v_1 v_m$ ancl discrete 0/1-valued variables $v_{m+1} \ldots v_n$, with $\phi_i(v_i) = O$ for the latter (where ϕ is defined in (9)). The first transformation is to reformulate the discrete variables as continuous variables each with the *constraints* that $O \leq v_i \leq 1$. This step may introduce new local minima at the intermediate values of v_i; if this possibility can be analyzed away, or designed away by adding a "bump term" such as the penalty term $\sum_i c_i v_i (1 - v_i)$ to E, then we have a valid transformation. l'he second transformation is to replace the constraints with penalty or barrier terms $\phi_i(v_i)$ added to E for unconstrained, continuous-valued optimization. Steps 1 and 2 together may sometimes be replaced by the one-step Mean Field Theory derivation of continuous-valued objectives for discrete-valued variables (first discussed in [Hop84] and extended by others inducting [Sim90, PS89, GY91]) with improved control over local minima. But in section 2 of Part II we will have occasion to separate the two steps,

As an example of these first two steps, the image region segmentation objective (3) can be refined to an analog neural net with discrete variables s $\in \{O, 1\}$ replaced by continuous variables

$l \in [0,1]$:

$$
\begin{aligned}
E(f, l^h, l^v) \quad = \quad & \frac{A}{2}\sum_{ij}(f_{ij} - d_{ij})^2 + \frac{B}{2}\sum_{ij}(f_{i+1,j} - f_{ij})^2(1 - l^v_{ij}) \\
& + \frac{B}{2}\sum_{ij}(f_{i,j+1} - f_{ij})^2(1 - l^h_{ij}) + \mu\sum_{ij}(l^h_{ij} + l^v_{ij}) \\
& + \sum_{ij}\phi_i(l^h_{ij}) + \sum_{ij}\phi_i(l^v_{ij})
\end{aligned}
\tag{19}
$$

Finally, we must refine the global objective E into an arbitrarily large number of infinitesimal-step ΔE terms for use in the simplest continuous-time dynamics. Using Taylor's theorem for small At,

$$
F_{\text{coarse}} = \Delta E \approx \text{At} \sum_i E_{,i}\dot{v}_i \equiv \Delta t F_{\text{fine}}[\dot{\mathbf{v}}] \tag{20}
$$

(so that $\sum_t F_{\text{coarse}} \approx \int dt F_{\text{fine}}$), where

$$
E_{,i} \equiv \frac{\partial E}{\partial v_i} = -\frac{1}{2}\sum_{ij} T_{ijk} v_j v_k - \sum_j T_{ij} v_j - h_i + \phi'(v_i), \tag{21}
$$

and v is a vector of variables comprised of all the f, 1", and l^h variables of (19). This third transformation step does not yet specify the associated transformations of the fine-scale cost term $C_{\text{fine}}[\{v_{s,t}\}]$ which we will work out in section 3. The result will be of the form $C_{\text{fine}}[\dot{\mathbf{v}}] = \sum_i K[\dot{v}_i, v_i]$ (cf. (117) of section 3.2). Together with (20), this gives us the Lagrangian

$$
LF_{\text{fine}}[[\dot{\mathbf{v}}, \mathbf{v}] = \sum_i \left(K[\dot{v}_i, v_i] + \frac{\partial E}{\partial v_i}\dot{v}_i \right) \tag{22}
$$

and the action functional

$$
S = \int dt L_{\text{fine}}. \tag{23}
$$

This action is in agreement with equation (1 1). For the region segmentation example, $\partial E/\partial v_i$ is given by (21).

In summary, we have *transformed* the problem three times along the way to the circuit-level functionality term in (20) and an associated Lagrangian. The transformations are intended to preserve (approximately) the fixed points of the equations of motion, while making the dynamics progressively more implementable as an analog neural network. Both thetransformations) validity (as measured by the functionality term of the original coarse-scale action (4)) and their efficiency (as measured by the cost term of (4)) must still be demonstrated, since the finer-scale versions of this action functional are only partially optimized. The three transformations used to obtain equation (20) were: (1) discrete variables $\rightarrow$ continuous variables, constrained to intervals; (2) constraints $\rightarrow$ penalty or barrier terms in unconstrained, continous optimization; and (3) temporal refinement: $F_t = \Delta E \approx \int dt \dot{E}$. (The refinement of C must still be worked out before we have a derivation of the fine-scale Lagrangian. See section 3.)

2.2 Greedy Functional Derivatives

Based on the foregoing work, we seek to derive continuous-time dynamics from suitable Lagrangians. This requires formulating the greedy derivative of (6) for use with continuous-time dynamics, hence formulating it as a functional derivative.

Following equation (5), we argued that the local cost and functionality terms $F_{s,t}$ and $C_{s,t}$ in a Lagrangian should depend on variables $x_{s',t'}$ only for $t' \leq t$, and that only variables with $t' = t$ should be varied in the optimization of $F_{s,t} + C_{s,t}$; all values of earlier variables are held fixed. Then F and C are said to be in *retarded interaction form*. These constraints can be imposed on any continuous-time Lagrangian in differential form,

$$
L,(x(t),\ x(t),\ x(t),\ldots), \tag{24}
$$

as follows. First we replace the derivatives by difference expressions $(x(t) - \mathbf{x}(t - \text{At}))/\text{At}$, and so on, taking care that the largest time t' to appear is t. This yields an approximate discrete-time

Lagrangian, which we then optimize with respect to x(t) by differentiating to find the dynamics. Then we take the limit as At $\to$ (). In that way we ensure that $t' \leq t$ (retarded interaction form) and that only variables for which $t' = t$ are actually optimized at time t, as required.

This procedure for finding the continuous-time dynamics for a Lagrangian in differential form (24) may be formalized by means of the greedy *functional derivative* introduced in [MG90, MM91]. Here we provide a new formal derivation of the greedy functional derivative δ_G which exploits the retarded interaction form of a Lagrangian.

Let N be a normal form operator on derivative expressions:

$$\begin{aligned} N[x(t)] &= x(t), \\ N[\dot{x}(t)] &= (x(t) - x(t - \mathrm{At}))\ /\mathrm{At}, \\ N[\ddot{x}(t)] &= (x(t) - 2x(t - \Delta t) + x(t - 2\Delta t))/(\Delta t)^2, \\ &\quad \text{and so on. Also} \\ N[F[y(t)]] &= F[N[y(t)]],\ \mathrm{y} = x(t), \dot{x}(t), \ddot{x}(t),\ \text{if } F \text{ is autonomous.} \end{aligned} \tag{25}$$

So N serves to replace time derivatives by temporal difference expressions for which t' $\leq t$, which we can differentiate with respect to $x(t)$. In other words, it suffices to put a Lagrangian L into retarded interaction form, so that a greedy variation can be taken while preserving its value in the $\Delta t \to$ O limit. (N is known in numerical analysis as the "backward divided difference operator".) Then the greedy functional derivative may redefined, evenon Lagrangians L not yet in retarded interaction form, so as to agree with (6): For any small Δt >0,

$$\begin{aligned} \frac{\delta_G}{\delta_G x(t)} \int d\hat{t} L(x(\hat{t}), \dot{x}(\hat{t}), \ldots) &\equiv \int d\hat{t} \delta(\hat{t} - t) \frac{\partial}{\partial x(\hat{t})} N L(x(\hat{t}), \dot{x}(\hat{t}), \ .) \\ &= \frac{\partial}{\partial x(t)} N L(x(t), \dot{x}(t), \ldots) \qquad \text{(as in (6))} \\ &= \frac{\partial}{\partial x(t)} L(N[x(t)], N[\dot{x}(t)], \ .) \\ &= \frac{\partial}{\partial x(t)} L(x(t), \frac{x(t) - x(t - \Delta t)}{At}, \ldots), \end{aligned} \tag{26}$$

where the last step used (25). Continuing,

$$\begin{aligned} \frac{\delta_G}{\delta_G x(t)} \int d\hat{t} L(x(\hat{t}), \dot{x}(\hat{t}), \ldots) &= \frac{\partial}{\partial x(t)} L(x(t), \frac{x(t) - x(t - \mathrm{At})}{\Delta t}, \ldots) \\ &= \Big(\sum_{n=0}^{\infty} \frac{1}{(\Delta t)^n} \frac{\partial}{\partial (d^n x(t)/dt^n)(t)} \Big) L(x(t), \dot{x}(t), \ldots) \\ &\quad \text{(by the chain rule)} \\ &= \int d\hat{t} \delta(\hat{t} - t) \Big(\sum_{n=0}^{\infty} \frac{1}{(\Delta t)^n} \frac{\partial}{\partial (d^n x(t)/dt^n)(\hat{t})} \Big) L(x(\hat{t}), \dot{x}(\hat{t}), \ldots) \\ &= \Big(\sum_{n=0}^{\infty} \frac{1}{(\Delta t)^n} \frac{\delta}{\delta (d^n x(t)/dt^n)(t)} \Big) \ d\hat{t} L(x(\hat{t}), \dot{x}(\hat{t}), \ldots). \end{aligned} \tag{27}$$

Here the functional derivatives $\delta/\delta(d^n x(t)/dt^n)$ are taken to be independent of one another as partial functional derivatives (so for example $\delta\dot{x}(\hat{t})/\delta x(t)$ = O, rather than $\delta\dot{x}(\hat{t})/\delta x(t) = d\delta(\hat{t} - t)/d\hat{t}$ *as* would be the case for total functional derivatives).

So the greedy functional derivative $\delta_G/\delta_G x(t)$ is given by the operator equation

$$\frac{\delta_G}{\delta_G x(t)} = \sum_{n=0}^{\infty} \frac{1}{(\Delta t)^n} \frac{\delta}{\delta (d^n x(t)/dt^n)(t)}, \tag{28}$$

where Δt is infinitesimal, Again, the conventional functional derivatives are independent of one another (they are partial functional derivatives). Needless to say, the highest powers of (l/At) will dominate all others in the limit At $\to$ O. For example if L depends on v and v, but not on higher time derivatives, then the greedy functional derivative will be $(1/\Delta t)\delta/\delta\dot{\mathrm{v}}$. This will generally be the case for our circuit Lagrangians.

We can derive analog, continuous-time network dynamics by applying the greedy functional derivative to the continous-time Lagrangian (22). Since the highest time-derivative in the Lagrangian is $\dot{v}_i$ for each variable v_i the greedy functional derivative is proportional to $\delta/\delta\dot{\mathbf{v}}$. Then the equations of motion become

$$\frac{\delta S}{\delta \dot{v}_i} = K_{,\dot{v}}[\dot{v}_i, v_i] + \frac{\partial E}{\partial v_i} = 0 \tag{29}$$

For $K[\dot{v},v] = (1/2)\tau_H \dot{v}_i^2/g'(g^{-1}(v_i))$, the circuit-level cost term which will be derived in section 3.2.3, and for an objective function E given by actuations (9) and (10), the greedy variation equations become Hopfield/Grossberg dynamics:

$$\tau_H \dot{u}_i + u_i = \sum_{jk} T_{ijk} v_j v_k + \sum_j T_{ij} v_j + h_i \quad \text{and} \quad v_i = g(u_i). \tag{30}$$

This type of dynamical system describes an analog neural network, and we will make no distinction between such a dynamical system and the neural network itself.

As an example, we may work out the dynamics for the region segmentation Lagrangian given by (22) and (19). Specializing the dynamics of (30) to the region segmentation objective (19), we can expand the first term of the objective to find a potential term $(A/2)f_{ij}^2$ for the f_{ij} variables. Then we find the standard Hopfield/Grossberg equations of motion for this analog network, which are

$$\begin{array}{lllll}
\tau_f \dot{e}_{ij} + e_{ij} & = & A\ d_{ij} - B(f_{ij} - f_{i+1,j})(1 - l_{ij}^v) - B(f_{ij} - f_{i,j+1})(1 - l_{ij}^v), & f_{ij} & = (1/A)e_{ij}, \\
\tau_k \dot{k}_{ij}^v + k_{ij}^v & = & B/2 \sum_{ij}(f_{i+1,j} - f_{ij})^2 - \mu, & l_{ij}^v & = g(k_{ij}^v), \\
\tau_k \dot{k}_{ij}^h + k_{ij}^h, & = & B/2 \sum_{ij}(f_{i,j+1} - f_{ij})^2 - \mu, & l_{ij}^h & = g(k_{ij}^h).
\end{array} \tag{31}$$

2.3 Theory for Refinement to Circuit Lagrangians

We have found a path of argument from computational first principles to specific neural networks, but the status of some of the steps along the path is still unclear. The basic problem is that various transformations of the original action functional (4) are required to get an implementable dynamical system, and limitations of causality and the simplicity of elementary processing devices require that the spatially and temporally distributed Lagrangian functional (such as (5) or (11)) be optimized only partially (as in the discussion following (5)).

Our approach to this basic problem is to catalog a variety of useful transformations that lead towards circuits or parallel algorithms, and to re-use the fundamental dynamical objective function (4), or closely related quantities, as a measure (i.e. a criterion) for judging the success of such transformations. Such a criterion may be called a *meta-objective* since it is an objective function used to select a dynamical objective function for the neural network dynamics.

This approach may be thought of as a symbolic search procedure to be carried out by human designers, who select the likely transformation sequences, with machine assistance in evaluating them and perhaps also performing them. On occasion it may be possible to eliminate the search procedure by proving the (meta-)optimality of a given Lagrangian, but we do not think that this will be possible in most cases.

2.3.1 Transformations of Lagrangians

Recall the three transformations leading to circuit-level Lagrangians in section 2.1.2:

T1. discrete variables $\rightarrow$ continuous variables constrained to intervals

T2. constraints $\rightarrow$ penalty or barrier terms in unconstrained continous optimization

T3. refinement: $F_t = \Delta E \approx \int dt \dot{E}$. (The refinement of C will be worked out in section 3.)

We comment on each of these transformations.

T1 and T3 are required to achieve a circuit implementation, but more generally they serve the purpose of making a parallel algorithm. Discrete-time update schemes may be introduced instead, but some care is required so that the updates of independent variables done in parallel don't have

the joint effect of increasing rather than decreasing E. For example, for some networks it is possible to "color" the variables with a small number of colors so that no two connected variables (x_i and x_j such that $T_{ij} \neq 0$) have the same color; then different colors can be updated at different times in a clocked objective function, and all the variables of the same color can be updated at once (even by discrete jumps) without interference in E. (Interference would mean that several variables would each, if updated alone, diminish E, but if the same updates were done together then E could increase.) Such (fairly standard) parallel update schemes are not so important for continous-time and analog-valued networks, whose descent dynamics are explicitly parallel.

Transformations like T2, which incorporate static constraints into the static optimization problem, may change the nature of the optimization problem significantly. Penalty and barrier terms on constraints that involve many variables destroy locality, unless they are further transformed to a local form by methods such as those described in [M G90]. In this case a minimization problem is replaced by a saddle-point problem. Alternatively one can introduce Lagrange multipliers, but that also changes the static optimization into a saddle point problem [1'1187], Either way, the dynamics associated with the Lagrangian functional loses its obvious convergence properties (because limit cycles around a saddle point become possible), and it may be necessary to engage in *meta-optimization* of some kind in order to secure convergence for a local circuit implementation. Another alternative, which requires clocked objective functions but does not explicitly introduce saddle points, is to use an algorithm similar to the "gradient projection algorithm" or "scaled gradient projection algorithms" [BT89] to repeatedly reestablish the constraints as the dynamics proceed. Such an alternative will be employed in section 2.1.6 of Part II.

In previous work [MG90] it has been demonstrated that static neural network objective functions may be transformed in a variety of ways in order to acheive design goals such as reduced wiring cost or attaining an implementable form while preserving the functionality (the fixed points) of an optimizing neural network. Likewise, in this paper we will introduce a number of transformations from one Lagrangian to another that satisfy design constraints while preserving or improving the functionality of a computation.

A fundamental aspect of (5) is that, clue to its linearity, it naturally supports the hierarchical decomposition of computational dynamics into large state changes (or decisions), each achieved through many smaller state changes or decisions. This is in analogy to multiscale or multigrid algorithms from numerical analysis, or to renormalization group ideas in statistical physics, or to the idea of stepwise refinement in the design of computer programs. As in (5), the action S can be decomposed into a sum over state-change decisions. But if each of these decisions is in turn made by a dynamical system consisting of a sequence of sub-decisions at a finer time scale (which may also involve a finer spatial scale), then we can relate the two time scales ("big" decisions and "sub-decisions") and reexpress the action in terms of the fine-scale decisions alone ("small" decisions):

$$
\begin{aligned}
S &= A \sum_{\text{big decisions}(\bar{s},\bar{t})} \tilde{C}_{\bar{s},\bar{t}}(\{\mathbf{x}(t')\}) + B \sum_{\text{big decisions}(\bar{s},\bar{t})} \tilde{F}_{\bar{s},\bar{t}}(\{\mathbf{x}(t')\}) \\
&= A \sum_{\text{big decisions}(\bar{s},\bar{t})} \left[\sum_{\text{sub-decisions}(s,t)} C_{s,t}(\{\mathbf{x}(t')\}) \right] \\
&\quad + B \sum_{\text{big decisions}(\bar{s},\bar{t})} \left[\sum_{\text{sub-decisions}(s,t)} F_{s,t}(\{\mathbf{x}(t')\}) \right] \\
&= A \sum_{\text{small decisions}} C_{s,t}(\{\mathbf{x}(t')\}) + B \sum_{\text{small decisions}(s,t)} F_{s,t}(\{\mathbf{x}(t')\}).
\end{aligned}
\tag{32}
$$

Notice that the step from equation (4) to equation (5), or more specifically to (7) and (8), can be given a similar hierarchical interpretation: we arc expressing a single quantity, optimized over the entire circuit convergence time, as a sum of quantities to be optimized more locally in time or space. The further refinement, to infinitesimal time steps, (23), is another example. Then equation (32) subsumes all these examples of hierarchical design.

2.3.2 Mets-Optimizatic]n

We have discussed the necessity for some criterion or figure of merit by which to compare alternative Lagrangians and tile dynamical systems to which they give rise. Generally we start with some global objective function such as S in (4), then transform it though a series of spatially and temporally localized Lagrangians of the form (5) to a final circuit-level Lagrangian L, which is only partially optimized (i.e. is greedily optimized) by the dynamics. Finally we wish to quantify the performance of the resulting dynamical system, i.e. to evaluate the quality of the associated computation, for example by computing the value of S at the end of a run. The [nets-optimization problem is to optimize the resulting evaluation, treating it as a functional of the exact form of L.

An obvious way to do that is by means of a retrospective (a posterior) evaluation of the original objective S of (4). But *optimizing* with respect to this protocol of retrospective evaluation of S_{coarse} seems out of the question, since that involves many repeated tests of the neural network dynamics with different values of the parameters that specify the (transformed) Lagrangian and is therefore far more expensive than one relaxation run of the network. (The parameterization of L may involve real-valued parameters or may simply be the discrete choice of a sequence of transformations to derive L from S_{coarse}.)

Fortunately the cost of optimizing S_{coarse} as a function of the form of L (i.e. the cost of meta-optimization) may by amortized over many inputs $\mathbf{h}$ (cf. (9)) to one network, drawn according to some probability distribution, or even over many network connection matrices T drawn according to another probability distribution. Optimizing M *may* be very expensive but the expense is amortized by using the resulting dynamics to improve the performance of many different computations. An apparent obstacle is that different $\mathbf{h}$ vectors and T matrices will in general have unrelated meta-objectives M, *so* amortization may be difficult to accomplish.

Such amortization may still be achieved if the meta-objective function $\mathcal{M}[L]$ is altered to become an average-case measure of S_{coarse}:

$$\mathcal{M}[L] = < S_{coar\ se}[L] >_{h,T} . \tag{33}$$

Just as in neural network learning procedures, the distribution average would be sampled by a finite sum over a *training set;* this sum would be optimized, and then a further sampling could bc made to test *generalization* from the training set to a testing set. If such generalization is to be expected, either on experimental evidence or according to theoretical criteria such as the Vapnik-Chervonenkis dimension [Vap82, BH89], then amortization will be possible. For the cost of computing (hence of optimizing) $\mathcal{M}[L]$ *is* multiplied by the size of the training set, but that large initial cost is then effectively divided tzy the number of times that L is used subsequently, which may be far larger than the training set. This gives the desired amortization.

Alternatively, one could amortize the cost of optimizing M by taking M to be a worst-case measure of S_{coarse} which can be optimized analy tically. The worst case performance is very hard to evaluate experimentally, but it may be more easily subject to analysis than the average-case performance, at least if we are allowed to alter the form of S_{coarse} somewhat. 'I'hat will be our approach in section 3.2.

3 CIRCUIT DYNAMICS

3.1 Refinement to a Circuit

Upon refinement, the Lagrangian L = C + F becomes

$$L = AN\Delta t + B\Delta E. \tag{34}$$

We would like to take the limit $\Delta t \to$ O, refining to infinitesimally small time steps in a continuous analog circuit. We expect this to be both simpler than a discrete-tizne (finite Δt) dynamics, and also more relevant to neural network implementations. But performing the greedy optimization of such a Lagrangian presents some surprising problems.

For instance, a first-orcler expansion of $\Delta E(\Delta t)$ yields a Lagrangian proportional to At: $L[\dot{\mathbf{v}}, \Delta t] = \Delta t(C + B\sum_i E_{,i}[\mathbf{v}]\dot{v}_i)$, which cannot be optimized with respect to At $\geq$ O without going outside the

expansion's domain of validity. To avoid this problem Δt might be taken to be a small constant, but that would make the entire ('os1 term $C = AN\Delta t$ constant and therefore irrelevant to the dynamic optimization problem. More seriously, partial optimization can only affect $\dot{\mathbf{v}}$ which appears linearly in this Lagrangian; $\dot{v}_i = \pm\infty$ will be the optimum, which would not only invalid ate the expansion of $E(t)$ again, but would violate physical limits on circuits as well. A somewhat more physical dynamics would result if we arbitrarily followed the analogy from the Lagrangians of physics and changed the cost term to a kinetic energy (1/2) $\sum_i \dot{v}_i^2$, but we have no computational justification) for doing so.

On the other hand, not expanding $\Delta E(\Delta t)$ at all leaves a fine-scale optimization problem which is equivalent to optimizing the full coarse-scale objective E in much less time, This is simply not possible. And even a second-order finite Taylor expansion of $\Delta E(\Delta t)$ is problematic, since the optimized values of Δt and $\dot{v}$ are likely to lie outside the expansion's small domain of suitability as an aproximation.

The essential problem here is that each fine-scale optimization, to be implementable as a circuit, must be *more* constrained than the coarse-scale optimization. We must stay within the domain of convergence of a Taylor expansion of $\Delta E(\Delta t)$, and we must not violate physical speed limits (e.g. for physical implementability we must prevent circuit time constants from becoming too small), and so on. Such constraints are either (a) direct physical limits on circuit implementations, or (b) computational limits on what can be achieved with a small amount of physical computing (computation which occurs in a physical medium) in time At. These constraints are generally too complex to state exactly in a simple Lagrangian.

We identify two general approaches to formulating such circuit constraints and the corresponding fine-scale Lagrangiaus. In the "underconstrained" approach, we impose simplified, loose versions of the physical and computational constraints on the optimization of L_{coarse}, in the hopes that the resulting dynamics will be constrained enough for a genuine physical implementation (perhaps at an even finer time scale). These loose constraints can be tightened up for analytic or computational convenience, and then expressed as penalty or barrier functions which are added to L to form L_{fine}, the fine-scale Lagrangian. By contrast, the "overconstrained" approach stays within the realm of physical irnplementation by hypothesizing a parameterized class of fine-scale Lagrangians known to be irnplementable, which can be thought of as alternative strategies, and optimizing some measure of their relationship to the original coarse-scale Lagrangian L. In particular, the cost terms of L_{fine} may be optimized while the functionality term is taken to be $\Delta E \approx \Delta t \dot{E}$ as in the coarse-scale Lagrangian. Thus the underconstrained approach applies looser constraints than implementability may actually require, and the overconstrained approach applies tighter constraints than are actually required. We give examples of each.

3.1.1 Underconstrained Refinement #1

We will require Av be small enough so that $\Delta E[\Delta \mathbf{v}]$ can be expanded to first (or second) order in a Taylor series, and that each $|\dot{v}_i|$ be bounded by a physical speed limitation. So we must optimize

$$L[\dot{\mathbf{v}}, \text{At}] = AN\Delta t + B\Delta E[\Delta \mathbf{v}] \tag{35}$$

subject to

$$||\dot{\mathbf{v}}||_\infty = \max_i |\dot{v}_i| \leq s \tag{36}$$

(where v $\approx \Delta \mathbf{v}/\Delta t$) and

$$||\Delta \mathbf{v}||_2 \leq r(v), \tag{37}$$

where $r(v)$ is chosen to ensure that a first (or second) order expansion of $\Delta E[\Delta \mathbf{v}]$ is sufficiently accurate. Also, there are two approaches to varying Δt. If we let At be optimized (subject to $\Delta t \geq$ O), the cost term in the Lagrangian will keep it small but not necessarily drive it to the continuum limit $\Delta t \to$ O. Or, we can let $\Delta t = \chi\tau$, where $\chi \in \{O, 1\}$ is a discrete dynamical variable which "stops" the neural network when χ is optimized to zero, and where τ is a small constant which we can analytically drive towards zero to extract continuum dynamics.

In the latter case, $||\Delta \mathbf{v}||_2 \approx \chi\tau||\dot{\mathbf{v}}||_2 \leq \chi\tau\sqrt{n}s$ is more restrictive in the limit $\tau \to 0$ than constraint (37) except when the network finally stops, at which time both constraints become

irrelevant. So we can drop constraint (37). If we express constraint (36) as a barrier function $\sum_i \phi_{\pm 1}(\dot{v}_i/s)$, the fine-scale Lagrangian becomes unconstrained:

$$L_{fine}[\dot{\mathbf{v}}, \chi] = \sum_i \phi_{\pm 1}(\dot{v}_i/s) + \chi\tau[AN + B\sum_i E_{,i}\dot{v}_i]. \tag{38}$$

Except for the new χ variable, this is the same form of Lagrangian for neural networks that we have proposed in [MG90, Mjo87]. The corresponding dynamics are (varying v, cf. (28))

$$\dot{v}_i = -sg_{\pm 1}(E_{,i}), \tag{39}$$

and varying χ to get the stopping criterion, we find the optimal values of χ occur only at the boundaries of the allowed domain of χ:

$$\chi = \Theta[s\sum E_{,i}g_{\pm 1}(E_{,i}) - AN]. \tag{40}$$

Here $\Theta(x)$ is the Heaviside function (1 for $x \geq$ O; O for $x <$ O).

3.1.2 Underconstrained Refinement #2

If, on the other hand, we let At be optimized freely, then we are taking a computational step that requires a small but nonzero amount of time to change the state by Av, which is constrained by both (36) and (37), which in turn are related by v $\approx \Delta\mathbf{v}/\Delta t$. We will express constraint (36) as $||\Delta\mathbf{v}||_\infty \leq s\Delta t$, which can be tightened to the more tractable

$$(1/s)\sum_i |\Delta v_i| \leq \text{At}. \tag{41}$$

Also we can tighten constraint (37) to

$$||\Delta\mathbf{v}||_\infty \leq \frac{r(v)}{\sqrt{n}} \equiv \hat{r}(v) \tag{42}$$

(which implies (37)). Optimizing $L[\dot{\mathbf{v}}, \Delta t]$ of (35) with respect to At, which occurs linearly in (35), as constrained by (41) just saturates the constraint: At = $(1/s)\sum_i |\Delta v_i|$.

The remaining constrained optimization is with respect to Av. Using barrier functions, we find an unconstrained Lagrangian

$$\hat{L}[\Delta\mathbf{v}] = \frac{AN}{s}\sum_i |\Delta v_i| + \sum_i E_{,i}\Delta v_i + \sum_i \phi_{\pm 1}\left(\frac{\Delta v_i}{\hat{r}(v)}\right), \tag{43}$$

or

$$\hat{L}[\Delta\mathbf{v}] = \sum_i E_{,i}\Delta v_i + \frac{AN\hat{r}(v)}{s}\sum_i \phi_{+/0/-}\left(\frac{\Delta v_i}{\hat{r}(v)}\right), \tag{44}$$

where $\phi_{+/0/-}(x) \simeq \phi_{\pm 1}(x) + |x|$. Also

$$\Delta v_i/\hat{r}(v) = \begin{cases} -1 & \text{if } E_{,i} - AN/s > 0 \\ 0 & \text{if } E_{,i} - AN/s < 0 \text{ and } E_{,i} + AN/s > 0 \\ +1 & \text{if } E_{,i} + AN/s < 0 \end{cases} \tag{45}$$

A number of calculations of bounding expressions $\hat{r}(v)$ are possible, but we will not pursue this approach further here.

3.1.3 Stopping Criterion

Lagrangians (38) and (43) each have intrinsic stopping criteria which compare the expected improvement in functionality ΔE with a cost of movement, and allow movement only when it is sufficiently beneficial. But E may not always be the right function for this purpose. A monotonic function $b(E)$ may be used in place of E in (8) ancl may likewise be decomposed into a sum of Δb terms. The latter would alter the tradeoff with the cost term for incomplete optimizations and

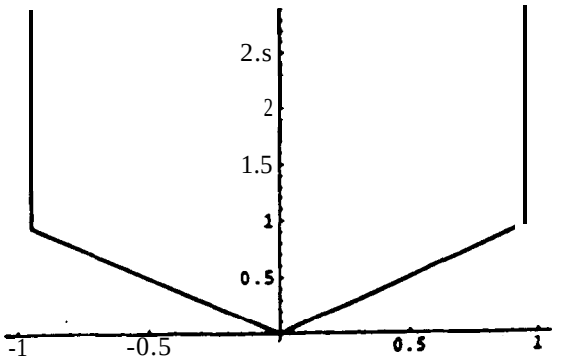

Figure 1: Potential $\phi_{+/0/-}(x)$ incorporates automatic stopping criterion. When other terms fail to alter the ordering among $\phi(-1)$, $\phi(0)$, and $\phi(1)$, then Av = O is favored and neuron v_i stops.

therefore the stopping criterion (the point at which a further decrease in F is smaller than the expected cost of obtaining it).

One major drawback of using a monotonic function b(E) in place of E in a Lagrangian is that if E is of the standard neural network form (9), it is already a sum of local terms and therefore close to neural implementation. By contrast, direct optimization of $b(E)$ requires a global calculation of E even to get the gradient, Vb = $b'\nabla E$ needed for the dynamics of every variable. One can circumvent this problem by transforming the objective function with a particular type of Legendre transformation [MG90]:

$$\chi b(E) \rightarrow -\sigma E + \chi\tau + \sigma b^{-1}(\tau). \tag{46}$$

In the resulting gradient dynamics, only the one variable σ requires computation of the objective function E. Unfortunately this transformation replaces a static minimization objet-tivc with a static saddle-point objective, since some of the new variables are to maximize rather than minimize the transformed objective. To find a Lagrangian which always converges, rather than cycling around the saddle point, may then require an appeal to meta-optimization (e.g. either experiment or deeper analysis) of the saddle-point-seeking Lagrangian.

3.2 Overconstrained Refinement: Mets-Optirrlization of K

A second, more systematic way to overcome the problems with refining the Lagrangian through expansion of $E(\Delta t)$ is to define a class of Lagrangians which are known to be physically implementable ancl mathematically tractable, though they are not the only physically implementable expressions for a circuit-level Lagrangian, and to pick the best member of the class based on a meta-optimization criterion. So we overconstrain the set of allowed Lagrangians and optimize. We will be able to do this theoretically for a meta-objective that measures worst-case performance of a Lagrangian for minimizing an especially simple class of neural network objective functions.

The allowable class of objective functions will be those of the form $E[\mathbf{v}]=-(1/2)\sum_{ij}T_{ij}v_iv_j-\sum_i h_i v_i+\sum_i\phi(v_i)$, in which the matrix T is negative semi-definite and has eigenvalues whose absolute values are bounded above by some number $t_{\max}$. An example of such an objective function is the hysteresis-free version of the common winner-take-all network objective [11"1'85] E = $(A/2)(\sum_i v_i - 1)^2 - \sum_i h_i v_i + \sum_i \phi(v_i)$. There is a straightforward generalization to the case in which different neurons v_i have different potential functions $\phi_i(v_i)$, but we won't work that out here. The negative-definite restriction on T is severe because it means that 1? must be unimodal (since each ϕ_i is unimodal too), Unimodal objectives have some computational uses, such as in the winner-take-all network or the "invisible hand" algorithm for matching [KY9 1], but our meta-optimization results will not be widely applicable until they are generalized to multi-modal objective functions. Nevertheless we can present the unimodal analysis as an example of the meta-optimization of a circuit-level Lagrangian.

What mathematical conditions would make a Lagrangian physically implementable, so the associated dynamics can be implemented with a circuit, ant] also result in good performance? The essential limiting factors for circuit speed arc the *time constants* (such as resistance-capacitance

products in an electrical circuit) that govern the approach to any stable state of any one- or two-element subcircuit. These time constants must be larger than some physical lower bound, say τ_{fast}. We also want the stable fixed points to be minima of some neural network objective E. Subject to these constraints, we want to minimize the slowest time constant for the full circuit (which as we will show is also larger than τ_{fast} Of course time constants are only defined for a local linearization of a dynamical system, so we must constrain them in the neighborhood of each attainable configuration, and we may optimize the worst case time constant over all such configurations.

With these points in mind, we define a constrained optimization problem over a limited class of Lagrangians of the form

$$L[\dot{\mathbf{v}}] = \sum_i K[\dot{v}_i, v_i] + \sum_i E_{,i}\dot{v}_i, \tag{47}$$

where the objective takes the form

$$E[\mathbf{v}] = -\frac{1}{2}\sum_{ij} T_{ij} v_i v_j - \sum_i h_i v_i + \sum_i \phi(v_i), \tag{48}$$

and $\mathbf{h}$ includes the input to the network. Note that the cost term in (47) is a sum over kinetic-energy terms each pertaining to only one neuron; this is a form of locality. Also the equivalence of stable fixed points and local minima of E can be ensured by simple constraints on K. (47) together with the time constant constraints and K constraints to be introduced specify the class of Lagrangians that we will call "circuit-implementable". 'T'his class is parameterized by the kinetic-energy function K from $\Re^2$ to $\Re$, suitably constrained.

Onc important property of equation (47) is that it retains its form under componentwise *reparameterizations* $v_i = f_i(x_i)$, where f_i is monotonically increasing, differentiable, and its inverse is differentiable. (Note that such repararneterizations form a continuous group under composition.) That is, under such a reparameterization the dE/dt term is invariant, and the K term, while not invariant, becomes another function $\tilde{K}[\dot{x}_i, x_i]$ of the corresponding new variables. So the problem of optimizing with respect to K can be solved equivalently in any such parameterization we choose, if only the objective ant] constraints are also chosen to be parameterization-invariant in this sense. We will insure that condition by deriving them from physical circuit time-constants for exponential convergence to fixed points.

The greedy **functional derivative was** derived in **section 2.2. We use that result to find the** greedy optimum of the action $\int dt L$ with respect to the trajectory $\mathbf{v}(t)$. The dynamical system that results from calculating the greedy variation of L with respect to v (i.e. the regular variation with respect to v) and setting it to zero is

$$\dot{v}_i = \tilde{K}[-E_{,i}, v_i], \tag{49}$$

where $\tilde{K}[w, v]$ is the inverse of $K[\dot{v}, v]_{\dot{v}}$ on its first argument. This forces us to constrain K to be monotonic in its first argument. Here we introduce the notation

$$w_i[\mathbf{v}] = -E_{,i} = \sum_j T_{ij} v_j + h_i - \phi'(v_i). \tag{50}$$

For stable fixed points to correspond to local minima of E (for which w = O), it suffices to assume that

$$\tilde{K}[0, v] = \text{O} \text{ and } \tilde{K}[w, v]_{,w} \geq \text{O} \tag{51}$$

for all w and v. The linearization of this dynamical system at v is

$$\Delta\dot{v}_i = \tilde{K}[w_i, v_i] + \sum_j A_{ij}\Delta v_j, \tag{52}$$

where

$$\begin{aligned} A_{ij} &= \frac{\partial}{\partial v_j}\tilde{K}[w_i, v_i] \\ &= \tilde{K}_{,w}[w_i, v_i]\Big(T_{ij} - \delta_{ij}\phi''(v_i)\Big) + \tilde{K}_{,v}\delta_{ij}. \end{aligned} \tag{53}$$

Now we are in a position to derive the constraints on the function Ii that result from considering the time-constants of the dynamics specified by A = (A_{ij}). We want the circuit elements and their

connections to be physically implementable, so we'll constrain one- and two-element subcircuits of the linearized system (52) to be slower than τ_{fast}. We do this by setting all elements of A to zero except for A_{ii} (for a one-element subcircuit) or $\{A_{ii}, A_{ij}, A_{ji}, A_{jj}\}$ (for a two-element subcircuit), to get a 1 x 1 or 2 x 2 matrix $\hat{A}(i)$ or $A(i, j)$. Furthermore, we may arbitrarily pick the subcircuit's fixed point v* by adjusting the input vector h; this does not alter any element of A or $\hat{A}$. In that case $K[w_i, v_i]$ = O, and the linearized dynamics (52) converges exponentially to $\mathbf{v}$ with a time constant determined by the largest eigenvalue $\{\lambda_i\}$ of the matrix A, i.e. by its matrix norm $||\hat{A}||_2$. So the physical constraint would be

$$\max_{\hat{A} \subset A} ||\hat{A}||_2 \leq 1/\tau_{\text{fast}}, \tag{54}$$

where A c A *means* that $\hat{A}$ is varied over all 1 x 1 and 2 x 2 submatrices of A and over all state vectors v.

The constraint (54) is parameterization-invariant. Invariance follows for any $\hat{A}$ by applying Taylor's theorem at a fixed point v* of $\mathbf{v}$, to get the linearized dynamics in a new coordinate system $\{x_i = f_i(v_i)\}$. The new matrix A is just a similarity transform $J\hat{A}J^{-1}$ of $\hat{A}$, where J is the (nonsingular) Jacobian of the change of coordinates. Therefore A and A have the same eigenvalues (cf. [Ner70], Theorem 5.2or 5.3) and $||\hat{A}||_2$ is parameterization-invariant as long as the Jacobian J is not singular (which ours never are). Furthermore, the identity of the 1 x 1 and 2 x 2 submatrices of A are invariant under our coordinate-wise reparameterizations $\{x_i = f_i(v_i)\}$. So the whole constraint (54) is parameterization-invariant. This invariance confirms the intuition that exponential convergence to a fixed point in one coordinate system $\{v_i\}$ (i.e. v –· v* $\approx$ c exp –λt) does not change its convergence exponent λ in another coordinate system $\{x_i = f_i(v_i)\}$.

Note that because each f_i is assumed to be monotonic, differentiable, and to have a differentiable inverse, constraints (51) are also parameterization-invariant. That's because each $w_i = -E_i$ is multiplied by $f_i'(v_i)$ in reparameterization $\{x_i = f_i(v_i)\}$, where O < $f_i'(v_i) < \infty$.

Constraint (54) is not a sufficiently convenient form for all our subsequent analysis, so we will relate the constraint to something more tractable. 'I'he matrix norm of each $\hat{A} \subset A$ *is* bounded above and below by multiples of $\max_{ab}|\hat{A}_{ab}|$ (cf. [G L83], p. 15):

$$\max_{ab} |\hat{A}_{ab}| \leq ||\hat{A}||_2 \leq \dim(\hat{A}) \max_{ab} |\hat{A}_{ab}|, \tag{55}$$

whence

$$\max_{ij} |A_{ij}| \leq \max_{\hat{A} \subset A} ||\hat{A}||_2 \leq 2 \max_{ij} |A_{ij}|, \tag{56}$$

where as before A ranges over all 1 x 1 and 2 x 2 submatrices of A. *So* a closely related but more tractable constraint may be formulated:

$$\max_{\mathbf{v}} \max_{ij} |A_{ij}(\mathbf{v})| \leq 1/\tau_{\text{fast}}. \tag{57}$$

Of course, the bounds of (56) hold regardless of what coordinate system is used tc, derive A, *so* long as A is expressed in the same coordinate system. Still, constraint (57) is *not* parameterization-invariant, since similarity transformations do not preserve the elements of a matrix. We will have occasion to use both (54) and (57) in what follows.

Since one K is to apply to many connection matrices T' and state vectors v, we will also constrain a worst-case estimate of the circuit speed over all T' in some allowable class T in the formula for A, and over all state vectors v for each connection matrix:

$$\max_{\mathbf{v}} \max_{T \in \mathcal{T}} \max_{\hat{A} \subset A} ||\hat{A}||_2 \leq 1/\tau_{\text{fast}}. \tag{58}$$

As previously mentioned, we take T to be the set of negative-se. midefinite connection matrices T', such that the absolute values of the T''s eigenvalues (i.e. T''s singular values) are bounded above by $t_{\max}$. Constraint (58) is parameterization-invariant but not as analytically tractable as the alternative,

$$\max_{\mathbf{v}} \max_{T \in \mathcal{T}} \max_{ij} |A_{ij}(\mathbf{v}, T)| \leq 1/\tau_{\text{fast}} \tag{59}$$

which will enter into the following analysis even though it is not parameterizat ion-invariant,

The invariance of constraint (58) is one reason to prefer the time-constant constraint (58) over the "speed limit" imposed in sections (3. 1. 1) and (3.1.2), which explicitly depends on the choice of variables, On the other hand the speed-limit constraints take into account the entirety of each trajectory, rather than just the behavior near (all possible) fixed points.

Next we must formulate the objective function, which will be a worst-case estimate of the much slower time constant for convergence of the full circuit (as opposed to 2 x 2 subcircuits). We want to minimize τ_{slow}, where

$$\begin{aligned} \tau_{\text{slow}} &= \max_{\mathbf{v}} \max_{T \in \mathcal{T}} \max_{i} 1/|\lambda_i(A(\mathbf{v}, T))| \\ &= \max_{\mathbf{v}} \max_{T \in \mathcal{T}} \max_{i} |\lambda_i(A^{-1}(\mathbf{v}, T))| \\ &= \max_{\mathbf{v}} \max_{T \in \mathcal{T}} ||A^{-1}(\mathbf{v}, T)||_2. \end{aligned} \tag{60}$$

Equivalently we want to maximize

$$\min_{\mathbf{v}} \min_{T \in \mathcal{T}} ||A^{-1}(\mathbf{v}, T)||_2^{-1}. \tag{61}$$

Again, the objective (60) will be parameterization-invariant because the time-constants arc invariant under similarity transformations.

Because the optimization of (60) with respect to $K[\dot{v}, v]$ subject to (58) is invariant under reparameterizations $x_i = f_i(v_i)$, we may change variables to $u_i = \phi_i'(v_i)$, calculate A for the linearized u variables, restate the optimization problem, and find the optimizing K. The functions ϕ are the single-variable potentials appearing in equation (48), so each ϕ_i' is monotonic, differentiable, and has a differentiable inverse. The variables u_i were introduced in equation (10). Using the $\mathbf{u}$ variables, one may express the dynamics by means of the Lagrangian

$$L = \sum_i \tilde{K}[\dot{u}_i, u_i] + \sum_i \frac{\partial E}{\partial u_i} \dot{u}_i, \tag{62}$$

whence the equation of motion

$$\dot{u}_i = \tilde{K}_{\dot{u}}^{-1}[-\frac{\partial E}{\partial u_i}, u_i] \tag{63}$$

(where the function inverse concerns only the first argument, $\dot{u}_i$, of $\tilde{K}_{,\dot{u}_i}$). This may be rewritten in terms of w_i from equation (50):

$$w_i = -\frac{\partial E}{\partial v_i} = -\frac{1}{g'(u_i)} \frac{\partial E}{\partial u_i} \tag{64}$$

which enables us to define

$$\hat{K}[w_i, u_i] = \tilde{K}_{,\dot{u}}^{-1}[w_i g'(u_i), u_i] \tag{65}$$

and reexpress the $\dot{u}_i$ dynamics as

$$\dot{u}_i = \hat{K}[w_i, u_i]. \tag{66}$$

Then the linearized dynamics is

$$\Delta \dot{u}_i = \hat{K}[w_i, u_i] + \sum_j A_{ij} \Delta u_j, \tag{67}$$

where $A_{ij} = \partial \hat{K}[w_i, u_i]/\partial u_j$, i.e.

$$-A_{ij} = \hat{K}_{,w}[w_i, v_i]\left(\tilde{T}_{ij} g'(u_i) + \delta_{ij}\right) - \hat{K}_{,u}[w_i, u_i]\delta_{ij}. \tag{68}$$

(We have defined $\tilde{T} \equiv -T$.)

So our optimization problem is to find $\hat{K}$ which solves the following optimization problem:

$$
\begin{aligned}
&\text{Maximize} \\
&\hat{\mathcal{O}} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \|A^{-1}(\mathbf{u},\tilde{T})\|_2^{-1} \\
&\text{with respect to (w. r.t.)} \\
&\hat{K}, \text{ subject to} \\
&\hat{\mathcal{C}} = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \max_{\hat{A}\subset A} \|\hat{A}\|_2 \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \hat{K}_{,wu} = \hat{K}_{,uw} \text{ and } \hat{K}_{,w} \ge 0 \text{ and } \hat{K}[0,u] = 0 \Big) \\
&\text{where} \\
&\qquad \tilde{\mathcal{T}} = \{\tilde{T} | \sigma_1(\tilde{T}) \le t_{\max} \text{ and } \tilde{T} \text{ is positive semi-definite}\}, \text{ and} \\
&\qquad -A_{ij} = \hat{K}_{,w}[w_i, v_i]\Big(\tilde{T}_{ij} g'(u_i) + \delta_{ij}\Big) - \hat{K}_{,u}[w_i, u_i]\delta_{ij}
\end{aligned}
\tag{69}
$$

and $\sigma_1(\tilde{T})$ is the largest singular value of $\tilde{T}$, i.e. the largest absolute value of any eigenvalue of 7'.

By introducing new notation

$$
\begin{aligned}
\mu[w, u] &= \hat{K}_{,w}[w, v] \\
\nu[w, u] &= -\hat{K}_{,u}[\mathbf{W}, v]
\end{aligned}
\tag{70}
$$

and translating the constraints appropriately, we can treat μ and ν as independent *functions* except for the constraint on the mixed partial derivatives. Then the problem (69) is equivalent to the following optimization problem:

$$
\begin{aligned}
&\text{Maximize} \\
&\hat{\mathcal{O}} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \|A^{-1}(\mathbf{u},\tilde{T})\|_2^{-1} \\
&\text{w.r. t. } (\mu,\nu), \\
&\text{subject to} \\
&\hat{\mathcal{C}} = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \max_{\hat{A}\subset A} \|\hat{A}\|_2 \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{,u} = -\nu_{,w} \text{ and } \mu \ge 0 \text{ and } \nu[0,u] = 0 \Big) \\
&\text{where} \\
&\qquad \tilde{\mathcal{T}} = \{\tilde{T} | \sigma_1(\tilde{T}) \le t_{\max} \text{ and } \tilde{T} \text{ is positive semi-definite}\}, \text{and} \\
&\qquad -A_{ij} = \mu[w_i, v_i]\tilde{T}_{ij} g'(u_i) + \delta_{ij}) + \nu[w_i, u_i]\delta_{ij}.
\end{aligned}
\tag{71}
$$

In the next section we will establish an approximate solution to this optimization problem: a (μ,ν) pair that satisfies all the constraints and comes within a factor of 2 of the globally optimal value of $\hat{\mathcal{O}}$. Here we simply make several observations about the optimization problem (71).

First, one of the most important questions about this problem, and our solution to it, is whether the restriction to positive semi-definite $\tilde{T}$'s can be removed. Connection matrices appearing in real applications can have bounded singular values, but rarely are all the eigenvalues of the same sign. Second, we note the close relation of this problem to a worst-case minimization of the condition number of *A, K(A)* = $\|A\|_2\|A^{-1}\|_2$. Since $\max_{ij}|a_{ij}| \le \|A\|_2$ and μ and ν can easily be rescaled by a constant while preserving their constraints, the two problems look quite similar. Indeed, maximizing *K(A)* over all $u, w, \tilde{T} \in \mathcal{T}$ subject to the p and ν constraints would yield an upper bound of $\tau_{\text{fast}}\kappa_{\max}$ for $\mathcal{O}_{\max}$. But our problem is more difficult because the extremization over $u, w, \tilde{T} \in \tilde{\mathcal{T}}$ is performed separately for the constraint and the objective.

3.2.1 Optimization of μ and ν

A useful auxiliary problem to (7 I) is obtained by replacing (54) with the non-invariant expression (57):

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \|A^{-1}(\mathbf{u},\tilde{T})\|_2^{-1} \\
&\text{w.r. t. } (\mathrm{p}, \nu), \\
&\text{subject to} \\
&\mathbf{C(c)} = \Big(c \max_{u,\ w,\tilde{T}\in\tilde{\mathcal{T}}} \max_{ij} |A_{ij}(\mathbf{u},\tilde{T})| \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{,u} = -\nu_{,w} \text{ and } \mu \ge 0 \text{ and } \nu[0,u] = 0\Big) \\
&\text{where} \\
&\qquad \tilde{\mathcal{T}} = \{\tilde{T} | \sigma_1(\tilde{T}) \le t_{\max} \text{ and } \tilde{T} \text{ is positive semi-definite}\}, \text{and} \\
&\qquad -A_{ij} = \mu[w_i, v_i]\Big(\tilde{T}_{ij} g'(u_i) + \delta_{ij}\Big) + \nu[w_i, u_i]\delta_{ij}.
\end{aligned}
\tag{72}
$$

Unlike the original problem (71), we will be able to solve this auxiliary problem exactly.

To solve the constrained maximization problem (72) and others like it, we will use the following proof strategy. Given objective $\mathcal{O}$ and constraints C, we will maximize some lower bound objective $\mathcal{O}_-$ such that $\mathcal{O}_-[\mu,\nu] \le \mathcal{O}[\mu,\nu]$, subject to *tightened* constraints $\mathcal{C}_-$ such that $\mathcal{C}_-[\mu,\nu] \Rightarrow \mathcal{C}[\mu,\nu]$. In this way we ensure that $\max(\mathcal{O}_-|\mathcal{C}_-) \le \max(\mathcal{O}|\mathcal{C})$. Likewise we will maximize some upper bound objective $\mathcal{O}_+$ such that $\mathcal{O}[\mu,\nu] \le \mathcal{O}_+[\mu,\nu]$, subject to loosened constraints C_+ such that $\mathcal{C}[\mu,\nu] \Rightarrow \mathrm{C}_+[\mu,\nu]$; this combination ensures that $\max(\mathcal{O}|\mathcal{C}) \le \max(\mathcal{O}_+|\mathcal{C}_+)$. Having solved both constrained optimization, we will see that both give the same value for the objective:

$$\max(\mathcal{O}_+|\mathcal{C}_+) = \max(\mathcal{O}_-|\mathcal{C}_-) \tag{73}$$

which implies that all the extremal values are the same:

$$\max(\mathcal{O}|\mathcal{C}) = \max(\mathcal{O}_-|\mathcal{C}_-) = \max(\mathcal{O}_+|\mathcal{C}_+). \tag{74}$$

Furthermore, the extremal values μ_-^* and ν_-^* of $\max(\mathcal{O}_-[\mu,\nu]|\mathcal{C}_-[\mu,\nu])$ all satisfy constraints C (since they satisfy $\mathcal{C}_-$) and thus constitute extremal values of $\max(\mathcal{O}[\mu,\nu])$ as well. Thus we will have solved the original constrained optimization problem of maximizing O with respect to C, by finding the maximal value and arguments μ^*, ν^*) at which the maximum is attained.

In the next section we will use this proof strategy to solve the auxiliary optimization problem *(72)*. A variant of the same argument can then be used to conclude that the (μ,ν) pair for the c = 2 auxiliary problem comes within a factor of two of solving the original optimization problem (71).

In fact, using (56), we see that the c = 1 version of (72) is a upper bound for (71) and the c = 2 version is an lower bound. In other words,

$$\max(\mathcal{O}|\mathcal{C}(c = 2)) \le \max(\hat{\mathcal{O}}|\hat{\mathcal{C}}) \le \max(\mathcal{O}|\mathcal{C}(c = 1)). \tag{75}$$

Furthermore, $\mathcal{C}(c=2)$ implies $\hat{\mathcal{C}}$ so that the extremal (p', ν^*) for $\max(\mathcal{O}|\mathcal{C}(c=2))$ are in the constraint set for $\max(\hat{\mathcal{O}}|\hat{\mathcal{C}})$. As it will turn out, $\max(\mathcal{O}|\mathcal{C}(c))$ is proportional to I/c, so $\hat{\mathcal{O}}(\mu^*, \mathrm{v}^*) = \mathcal{O}(\mu^*,\nu^*)$ is proven to be within a factor of two of its optimal value, $\max(\mathcal{O}|\mathcal{C})$. In other words,

$$\mathcal{O}(\mu^*,\nu^*) \equiv \max(\mathcal{O}|\mathcal{C}(c = 2)) \le \max(\hat{\mathcal{O}}|\hat{\mathcal{C}}) = 2\ \max(\mathcal{O}|\mathcal{C}(c = 2)) \tag{76}$$

which implies

$$(1/2)\ \max(\hat{\mathcal{O}}|\hat{\mathcal{C}}) \le \mathcal{O}(\mu^*,\nu^*), \equiv \max(\mathcal{O}|\mathcal{C}(c = 2)) \tag{77}$$

and (μ^*,ν^*) is an approximate solution (satisfying the constraints and optimizing the objective to within a factor of two) of the meta-optimization problem (71) or equivalently (69).

3.2.2 Solution of the Auxiliary Problem

We may solve the auxiliary problem for c = 1, then scale it to any other c by scaling τ_{fast} appropriately. So we'll assume $c=1$ in the following solution of (72), The basic strategy will be to obtain

upper bounds by restricting consideration to diagonal connection matrices $\tilde{T}$, and to compare these upper bounds with lower bounds that follow from matrix theory. In some cases, we will find it useful to repeat the above reasoning to solve the bounding constrained optimization problems themselves. For example, $\max(\mathcal{O}_{-}$ [[!-) will be found by way of $\max(\mathcal{O}_{--}|\mathcal{C}_{--})$ and $\max(\mathcal{O}_{-+}|\mathcal{C}_{-+})$. But first we will treat the upper bound $\max(\mathcal{O}_{+}|\mathcal{C}_{+})$.

By simply restricting the class T in problem (72) to the subset $\tilde{\mathcal{T}}_{+}$ of $\tilde{T}$ matrices which are also diagonal, we simultaneously increase the value of $\mathcal{O}[\mu,\nu]$ (since it's a minimum over a proper subset of $\tilde{T}\in\mathcal{T}$) and loosen the constraint $\mathcal{C}[\mu, u]$. So one lower bound optimization problem is:

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O}_{+1} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \|A^{-1}(\mathbf{u},\tilde{T})\|_2^{-1} \\
&\text{w.r.t. } (\mu,\nu), \\
&\text{subject to} \\
&\mathcal{C}_{+1} = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \max_{ij} |A_{ij}(\mathbf{u},\tilde{T})| \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{,u} = -\nu_{,w} \text{ and p} \ge \text{O and } \nu[0,u] = \text{O} \Big) \\
&\text{where} \\
&\qquad \tilde{\mathcal{T}}_{+} = \{\tilde{T}|\tilde{T} \text{ is diagonal and al } (\tilde{T}) \le t_{\max} \text{ and } \tilde{T} \text{ is positive semi-definite}\}, \text{ and} \\
&\qquad -A_{ij} = \mu[w_i, v_i]\big(\tilde{T}_{ij} g'(u_i) + \delta_{ij}\big) + \nu[w_i, u_i]\delta_{ij}.
\end{aligned}
\tag{78}
$$

This will not be the sought-after $\mathcal{O}_{+}$ and $\mathcal{C}_{+}$, but it moves in the right direction since O $\le \mathcal{O}_{+1}$ and C $\Rightarrow \mathcal{C}_{+1}$.

If $\tilde{T}$ is diagonal then so is A. For a diagonal matrix $A = \text{diag}(a_i)$, $\|A^{-1}\|^{-1} = \min_i |a_i|$ and $\max_{ij}|A_{ij}| = \max_i |a_i|$. So we can calculate more detailed bounds:

$$
\begin{aligned}
\mathcal{O}_{+1} &= \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \min_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1) + \nu_i\right| \\
&\le \min_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \min_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1) + \nu_i\right|_{w=0} \\
&= \min_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \min_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1)\right|_{w=0} && (\text{since } \nu[0,u] = O) \\
&= \min_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \min_i \mu_i(|\tilde{T}_{ii}| g_i' + 1)\Big|_{w=0} \\
&= \min_u \min_i \mu_i\Big(\big(\min_{\tilde{T}\in\tilde{\mathcal{T}}_{+}} |\tilde{T}_{ii}|\big) g_i' + 1\Big)\Big|_{w=0} && (\text{since } \mu \ge 0 \text{ and } g_i' \ge 0) \\
&= \min_u \min_i \mu[0, u_i] && (\text{since } \min_{\tilde{T}\in\tilde{\mathcal{T}}_{+}} |\tilde{T}_{ii}| = 0) \\
&= \min_u \mu[0, u] \\
&\equiv \mathcal{O}_{+}[\mu,\nu].
\end{aligned}
\tag{79}
$$

Likewise we can bound the main constraint of $\mathcal{C}_{+1}$, which is that $\hat{\mathcal{C}}_{+1} \le 1/\tau_{\text{fast}}$, where

$$
\begin{aligned}
\hat{\mathcal{C}}_{+1} &= \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \max_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1) + \nu_i\right| \\
&\ge \max_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \max_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1) + \nu_i\right|_{w=0} \\
&= \max_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \max_i \left|\mu_i(\tilde{T}_{ii} g_i' + 1)\right|_{w=0} && (\text{since } \nu[0,u] = 0) \\
&= \max_{u,\tilde{T}\in\tilde{\mathcal{T}}_{+}} \max_i \mu_i(|\tilde{T}_{ii}| g_i' + 1)\Big|_{w=0} \\
&= \max_u \max_i \mu_i\Big(\big(\max_{\tilde{T}\in\tilde{\mathcal{T}}_{+}} |\tilde{T}_{ii}|\big) g_i' + 1\Big)\Big|_{w=0} && (\text{since } \mu \ge 0 \text{ and } g_i' \ge 0) \\
&= \max_u \max_i \mu[0, u_i]\Big(t_{\max} g'(u_i) + 1\Big) && (\text{since } \max_{\tilde{T}\in\tilde{\mathcal{T}}_{+}} |\tilde{T}_{ii}| = t_{\max}) \\
&= \max_u \mu[0, u]\Big(t_{\max} g'(u) + 1\Big) \\
&\equiv \mathcal{C}_{+}[\mu,\nu].
\end{aligned}
\tag{80}
$$

So the upper bound optimization problem becomes

$$
\begin{aligned}
&\text{Maximize} \\
&\quad \mathcal{O}_+ = \min_u \mu[0, \text{If}] \\
&\text{w.r.t. } (\mu, \nu), \\
&\text{subject to} \\
&\quad \mathcal{C}_+ = \Big(\max_u \mu[0,u] \cdot (t_{\max} g'(u) + 1) \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{w,u} = -\nu_{w} \text{ and } \mu \ge 0 \text{ and } \nu[0,u] = 0\Big)
\end{aligned}
\tag{81}
$$

To this optimization problem we propose the solution (μ_+^*, ν_+^*):

$$
\begin{aligned}
\mu_+^*[w,u] &= 1/\tau_{\text{fast}}\ (t_{\max} g_0 + 1) \\
\nu_+^*[w,u] &= 0,
\end{aligned}
\tag{82}
$$

where $g_0 \equiv \max_u g'(u)$. These values for μ and ν are *constant*, i.e. independent of w and u, so the mixed partial derivative constraint of problem (81) is trivially satisfied. Clearly also $\mu_+^* \ge 0$ and $\nu_+^*[0,u] = 0$ are satisfied. The $\hat{\mathcal{C}}_+ \le 1/\tau_{\text{fast}}$ constraint can also be verified:

$$
\hat{\mathcal{C}}_+[\mu_+^*, \nu_+^*] = \max_u \mu_+^*[0,u]\Big(t_{\max} g'(u) + 1\Big) = \frac{\max_u \Big(t_{\max} g'(u) + 1\Big)}{\tau_{\text{fast}}(t_{\max} g_0 + 1)} = 1/\tau_{\text{fast}}. \tag{83}
$$

So (μ_+^*, ν_+^*) satsifies the desired constraints. The objective is $\mathcal{O}_+[\mu_+^*, \nu_+^*] = \min_u \mu_+^*[0,u] = 1/\tau_{\text{fast}}(t_{\max} g_0 + 1)$. But from the constraints we see this value is also an upper bound for $\mathcal{O}_+[\mu, \nu]$ as follows:

$$
\begin{aligned}
1/\tau_{\text{fast}} &\ge \hat{\mathcal{C}}_+[\mu, \nu] \\
&= \max_u \mu[0,u]\Big(t_{\max} g'(u) + 1\Big) \\
&\ge (\min_u \mu[0,u])\Big(\max_u (t_{\max} g'(u) + 1)\Big) \\
&= \mathcal{O}_+[\mu, \nu]\Big(t_{\max} g_0 + 1\Big),
\end{aligned}
\tag{84}
$$

which implies $\mathcal{O}_+ \le 1/\tau_{\text{fast}}(t_{\max} g_0 + 1)$. So (μ_+^*, ν_+^*) in *(82)* solves problem (81).

Next, we use matrix theory to find and solve a constrained optimization problem $\max(\mathcal{O}_- | \mathcal{C}_-)$ which can serve as a lower bound for $\max(\mathcal{O}|\mathcal{C})$.

To bound $\mathcal{O}$ below (in problem (72)), we must simplify $||A^{-1}||_2^{-1}$. In matrix notation, $||A^{-1}||_2^{-1}$ is just $\sigma_n(A)$, the smallest singular value of A. Also A is given by the matrix expression

$$
A = \text{diag}(\mu)(\tilde{T}\text{diag}(g') + 1) + \text{diag}(\nu). \tag{85}
$$

The smallest singular value $\sigma_n(M + N)$ of a sum of matrices M and N is bounded below by $\sigma_n(M) - \sigma_1(N)$, as shown for example in [GL83] (Cor. 8.3-2, p.286). We will take $A = M + N$ with $N = \text{diag}(\nu)$ and use $\sigma_1(\text{diag}(\nu)) = \max_i |\nu_i|$ to find a lower bound $\mathcal{O}_-$ for $\mathcal{O}$:

$$
\mathcal{O} \ge \mathcal{O}_- \equiv \min_{u,w,\tilde{T} \in \mathcal{T}} \Big[\sigma_n\Big(\text{diag}(\mu)(\tilde{T}\text{diag}(g') + 1)\Big) - \max_i |\nu_i|\Big]. \tag{86}
$$

We can also bound the main constraint of $\mathcal{C}$, which is that $\hat{\mathcal{C}} \le 1/\tau_{\text{fast}}$. We will use the fact that $\sigma_1(M + N) \le \sigma_1(M) + \sigma_1(N)$, which is also shown in [GL83] (Cor. 8.3-2, p.286). The bound is as follows:

$$
\begin{aligned}
\hat{\mathcal{C}}[\mu, \nu] &= \max_{u,w,\tilde{T} \in \mathcal{T}} \max_{ij} |A_{ij}| \\
&\le \max_{u,w,\tilde{T} \in \mathcal{T}} ||A||_2 \\
&\qquad \text{(standard matrix norm bounds, e.g. [GL83], 2.2-10, p. 15)} \\
&= \max_{u,w,\tilde{T} \in \mathcal{T}} \sigma_1(A) \\
&\le \max_{u,w,\tilde{T} \in \mathcal{T}} \Big[\sigma_1\Big(\text{diag}(\mu)(\tilde{T}\text{diag}(g') + 1)\Big) + \max_i |\nu_i|\Big] \\
&\equiv \hat{\mathcal{C}}_-[\mu, \nu].
\end{aligned}
\tag{87}
$$

So the lower bound optimization problem becomes

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O}_- = \min_{14,11),7\text{-}67} \left[\sigma_n\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\right) - \max_i |\nu_i|_1 \\
&\text{w.r.t. } (\mu, \nu), \\
&\text{subject to} \\
&\mathcal{C}_- = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \left[\sigma_1\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) + \max_i |\nu_i|\right] \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{,u} = -\nu_{,w} \text{ and } \mu \ge \mathrm{O} \text{ and } \nu[0,u] = \mathrm{O}).
\end{aligned}
\tag{88}
$$

Consider the related optimization problem

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O}_{-+1} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \left[\sigma_n\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) - \max_i |\nu_i|\right] \\
&\text{w.r.t. } (\mu, \nu), \\
&\text{subject to} \\
&\mathcal{C}_{-+1} = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \left[\sigma_1\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) + \max_i |\nu_i|\right] \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu \ge \mathrm{O} \text{ and } \nu[0,u] = \mathrm{O}),
\end{aligned}
\tag{89}
$$

which differs from (88) by removing the partial derivative constraint that relates μ and ν. Clearly if we solve this problem and find a solution that also obeys the partial derivative constraint, then we will have solved the original problem. That is what we will do. But the new problem (89) can be further simplified by observing that the optimal ν^*_{-+1} must be identically zero; otherwise, an optimal $(\mu^*_{-+1}, \nu^*_{-+1} \# \mathrm{O})$ would have a lower value of the objective than $(\mu^*_{-+1}, \mathrm{O})$ which equally well satisfies the constraint C_{-+1}; that would contradict the assumed optimality of $(\mu^*_{-+1}, \nu^*_{-+1} \# \mathrm{O})$.

So to solve $\max(\mathcal{O}_-|\mathcal{C}_-)$, i.e. problem (88), it suffices to (a) solve problem (89) assuming u = O, i.e. to solve:

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O}_{-+} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) \\
&\text{w.r.t. } (\mu, \nu), \\
&\text{subject to} \\
&\mathcal{C}_{-+} = \Big(\max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_1 \mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu \ge 0,
\end{aligned}
\tag{90}
$$

and then (b) verify that the mixed derivative constraint $\mu_{,u} = -\nu_{,w}$, $(= \mathrm{O})$ is satisfied by the solution (μ^*_{-+}, O) to *(90)*. Furthermore, the optimizing values (μ^*_-, ν^*_-) will just be (μ^*_{-+}, O).

We will solve $\max(\mathcal{O}_{-+}|\mathcal{C}_{-+})$ using the same strategy as for $\max(\mathcal{O}|\mathcal{C})$ itself: by constructing an upper bound problems by restricting to diagonal connection matrices $\tilde{T} \in \tilde{\mathcal{T}}_+ = \tilde{\mathcal{T}} \cap$ {diagonal matrices}, and a lower bound problem using more matrix theory, and showing that they have a common solution.

The upper bound for $\mathcal{O}_{-+}$ is calculated as follows:

$$
\begin{aligned}
\mathcal{O}_{-+} &= \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) \\
&\le \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \sigma_n\Big(\mathrm{diag}(\mu)(\tilde{T}'\mathrm{diag}(g') + 1)\Big) \\
&= \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \min_i \mu_i(\tilde{T}'_{ii} g'_i + 1) \\
&\le \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \min_i \mu_i(|\tilde{T}'_{ii}| g'_i + 1) \\
&= \min_{u,w} \min_i \mu_i\Big(\min_{\tilde{T}\in\tilde{\mathcal{T}}_+} |\tilde{T}'_{ii}| g'_i + 1\Big) \\
&= \min_{u,w} \min_i \mu[w_i, u_i] \qquad \text{(since } \min_{\tilde{T}\in\tilde{\mathcal{T}}_+} |\tilde{T}'_{ii}| = 0) \\
&= \min_{u,w} \mu[w, u] \\
&\equiv \mathcal{O}_{-++}.
\end{aligned}
\tag{91}
$$

The corresponding (lower) bound for $\mathcal{C}_{-+}$ is calculated as follows:

$$
\begin{aligned}
\mathcal{C}_{-+} &= \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_1\Big(\mathrm{diag}(\mu)(\tilde{T}\mathrm{diag}(g')+1)\Big) \\
&\geq \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \sigma_1\Big(\mathrm{diag}(\mu)(\tilde{T}\mathrm{diag}(g')+1)\Big) \\
&= \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \max_i \mu_i(\tilde{T}_{ii}g'_i+1) \\
&= \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}_+} \min_i \mu_i(\tilde{T}_{ii}g'_i+1) \qquad \text{(since } \tilde{T}_{ii}\geq 0) \\
&= \max_{u,w} \min_i \mu_i\Big(\max_{\tilde{T}\in\tilde{\mathcal{T}}_+} \tilde{T}_{ii}g'_i+1\Big) \\
&= \max_{u,w} \min_i \mu[w_i,u_i]\sim n,axg'(it,)\ +1) \qquad (\text{since } \max_{\tilde{T}\in\tilde{\mathcal{T}}_+} \tilde{T}_{ii} = t_{\max}) \\
&= \max_{u,w} \mu[w,u]\Big(t_{\max}g'(u)+1\Big) \\
&\equiv \hat{\mathcal{C}}_{-++}.
\end{aligned}
\tag{92}
$$

So the upper bound optimization problem becomes similar to problem (81):

$$
\begin{aligned}
&\text{Maximize} \\
&\quad \mathcal{O}_{-++} = \min_{u,w} \mu[w,u] \\
&\text{w.r.t. } (\mu,\nu), \\
&\text{subject to} \\
&\quad \mathcal{C}_{-++} = \Big(\max_{u,w} \mu[w,u]\Big(t_{\max}g'(u)+1\Big) \leq 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu \geq 0\Big)
\end{aligned}
\tag{93}
$$

To this optimization problem we again propose the solution (cf. equation)

$$
\mu^*_{-++}[w,u] = 1/\tau_{\text{fast}}(t_{\max}g_0+1), \tag{94}
$$

where $g_0 \equiv \max_u g'(u)$. The proof for this solution is the same as that of the solution of problem (81) by equation (82), except that now w must be optimized everywhere u is. This establishes the solution of problem (93) by equation (94).

We must now find a lower bound $\max(\mathcal{O}_{-+-}|\mathcal{C}_{-+-})$ for $\max(\mathcal{O}_{-+}|\mathcal{C}_{-+})$, and to do so, we require another matrix theory result: that for positive semi-definite matrices M and N, $\sigma_n(M+N) \geq \sigma_n(M)+\sigma_n(M)$ [SgS90].

(Note on the proof so far: We could not use this result earlier since $\mathrm{diag}(\nu)$ was not positive semi-definite. Also the use of this result and equation (92) are the only places in the proof that depend on the assumption that $\tilde{T}$ is positive sen~i-definite.)

Thus,

$$
\begin{aligned}
\mathcal{O}_{-+} &= \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n\Big(\mathrm{diag}(\mu)\tilde{T}\mathrm{diag}(g')+\mathrm{diag}(\mu)\Big) \\
&\geq \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n\Big(\mathrm{diag}(\mu)\tilde{T}\mathrm{diag}(g')\Big) + \sigma_n(\mathrm{diag}(\mu))\Big] \\
&\geq \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \Big[\sigma_n(\mathrm{diag}(\mu))\sigma_n(\tilde{T})\sigma_n(\mathrm{diag}(g')) + \sigma_n(\mathrm{diag}(\mu))\Big] \\
&\qquad (\text{since } \|MN\|_2 \leq \|M\|_2\|N\|_2, [\text{GL83}]\text{ p.16}) \\
&= \min_{u,w} \Big[\sigma_n(\mathrm{diag}(\mu))\Big(\min_{\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n(\tilde{T})\Big)\sigma_n(\mathrm{diag}(g')) + \sigma_n(\mathrm{diag}(\mu))\Big] \\
&= \min_{u,w} \min_i \mu[w_i,u_i] \\
&\qquad (\text{since } \min_{\tilde{T}\in\tilde{\mathcal{T}}} \sigma_n(\tilde{T}) = 0) \\
&= \min_{u,w} \mu[w,u] \\
&\equiv \mathcal{O}_{-+-}[\mu].
\end{aligned}
\tag{95}
$$

Likewise,

$$
\begin{aligned}
\mathcal{C}_{-+} &= \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \sigma_1\Big(\mathrm{diag}(\mu)\tilde{T}\mathrm{diag}(g') + \mathrm{diag}(\mu)\Big) \\
&\le \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \Big[\sigma_1\Big(\mathrm{diag}(\mu)\tilde{T}\mathrm{diag}(g')\Big) + \sigma_1(\mathrm{diag}(\mu))\Big] \\
&\qquad \text{(since } \|M+N\|_2 \le \|M\|_2 + \|N\|_2\text{, [GL83] Cor 8.3-2)} \\
&\le \max_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \Big[\sigma_1(\mathrm{diag}(\mu))\sigma_1(\tilde{T})\sigma_1(\mathrm{diag}(g')) + \sigma_1(\mathrm{diag}(\mu))\Big] \\
&\qquad \text{(since } \|MN\|_2 \le \|M\|_2\|N\|_2\text{, [GL83] p.16)} \\
&= \max_{u,w}\sigma_1(\mathrm{diag}(\mu))\Big(\Big(\max_{\tilde{T}\in\tilde{\mathcal{T}}}\sigma_1(\tilde{T})\Big)\sigma_1(\mathrm{diag}(g')) + 1\Big) \\
&= \max_{u,w}(\max_i \mu[w_i,u_i])\Big(t_{\max}(\max_i g'(u_i) + 1\Big) \\
&\qquad \text{(since } \max_{\tilde{T}\in\tilde{\mathcal{T}}}\sigma_1(\tilde{T}) = t_{\max}\text{)} \\
&\equiv \hat{\mathcal{C}}_{-+-}[\mu].
\end{aligned}
\tag{96}
$$

We can assemble these bounds into the constrained optimization problem

$$
\begin{aligned}
&\text{Maximize} \\
&\quad \mathcal{O}_{-+-} = \min_{u,w} \mu[w,u] \\
&\text{w.r.t. } (\mu,\nu), \\
&\text{subject to} \\
&\quad \mathcal{C}_{-+-} = \Big(\max_{u,w}(\max_i \mu[w_i,u_i])\Big)\Big(t_{\max}(\max_i g'(u_i) + 1\Big) \le 1/\tau_{\text{fast}} \\
&\qquad\qquad \text{and } \mu \ge 0\Big).
\end{aligned}
\tag{97}
$$

To this optimization problem we once again propose the constant solution (cf. equation)

$$
\mu^*_{-+-}[w,u] = 1/\tau_{\text{fast}}(t_{\max}g_0 + 1), \tag{98}
$$

where $g_0 \equiv \max_u g'(u)$. Clearly the constraint $\mu^*_{-+-} \ge 0$ is satisfied. The $\hat{\mathcal{C}}_{-+-} \le 1/\tau_{\text{fast}}$ constraint can also be verified:

$$
\hat{\mathcal{C}}_{-+-}[\mu^*_{-+-}] = \max_{w,u}\mu^*_{-+-}[w,u]\Big(t_{\max}g'(u) + 1\Big) = \frac{\Big(t_{\max}g'(u) + 1\Big)}{\tau_{\text{fast}}(t_{\max}g_0+1)} = 1/\tau_{\text{fast}}. \tag{99}
$$

So μ^*_{-+-} satsifies the desired constraints. The objective is $\mathcal{O}_{-+-}[\mu^*_{-+-}] = \min_{w,u}\mu^*_{-+-}[w,u] = 1/\tau_{\text{fast}}(t_{\max}g_0 + 1)$. But, once again, from the constraints we know this value is also an upper bound for $\mathcal{O}_{-+-}[\mu]$:

$$
\begin{aligned}
1/\tau_{\text{fast}} &\ge \hat{\mathcal{C}}_{-+-}[\mu] \\
&= \max_{w,u}\mu[w,u]\Big(t_{\max}g'(u) + 1\Big) \\
&\ge (\min_{w,u}\mu[w,u])\Big(\max_{w,u}(t_{\max}g'(u) + 1)\Big) \\
&= \mathcal{O}_{-+-}[\mu,\nu]\,(t_{\max}g_0 + 1),
\end{aligned}
\tag{100}
$$

which implies $\mathcal{O}_{-+-} \le 1/\tau_{\text{fast}}(t_{\max}g_0 + 1)$. So equation μ^*_{-+-} in (98) solves problem (97).

We have previously solved problem μ^*_{-++} in (93) with equation (94). The resulting maximal values of $\mathcal{O}$ are the same for the two problems (97) and (93) ($\max(\mathcal{O}_{-+-}|\mathcal{C}_{-+-}) = \max(\mathcal{O}_{-++}|\mathcal{C}_{-++}) = 1/\tau_{\text{fast}}(t_{\max}g_0 + 1)$), and they are attained by the same μ^* = constant functions. Since these were lower and upper bounds for $\max(\mathcal{O}_{-+}|\mathcal{C}_{-+})$, we conclude that the same p* and maximal value of $\mathcal{O}$ also solve problem (90), namely the calculation of $\max(\mathcal{O}_{-+}|\mathcal{C}_{-+})$. But in the discussion of problem (90) we pointed out that, if $\mu^*_{-+,u}$ = O (as it certainly is, since μ^*_{-+} is a constant independent of both u and w), then (μ^*_{-+}, u = O) is also a solution (μ^*_-, ν^*_-) of problem (88). This result is the sought--after lower bound for the original problem (72), and may be joined with the solution of (81) (an upper bound for (72)) by (82) to finish the entire problem: $\max(\mathcal{O}_-|\mathcal{C}_-) = \max(\mathcal{O}|\mathcal{C}) = \max(\mathcal{O}_+|\mathcal{C}_+) = 1/\tau_{\text{fast}}(t_{\max}g_0 + 1)$; and the optimum is attained at (p", ν^*) = (μ^*_-, ν^*_-) = (μ^*_+, ν^*_+), i.e.

$$
\begin{aligned}
\mu^*[711, {}^{\text{"}}11] &= 1/\tau_{\text{fast}}(t_{\max}g_0 + 1) \\
\nu^*[w,u] &= 0
\end{aligned}
\tag{101}
$$

is shown to be a solution of (72) for c = 1. Other values of c may be absorbed into the definition of τ_{fast}. So we have established Lemma 1:

Lemma 1. The optimization problem

$$
\begin{aligned}
&\text{Maximize} \\
&\mathcal{O} = \min_{u,w,\tilde{T}\in\tilde{\mathcal{T}}} \|A^{-1}(\mathbf{u},\tilde{T})\|_2^{-1} \\
&\text{w.r.t } (\mu,\nu), \\
&\text{subject to} \\
&\mathrm{C}(c) = \Big(\max_{u,\,w,\,\tilde{T}\in\tilde{\mathcal{T}}} c \max_{ij} |A_{ij}(\mathbf{u},T)| \le 1/\tau_{\text{fast}} \\
&\qquad \text{and } \mu_{,u} = -\nu_{,w} \text{ and } \mu \ge \mathrm{O} \text{ and } \nu[0,\mathrm{u}] = \mathrm{O}) \\
&\text{where} \\
&\qquad \tilde{\mathcal{T}} = \{\tilde{T} | \sigma_1(\tilde{T}) \le t_{\max} \text{ and } \tilde{T} \text{ is positive senli-definite}\}, \text{ and} \\
&\qquad -A_{ij} = \mu[w_i, v_i]\,(\tilde{T}_{ij} g'(u_i) + \delta_{ij}) + \nu[w_i, u_i]\delta_{ij}
\end{aligned}
\tag{102}
$$

has as one solution

$$
\begin{aligned}
\mu^*[\mathrm{w}, u] &= 1/(c\tau_{\text{fast}}(t_{\max} g_0 + 1)) \\
\nu^*[w, u] &= \mathrm{o}.
\end{aligned}
\tag{103}
$$

It remains only to translate this solution for $\mu[w,u]$ and $\nu[w,u]$ back into a function $\hat{K}$ (as called for in *(69))* and thence to the desired "kinetic energy" or "cost of movement" function $\tilde{K}[\dot{u},u]$ or its equivalent, $K[\dot{v}, \mathrm{v}]$.

3.2.3 Approximate Solution of the Meta-Optimization Problem

¿From equation (77), we can apply Lemma 1 with c = 2 to find a (p", ν^*) pair which comes within a factor of 2 of solving the meta-optimization problem (71) or equivalently (69). (Note that (77) was derived assuming that $\max(\mathcal{O}|\mathcal{C}(c))$ is proportional to I/c, which has now been established in Lemma 1.) Changing back to $\hat{K}$ notation,

$$
\hat{K}_{,w} = 1/\tau_H, \quad \hat{K}_{,u} = \mathrm{o}, \tag{104}
$$

where

$$
\tau_H = 2\tau_{\text{fast}}(t_{\max} g_0 + 1) \tag{105}
$$

is a constant. (The factor of 2 comes from c = 2.) The general solution of these partial differential equations is $\hat{K}[w,u] = w/\tau_H$ -t- c_1, but from the statement of problem (69) we must take $\hat{K}[0, u] = c_1 = \mathrm{O}$. Then

$$
\hat{K}[w,u] = w/\tau_H. \tag{106}
$$

Using (65),

$$
\tilde{K}_{,\dot{u}}[\dot{u}, \mathrm{u}] = \hat{K}^{-1}[\dot{u},u]g'(u) = \tau_H \dot{u} g'(u). \tag{107}
$$

This has the solution $\tilde{K}[\dot{u}, \mathrm{u}] = (\tau_H/2)\dot{u}^2 g'(u) + c_2(u)$. But the term $c_2(u)$ has no effect on the dynamics, since its greedy derivative is zero, and without loss of generality we can take $c_2(u) = \mathrm{O}$. Then

$$
\tilde{K}[\dot{u},u] = \frac{\tau_H}{2}\dot{u}^2 g'(u). \tag{108}
$$

This is the sought-after kinetic energy or cost term for $\dot{u}$, and the associated equation of motion is (from equation (63))

$$
\begin{aligned}
\dot{u}_i &= \frac{-1}{\tau_H}\Big(\sum_j T_{ij} v_j + h_i - \mathrm{U}_{,i}\Big), \\
v_i &= g(u_i).
\end{aligned}
\tag{109}
$$

This $\tilde{K}$ may also be translated back to a Lagrangian expressed directly in terms of ii, using $K[\dot{v}, v] = \tilde{K}[\dot{u}(v), u(v)]$:

$$
K[\dot{v}, v] = \frac{\tau_H}{2}\dot{v}^2/g'(g^{-1}(v)), \tag{110}
$$

or equivalently

$$K[\dot{v}, v] = \frac{\tau_H}{2}\dot{v}^2\phi''(v). \tag{111}$$

If $g(u)$ is linear (i.e. if $\phi(v)$ is quadratic), this kinetic energy expression is proportional to the conventional $(m/2)\dot{v}^2$ expression encountered in physics, but for nonlinear g this expression is different from a kinetic energy in physics, (110) is the circuit cost-of-movement (or kinetic energy) term used elsewhere in this paper, and a greedy variation of the associated action functional yields equations of motion equivalent to the Hopfield/Grossberg dynamics of (109).

Assembling Lemma 1 and (47), (48), (52), (58), (61), (77), (105), and (1 11), we have demonstrated the following theorem:

Theorem 1. The linearized dynamics determined by a greedy variation of the Lagrangian

$$\begin{aligned} L[\dot{\mathbf{v}}] &= \textstyle\sum_i K[\dot{v}_i, v_i] + \sum_i E_{,i}\dot{v}_i, \text{ with} \\ E[\mathbf{v}] &= -\tfrac{1}{2}\textstyle\sum_{ij} T_{ij}v_iv_j - \sum_i h_iv_i + \sum_i \phi(v_i), \text{ and} \\ & \phi'(v) = g^{-1}(v) \text{ and } g_0 = \max_u |g'(u)| \end{aligned} \tag{112}$$

may be computed to be

$$\begin{aligned} \Delta\dot{v}_i &= \tilde{K}[w_i, v_i] + \textstyle\sum_j A_{ij}\Delta v_j \\ \text{where} & \\ A_{ij} &= \tilde{K}_{,w}[w_i, v_i]\Big(T_{ij} - \delta_{ij}\phi''(v_i)\Big) + \tilde{K}_{,v}\delta_{ij}, \text{ and} \\ & w_i \equiv -E_{,i}, \text{ and } \tilde{K}[K[\dot{v}, v], v] = \dot{v}. \end{aligned} \tag{113}$$

If we define the objective

$$\mathcal{M}_\tau(K) = \min_{\mathbf{v}} \min_{T\in\mathcal{T}} ||A^{-1}(\mathbf{v}, T)||_2^{-1}, \tag{114}$$

where

$$\mathcal{T} = \{T | \sigma_1(T) \le t_{\max} \textbf{ and } T \text{ is negative semi-definite}\}, \tag{115}$$

and if we impose the constraints on K that

$$\begin{aligned} &(a) && \max_{\mathbf{v}} \max_{T\in\mathcal{T}} \max_{\hat{A}\subset A} ||\hat{A}||_2 \le 1/\tau_{\text{fast}}, \\ & && \text{(where } \hat{A} \text{ runs over 1 x 1 and 2 x 2 submatrices of A), and} \\ &(b) && K \text{ is continuous in its first and second derivatives,} \\ &(c) && \tilde{K}[0, v] = \text{O, and} \\ &(d) && \tilde{K}[w, v]_{,w} \le 0, \end{aligned} \tag{116}$$

then the function

$$\begin{aligned} K[\dot{v}, v] &= (\tau_H/2)\dot{v}^2\phi''(v) \text{ where} \\ \tau_H &= 2\tau_{\text{fast}}(t_{\max}g_0 + 1) \end{aligned} \tag{117}$$

satisfies the constraints and comes within a factor of two of the globally maximal value of $\mathcal{M}(K)$ subject to these constraints. Furthermore, the objective $\mathcal{M}_\tau$ and the constraints (a) – (d) in (1 16), with definitions of A, $\tilde{K}$ and w as in (113) are invariant with respect to coordinatewise reparameterizations $x_i = f_i(v_i)$ in which each f_i is monotonically increasing, differentiable, and has a differentiable inverse.

3.2.4 Notes on the Solution

If ϕ differs from one neuron to the next, and is indexed by i as ϕ_i, then the optimal K term will still have the above form if it too is allowed to depend on i. The proof in section 3.2.2 can easily be altered to establish this generalization of the result.

Note that (105) relates the fastest physical time scale τ_{fast} in a circuit to an optimal value of the neural time scale τ_H appearing in Hopfield's version of the analog *neural* network [Hop84], ancl the two are not the same. The best value for the neural time constant is the slowest time constant in the system. The ratio of the latter to the fastest time constant is roughly the product of the neural gain g_0 and largest eigenvalue of $\tilde{T}$.

We note a change of variable which simplifies the kinetic energy term in the above dynamics, for use in the next section:

$$\begin{aligned} L[\dot{\mathbf{w}}] &= \textstyle\sum_i \frac{1}{2}\dot{w}_i^2 + \sum_i \frac{\partial E}{\partial w_i}\dot{w}_i, \\ \partial L/\partial \dot{w}_i(t) &= 0 \Rightarrow \dot{w}_i + \partial E/\partial w_i = 0, \text{ i.e.} \\ \dot{w}_i &= -\partial E/\partial w_i \end{aligned} \tag{118}$$

which is supposedtobe identical to $\dot{u}_i = -\partial E/\partial v_i$, $v_i = g(u_i)$ (cf. (12)). This can be arranged by choosing w:

$$\begin{aligned} \frac{dw_i}{du_i}\dot{u}_i &= \frac{\partial E}{\partial u_i}\frac{dv_i}{dw_i} \\ \Rightarrow \frac{dw_i}{du_i} &= \frac{dv_i}{dw_i} = \frac{dv_i}{du_i}/\frac{dw_i}{du_i} \\ \Rightarrow \frac{dw_i}{du_i} &= \sqrt{g'(u_i)} \end{aligned} \tag{119}$$

i.e.

$$w_i = \int^{u_i} du\sqrt{g'(u)} \text{ and } v_i = \int^{w_i} dw\sqrt{g'(u(w))}. \tag{120}$$

4 Discussion and Conclusions

We introduced a Lagrangian formulation of the relaxation dynamics of neural networks which compute by optimizing an objective function in a standard neural network form. This optimization involves a trading-off cost and functionality in the formulation of optimization problems. 'The Lagrangian formulation makes novel use of a *greedy functional derivative*, which we defined and computed. With these tools we demonstrated the use of three levels of optimization in the design of relaxation neural network dynamics: the original objective E, the Lagrangian L, and a meta-objective M which measures cost and functionality over many trials of the network.

Applications of the Lagrangian formulation were divided into two broad groups: analog circuit Lagrangians, and Lagrangians that require a hidden switching mechanism to implement as a circuit. At the circuit level, we showed that a limited meta-optimality criterion is nearly optimized (within a factor of two of the global optimum) by a Lagrangian corresponding to the conventional Hopfield-Grossberg continuous-time analog neural network dynamics; we also provided several alternative Lagrangians which might be preferable under less analytically tractable meta-optimality criteria. In part 11 of this work we shall introduce a generalization of such relaxation Lagrangians to cyclic Lagrangians with clocked objective functions, which have a simple circuit implementation involving external clock signals. We shall present suitable algebraic notation including a *clocked sum* and *clamped variables* and use the notation concisely to express neural network dynamics for a variant of line minimization and for relaxation networks that contain feed-forward networks.

Acknowledgements

We wish to acknowledge Charles Garrett and K. Srinivas for unpublished simulations; also discussions with Roger Smith, Chien-Ping Lu, Anand Rangarajan, Paul Cooper, Stan Eisenstat and Alain Martin; also the hospitality of the Institute for Theoretical Physics at Santa Barbara. This research was supported in part by AFOSR grant AFOSR-88-0240 and by ONR grant NOO014-92-J-4048.

References

[BH89] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation,* 1(1):151–160, Spring 1989.

[BK93] Joachim Buhmann and Hans Kuhnel. Complexity optimized data clustering by competitive neural networks. *Neural Computation,* 5(1):75-88, January 1993.

[11'1'89] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation,* chapter 3, pages 210-217. Prentice Hall, 1989.

[DW87] R. Durbin and D. Willshaw. An analog approach to the traveling salesman problem using an elastic net method. *Nature,* 326:689691, 1987.

[GL83] Gene 11. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1983.

[Gro88] Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, I: 17 61, 1988.

[GY91] D. Geiger and A. L. Yuille. A common framework for image segmentation. *International Journal of* Computer *Vision*, 6(3):227 243, August 1091.

[Hop84] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, vol. 81:3088-3092, May 1984.

[HT85] J. J. Hopfield and D. W. Tank. 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, vol. 5 2:141-152, 1985.

[KMY86] Christof Koch, Jose Marroquin, and Alan Yuille. Analog "neuronal" networks in early *vision. Proceedings of the National Acadamy of Sciences USA, 83,* June 1986.

[KY91] J. J. Kosowsky and A. L. Yuille. The invisible hand algorithm: Solving the assignment problem with statistical physics '1'ethnical Report 91-1, Harvard Robotics laboratory, 1991.

[MG90] Eric Mjolsness and Charles Garrett. Algebraic transformations of objective functions *Neural Networks, 3:651-669,* 1990.

[Mjo85] Eric Mjolsness. *Neural Networks, Pattern Recognition, and Fingerprint Hallucination* PhD thesis, California Institute of 'Technology, 1985. See section 11.3.1 for wiring cost.

[Mjo87] Eric Mjolsness. Control of attention in neural networks. In *Proc. of First International Conference on Neural Networks,* volume vol. II, pages 567-574. IEEE, 1987.

[MM91] Eric Mjolsness and Willard L. Miranker. A Lagrangian approach to fixed points. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Neural Information* Processing *Systems 5'*. Morgan Kaufmann, 1991.

[Ner70] Evar D. Nering. *Linear Algebra and Matrix Theory.* John Wiley & Sons, Inc, second edition, 1970.

[PB87] John C. Platt and Alan H. Barr. Constrained differential optimization. In Dana Z. Anderson, editor, Neural information *Processing Systems.* American Institute of Physics, 1987.

[PS89] C. Peterson and B. Soderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems, 1(3), 1989.*

[RC91] A. Rangarajan and R. Chellappa. A continuation method for image estimation and segmentation. l'ethnical Report CAR-TR-586, Center for Automation Research, University of Maryland, October 1991.

[RGF90] K. Rose, E. Gurewitz, and G. Fox. Statistical mechanics and phase transitions in clustering. *Phys. Rev. Lett, 65(8), 1990.*

[SgS90] G. W. Stewart and Ji guaang Sun. *Matrix Perturbation Theory.* Academic Press, 1990. Theorem on p.25-26, using definitions on p.3.

[Sim90] Petar D. Simic. Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimization. *Network: Computation in Neural Systems*, 1(1):89-*103,* January 1990.

[Tre91] Volker Tresp. A neural network approach for three-dimensional object recognition. In *Neural Information Processing Systems 3.* Morgan Kaufmann, 1991.

[Vap82] V. N. Vapnik. *Estimation of Dependences Based on Emprirical Data.* Springer Verlag, 1982.

[VJ89] B. D. Vujanovic and S. E. Jones. *Variational Methods in Nonconservative Phenomena.* Academic Press, 1989.

[YHP91] A. Yuille, K. Honda, and C. Peterson. Particle tracking by deformable templates. In *International Joint Conference on Neural Networks,* pages I-7 to I-12. IEEE, July 1991.

A Lagrangian Formulation of Neural Networks II: Clocked Objective Functions and Applications

Willard L. Miranker[1] and Eric Mjolsness[2]

[1] Department of Computer Science
and Neuroengineering and Neuroscience Center
Yale University New Haven CT 06520
and
Research Staff Member, Emeritus,
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

[2] Jet Prepulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena CA 91103-3099

Abstract

In Part I of this work we showed how a tradeoff between measures of neural net cost (of operation) and functionality (efficacity) could be used to derive the dynamics of the net, and in particular, optimize thereby a class of objective functions. Here we extend that methodology; a Lagrangian formulation and greedy variation to treat more ramified problems. We introduce a notion of clocking and a class of clocked objective functions to do this. A kind of switching dynamics occurs which is suitable for many applications. This notational clocking calculus makes for time-scaled computational techniques employing a "focus of attention" (similar to saccading, foveation, and covert attention in biological vision). Experiments dealing with applications are referenced.

1 INTRODUCTION

In Part I of this work [MM] (to be referred to hereafter, simply as Part 1) we introduced a Lagrangian formulation of the dynamics of a class of relaxation-based analog neural networks. These Lagrangians incorporate a trade-off between measures of the operational cost and the functionality (efficacy) of neural networks employed to optimize a given objective function E. Because of the need for nonconservative or dissipative dynamics, our Lagrangians are to be varied in a nonstandard way using the so-called "greedy variation". This results in dissipative analog circuit dynamics described by first-order systems of differential equations. Within a class of candidate Lagrangians, we proved the near-optimality (under a suitable meta-objective function) of a particular Lagrangian corresponding to the Hopfield/Grossberg analog circuit dynamics. However, for efficiency, elaborate computations may require more complex dynamics specified at a coarser scale of temporal resolution, and this is a theme of the present work.

Here (in Part II) we proceed to consider more elaborate Lagrangians which are capable of specifying non autonomous dynamics. For example the dynamics may depend on which subset of the problem variables is currently being optimized, as well as the subset next to be optimized. This kind of "[switching" dynamics occurs in many applications and requires a more general formulation of the Lagrangian which we develop [n section 2 we introduce a time-varying or switched version of the problem objective function E, *called* a "clocked objective function". We relate it to

our Lagrangian formulation of dynamics, producing so-called cyclic Lagrangians We develop suitable notation for expressing a number of existing optimization methods in terms of such clocked objectives. Reference is made to a number of experiments, application and computation, which utilize this clocking calculus. In section 3 we show how to specialize these ideas to the case of a computational "focus of attention" (similar to saccading, foveation, and covert attention in biological vision) which iteratively and opportunistically selects a subset of the problem's variables for optimization, and optimizes them. We show how to develop Lagrangians on different problem scales, Greedy variation then leads to the dynamics relevant to each scale. The working of the clocking or switching in the problem development and its solution is worked out, In section 4 we derive and relate various particular focus of attention mechanisms, inducting several which have been tested in previously reported computer experiments. These include priority queue attention, multiscale attention, jumping and rolling windows of attention, spreading activation (of neurons) and orthogonal windows. Section 5 provides a summary.

2 DYNAMICS WITH SWITCHING: VIRTUAL NETWORKS

Suppose we have hardware capable of switching different sets of neuron output values from a static (backup) memory into an active neural network, where they can be updated. With such hardware it is possible to implement a computation which would require a much larger neural network if every neuron were to be actively updated at all times. This situation would be analogous to the use of virtual memory in a conventional computer, in which one has a limited amount of physical memory (Random Access Memory) augmented by a much larger amount of secondary storage (magnetic disks). Equally, it is analogous to the distinction between the small cache memory associated with a central processing unit, and the larger physical memory (RAM). In either part of the memory hierarchy a relatively small and fast memory, in concert with a relatively large and slow memory, simulates a large fast memory (with occasional slowdowns due to page faults or cache misses). In like manner, we seek to design a switching mechanism for obtaining the computational power of a large neural network with a small neural network plus a large, slow and relatively inexpensive memory. Furthermore, in some cases it will prove possible to dispense with the slow memory entirely.

Such a system would be useful not only for making space-time tradeoffs in situations where only a limited amount of spatial resources (neurons and connections) are available, but also for formulating search algorithms (such as binary search) which can't be fully parallelized due to their unpredictable total resource requirements.

What kind of cost and functionality terms would model this situation? This is a hierarchical design problem. At a coarse time scale, we have just two kinds of decisions to make: what the active set of neurons (the *focus of* attention) is to be at any given time, and what their new values are to be after some period of active dynamics. (In the memory hierarchy analogy, one would like to decide which part of slow memory to bring into fast memory as some computation progresses.) At a fine time scale we must repeatedly make circuit-implementable state changes which move towards answering these two coarse-scale questions.

A strong constraint on the system is that, under reasonable cost metrics such as network space-time volume, no savings will be realized unless the focus-of-attention decision has converged to a definite answer by the time a switch of attention is to be made (i.e. by the time that decision is to be implemented); partial answers as to which neurons should be active would just force all the candidate neurons to be active. (An attentive neural network which unhappily violates this constraint is described in [Mjo87].) Of course one can contemplate dynamics in which by way of example a linear combination of neuron values is made active, but such a system should be designed by introducing new variables for the linear combinations and a discrete switching circuit which still, to be physically cost-effective, makes definite decisions about the active set of neurons.

So, our problem is to find both coarse-scale and fine-scale cost and functionality terms to model a focus-of-attention mechanism which switches many stored neuron values into and out of a small active network, where the neural values are updated. We will not consider all aspects of this problem. Rather we shall show how the Lagrangian formalisms provide a tractable framework for

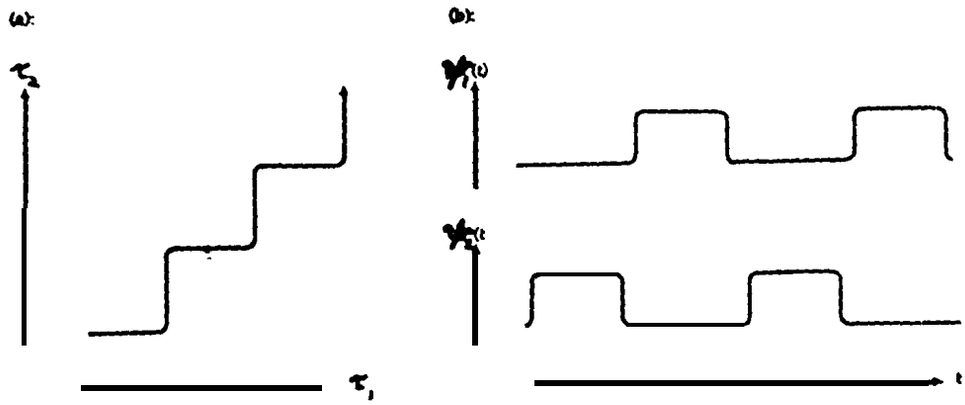

Figure 1: Two time variables τ_1 and τ_2 may increase during nonoverlapping intervals of an underlying physical time variable, t. For example $\tau_1 = \int dt\psi_1(t)$ and $\tau_2 = \int dt\psi_2(t)$ where $\psi_1 = d\tau_1/dt$ and $\psi_2 = d\tau_2/dt$ are nonoverlapping clock signals. (a) The parametric curve $(\tau_1(t), \tau_2(t))$. (b) The functions $\psi_1(t)$ and $\psi_2(t)$.

our approach. This is illustrated through derivation of a few plausible Lagrangians in the form of clocked objective functions. Related work appears in [Coo89, Mjo87, MM91, BSB+ 91].

2.1 Cyclic Lagrangians

In discussing coarse-scale cost and functionality terms, the idea of a repeating cycle of a fixed set of dissimilar coarse-scale decisions will be fundamental. This idea is analogous to a "loop" in programming, or to the use of cyclic clock signals to control an electronic circuit. The idea may be expressed in terms of Lagrangians in several different contexts which we will discuss here. In all cases we will find a simple formulation in terms of a "clocked objective function" [MGM91]: a version of the AE functionality term of the Lagrangian in which the structure of E is regarded as time-dependent according to a temporal cycle corresponding to the fixed cycle of coarse-scale decisions. The possibility of formulating a cyclic Lagrangian in terms of a clocked objective function was introduced in section 2.1.1 of Part 1, equations (17) and (18).

As an example, consider a line-minimization algorithm for local optimization. Repeatedly, one calculates the gradient at a current location x, does a one-dimensional minimization of the objective function along the gradient direction, and updates x. During the cycle it is necessary to store an old configuration $\mathbf{x}^{\text{old}}$ for use in updating x, and to reset to zero the scalar parameter s which measures displacement in the gradient direction.

To express these ideas we recall the clocked objective function notation [MGM91]: Suppose that we have a small set of objective functions $\{E_\alpha\}$ which are to be partially relaxed (i.e. partially optimized) in a cycle. We define one nonoverlapping clock function, $\psi_\alpha(t)$ = O or 1 (with $\sum_\alpha \psi_\alpha(t) \leq 1$), for each phase $\alpha = 1, 2, \ldots, A$ of the cycle. The clocked objective function is written as

$$E_{\text{clocked}}[\mathbf{x}, t] = \sum_\alpha \psi_\alpha(t) E_\alpha[\mathcal{X}_\alpha^{\text{free}} | \mathcal{X}_\alpha^{\text{fixed}}], \tag{1}$$

where $\mathcal{X}_\alpha^{\text{free}}$ and $\mathcal{X}_\alpha^{\text{fixed}}$ arc subsets of variables from the entire set $\{x_i\}$. During phase α (i.e. when $\psi_\alpha(t) = 1$), $E_{\text{clocked}} = E_\alpha[\mathcal{X}_\alpha^{\text{free}} | \mathcal{X}_\alpha^{\text{fixed}}]$ is to be extremized with respect to all variables in $\mathcal{X}_\alpha^{\text{free}}$, while all variables in $\mathcal{X}_\alpha^{\text{fixed}}$ are to be held fixed or clamped. Figure 1 shows one interpretation of the nonoverlapping clock functions ψ_α (t).

For example, a simple clocked objective function for line minimization would be

$$\begin{aligned} E_{\text{clocked}} = \; & \psi_1(t)\frac{1}{2}\left(||\mathbf{x}^{\text{old}} - \mathbf{x}||^2 + s^2\right)[\mathbf{x}^{\text{old}}, s|\mathbf{x}] && \text{(initialize } \mathbf{x}^{\text{old}} \text{ and s)} \\ & + \psi_2(t)\left(E[\mathbf{x} + s\nabla E[\mathbf{x}^{\text{old}}]]\right)[s|\mathbf{x}, \mathbf{x}^{\text{old}}] && \text{(line minimization)} \\ & + \psi_3(t)\frac{1}{2}\left(||\mathbf{x} - \mathbf{x}^{\text{old}} - s\nabla E[\mathbf{x}^{\text{old}}]||^2\right)[\mathbf{x}|s, \mathbf{x}^{\text{old}}] && \text{(update } \mathbf{x}\text{)}. \end{aligned} \tag{2}$$

Since the $\alpha = 1$ and $\alpha = 3$ phases are especially easy quadratic optimizations, one could arrange that these terms are relaxed almost to zero during the clock phase interval appropriate to each. Then equation (2) is a continuous-time refinement of the coarsf-scale Lagrangian's decision cycle, which partially relaxes E in a gradient direction and then resets the variables for the next partial relaxation. At the end of phase 2 in each cycle, the clocked objective function takes the value of E at the new point. So the clocked objective function may be interpreted as a refinement of the functionality term of the coarse-scale decision-cycle Lagrangian. This interpretation also requires that the appropriate variables be held fixed at the correct times; this may be achieved with a cost term C_α which strongly penalizes any change in the clamped variables for the relevant clock phase.

Many variations on equation (2) are possible; the clocked objective could interpolate an extra cycle for the calculation of the gradient vector, and the x used to calculate the gradient could be taken as the $\mathbf{u} = g^{-1}(\mathbf{v})$ rather than v variables for E, and soon.

2.1.1 Relation of E_{clocked} to E

So far we have only argued that clocked objective functions provide an interesting special case of the distributed Lagrangian equation (5) in Part I; we have not shown how they can be related to the static objective function E or the dynamic objective function equation (4) in Part I with functionality term $F = E_{\text{final}} - E_{\text{initial}}$. Here we will discuss three different classes of clocked objective functions, each of which can be used to make some progress on minimizing E in every complete clock cycle so that AE $\leq$ Ofor each cycle even though the functionality term is not simply equal to ΔE. In this section we refer to such a clocked objective as "valid" for objective E.

Transient Terms For the first claw of clocked objective functions, of which the line minimization objective (2) is an example, E_{clocked} is valid if one of its components E_β is equal to E itself, perhaps with restricted arguments, and if the other components can each be expected to relax to near-zero values within their own phase of the cycle. These other components will be referred to as *transient terms* of a clocked objective function, since they approach zero quickly. Then progress is definitely made during phase β, and at least no harm is done (i.e. no increase in E_β is suffered) in the other phases a. Generally these other phases are used to ensure the suitability of the arguments of $E_\beta = E$.

Subspace Terms In the second class **of** valid clocked objective **functions,** E_α **is** equal to E during all clock phases, except that it is a function of different sets of variables (or more generally, is a function on different submanifolds) during different clock phases. We will refer to this type of term a-s a *subspace term* of a clocked objective function. There can be no significant calculation required to decide what subset of variables E depends on during each phase (otherwise we'd need a further phase to make that calculation). One simple arrangement is to partition all variables into a few blocks $\mathcal{X}_\alpha$, with the variables in one block allowed to change during each phase of the clock. Then equation (1) simplifies (since every E_α is just E) to

$$E_{\text{clocked}}[\mathbf{x}, t] = \sum_{\text{a}} \psi_\alpha(t) E[\mathcal{X}_\alpha^{\text{free}} | \mathcal{X}_\alpha^{\text{fixed}}]. \tag{3}$$

This permits concise expression of blockwise coordinate descent algorithms.

It 1S perhaps surprising that $E_{\text{clocked}}[\mathbf{x}, t]$ is not numerically equal to $E[\mathbf{x}(t)]$ for all t in this case, owing to the nonoverlapped clock factors $\psi_\alpha(t) \bullet [0, 1]$ whose sum varies between O (between phases) and 1 (during a clock phase). As we will see in the next section (2.1.2), this is necessary so that the continuous-time Lagrangian will force all variables to completely stop changing between clock phases, as they should.

We note that the second class of clocked objective functions can be used for the discrete parallelization scheme mentioned at the beginning of section 2.3.1 of Part I. l'here we postulated a partition of the network variables into a small number of "colored" blocks, with neighboring variables in the network having different colors. (Colors are in a correspondence with phases.) Such a partition can be used to ensure noninterference of discrete-time parallel update dynamics. Clearly equation (3) is the correct clocked objective for this situation, and F would just be $\Delta E_{\text{clocked}}$.

Control Terms For the third class of valid clocked objective functions which perform optimization, onc constituent objective E_β is again taken to be E with restricted arguments (a subspace term, as in the first and second classes), ancl the other phases either relax to nearly zero (being

composed of transient terms as in the first class) or serve to determine the choice of active arguments for phase β without directly changing any of the original variables x. Since this last type of objective is a sum of terms that only involve variables that control membership in the active set of arguments for β, its constituent terms will be referred to as *control terms* in a clocked objective function. Clocked objective functions with control terms are the class of objective functions most relevant to the attention mechanisms of section 4. In that section we will have occasion to use clocked objectives containing a variety of subspace terms, transient terms and control terms.

2.1.2 Lagrangians for Clocked Objective Functions

We have seen in equation (17) of Part I how clocked objective functions may arise from coarse-scale Lagrangians, in which the the functionality term takes on a cyclic sequence of different forms. Our purpose now is to relate such clocked objective functions (as in (l)) to continuous-time Lagrangians.

The essential feature of a single term $E_\alpha[\mathcal{X}_\alpha^{\text{free}}|\mathcal{X}_\alpha^{\text{fixed}}]$ in a clocked objective function E is that it depends only on some of the variables, the rest being held constant at their earlier values. This gives a property expressible in terms of derivatives:

$$\frac{\partial E_\alpha}{\partial x_i}[\mathcal{X}_\alpha^{\text{free}}|\mathcal{X}_\alpha^{\text{fixed}}] = \chi_{\alpha i}\frac{\partial E_\alpha}{\partial x_i}, \tag{4}$$

where $\chi_{\alpha i} \in \{O, 1\}$ is a *constant* which indicates the presence ($\chi = 1$) or absence ($\chi = O$) of x_i in $\mathcal{X}_\alpha^{\text{free}}$. (For fixed $\alpha, \chi_{\alpha i}$ is an indicatrix for $\mathcal{X}_\alpha^{\text{free}}$). Consequently, $E_\alpha[\mathcal{X}_\alpha^{\text{free}}|\mathcal{X}_\alpha^{\text{fixed}}]$ is a low-dimensional *slice* (restriction) of the higher-dimensional function $E_\alpha[\mathcal{X}_\alpha]$, evaluated at values of the fixed parameters which are dictated by the state vector x at the beginning of the α-th phase.

¿From equations (3) and (4) we may now calculate $\partial E_{\text{clocked}}/\partial x_i$:

$$\frac{\partial E_{\text{clocked}}}{\partial x_i} = \sum_\alpha \psi_\alpha(t)\chi_{\alpha i}\frac{\partial E_\alpha}{\partial x_i}, \tag{5}$$

which is nonzero at any given time t only if x_i is in the free set of variables at that time.

We can take the final continuous-time Lagrangian to be

$$L = \sum_i \left(K[\dot{x}_i, x_i] + \frac{\partial E_{\text{clocked}}}{\partial x_i}\dot{x}_i \right), \tag{6}$$

where K is a cost-of-motion term (see section 1.1 of Part I). To see that this is consistent with the desired pattern of fixed variables as a function of time, we examine the resultant dynamics. As in equation (30) of Part 1, varying $\dot{x}_i$ and using $\sum_\alpha \psi_\alpha \chi_{\alpha i} \in \{O, 1\}$, and defining $\tilde{K}[w, x]$ as the inverse of $K[\dot{x}, x]_{,\dot{x}}$ with respect to its first argument, the equations of motion are

$$\dot{x}_i = \tilde{K}[-\sum_\alpha \psi_\alpha(t)\chi_{\alpha i}\frac{\partial E_\alpha}{\partial x_i}, x_i] = \sum_\alpha \psi_\alpha(t)\chi_{\alpha i}\tilde{K}[-\frac{\partial E_\alpha}{\partial x_i}, x_i]. \tag{7}$$

Here we have used equation (5) and $\tilde{K}[0, x] = O$ to simplify the equations of motion. The factor of $\psi_\alpha(t)\chi_{\alpha i}$ ensures that the correct variables are clamped at the correct times.

Equation (6) is appealing because it has the same form as the continuous-time Lagrangian for unclocked objective functions, equation (22) of Part 1. This is the desired relationship between continuous-time Lagrangians and clocked objectives. Because of equation (6) it will often suffice to give the clocked objective alone, omitting the Lagrangian, in order to specify a network's dynamics.

2.1.3 Notation for Clocked Objective Functions

Equations such as (4) can be expressed in a more convenient notation for algebraic calculations (by human or computer). From an algebraic point of view, (4) may be regarded as the x_i derivative of $\bar{E}_\alpha$, a version of E_α in which all fixed variables $x_j \in \mathcal{X}_\alpha^{\text{fixed}}$ are simply replaced by *clamped variables* (or "fixed variables") $\bar{x}_j$ for which

$$\frac{\partial \bar{x}_j}{\partial x_i} = O \text{ despite the fart that } \frac{\partial x_j}{\partial x_i} = \delta_{ij}. \tag{8}$$

The actual value of x_j is updated to the current value of $\bar{x}_j$ only at the (otherwise irrelevant) time intervals between the nonoverlapped clock phases, when $\sum_\alpha \psi_\alpha(t) = 0$. Equation (4) follows directly from this interpretation of $E_\alpha[\mathcal{X}_\alpha^{\text{free}}|\mathcal{X}_\alpha^{\text{fixed}}]$ in terms of E_α.

In fact, we can design notation for the substitution that relates $\bar{E}$ to E. Define

$$x\{\chi\} = \chi x + (1-\chi)\bar{x} \quad \text{so} \quad \mathbf{x}\{\boldsymbol{\chi}_\alpha\} = \boldsymbol{\chi}_\alpha \cdot \mathbf{x} + (\mathbf{1} - \boldsymbol{\chi}_\alpha) \cdot \bar{\mathbf{x}} \tag{9}$$

where χ is a binary (zero- or one-valued) scalar (or can easily be rounded to zero or one) and $\boldsymbol{\chi}_\alpha$ is just the constant array $\chi_{\alpha i}$ which specifics with its zero-valued entries which variables are clamped in each phase α. With this notation, $\bar{E}_\alpha$ is just $E_\alpha[\mathbf{x}\{\boldsymbol{\chi}_\alpha\}]$, i.e.

$$\bar{E}_\alpha[\mathcal{X}_\alpha^{\text{free}}|\mathcal{X}_\alpha^{\text{fixed}}] = E_\alpha[\mathbf{x}\{\boldsymbol{\chi}_\alpha\}]. \tag{lo}$$

We will use $E_\alpha[\mathbf{x}\{\boldsymbol{\chi}_\alpha\}]$ as the preferred notation. Furthermore $\boldsymbol{\chi}$ need not be a constant; it can be replaced with any vector-valued expression $\boldsymbol{\pi}(\boldsymbol{\xi})$ involving variables $\boldsymbol{\xi}$. Equation (9) would still define

$$\mathbf{x}\left\{\boldsymbol{\pi}(\boldsymbol{\xi})\right\} = \Theta(\boldsymbol{\pi}(\bar{\boldsymbol{\xi}}) - \mathbf{1/2}) \cdot \mathbf{x} + \Theta(\mathbf{1/2} - \boldsymbol{\pi}(\bar{\boldsymbol{\xi}}))\ \mathbf{x}, \tag{11}$$

where

$$\mathbf{c)(r)} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Θ is defined componentwise on vectors. 'l'he purpose of the Θ function in (11) is to round $\boldsymbol{\pi}(\boldsymbol{\xi})$ to zero or one, with a boundary at 1/2. Note that, in agreement with equation (9) in which χ is a constant, $\boldsymbol{\chi}$ is clamped in equation (1 1). That is because x's focus of attention cannot shift during the phase in which x is being relaxed without incurring excessive and uncontrolled switching costs.

As a further notational refinement, we may drop the explicit $\psi(t)$ functions from our notation by defining a *clocked* SUITI,

$$\bigoplus_\alpha E_\alpha \equiv \sum_\alpha \psi_\alpha(t) E_\alpha \tag{13}$$

which may be written out term-by-term as

$$E_1 \oplus E_2 \oplus \ldots E_A. \tag{14}$$

(The "$\oplus$" symbol is evocative both of a rolling "+" sign, and of an analog clock face.) Of course the periodic functions $\psi_\alpha(t)$ still have to be specified before the clocked sum is a well-defined quantity. The clocked sum is neither commutative nor associative, but we may take it to associate over the ordinary sum:

$$\sum_a \bigoplus_\alpha E_{a\alpha} \equiv \bigoplus_\alpha \sum_a E_{a\alpha} \tag{15}$$

Moreover, parenthesized expressions such as $E_1 \oplus (E_2 \oplus E_3)$ may be used to denote nested loops in which for example E_2 and E_3 are repeatedly relaxed in an inner loop, within one phase of an outer loop, and E_1 is relaxed once during the other phase of the outer loop. Again the timing would be controlled by external functions ψ_α (t), which must still be specified separately.

Note that the use of clocked objective functions is reminiscent of time ordering of operators in quantum physics. See also the so-called Feynman entangling calculus [MW66].

Perhaps the most important algebraic property of the clocked sum, for the purpose of formulating descent algorithms, is its commutation with partial differentiation:

$$\frac{\partial}{\partial x_i} \bigoplus_\alpha E_\alpha = \bigoplus_\alpha \frac{\partial}{\partial x_i} E_\alpha. \tag{16}$$

This follows directly from the definition of the clocked sum. The right hand side of equation (16) could be used as the time-dependent descent direction in a gradient-descent algorithm.

We may conventionally expect to find the $\oplus$ signs outside the + signs in a clocked objective function, and accordingly we assign $\oplus$ a lower grammatical precedence than + in otherwise ambiguous expressions. So by convention, $E_1 \oplus E_2 + E_3$ means $E_1 \oplus (E_2 + E_3)$.

With the addition of clamped variables x, conditional variables $x\{\chi\}$, and clocked sums $\bigoplus_\alpha E_\alpha$, we are able to concisely express a wide variety of cloc'keel objective functions. For example the line minimization objective (2) becomes

$$\begin{aligned} E_{\text{clocked}} \;=\; & s^2/2 + ||\mathbf{x}^{\text{old}} - \bar{\mathbf{x}}||^2/2 && \text{(initialize s, } \mathbf{x}^{\text{old}}\text{)} \\ & \oplus E[\bar{\mathbf{x}} + s\nabla E[\bar{\mathbf{x}}]] && \text{(line minimization)} \\ & \oplus ||\mathbf{x} - \bar{\mathbf{x}}^{\text{old}} - \bar{s}\nabla E[\bar{\mathbf{x}}]||^2/2 && \text{(update x),} \end{aligned} \tag{17}$$

or what may be easier to implement as a circuit,

$$\begin{aligned} E_{\text{clocked}} = & \; s^2/2 + ||\mathbf{x}^{\text{old}} - \bar{\mathbf{x}}||^2/2 + ||\mathbf{w} - \nabla E[\bar{\mathbf{x}}]||^2/2 \\ & \quad \text{(initialize s, } \mathbf{x}^{\text{old}}\text{; find gradient w)} \\ & \oplus E[\bar{\mathbf{x}} + s\bar{\mathbf{w}}] && \text{(line minimization)} \\ & \oplus ||\mathbf{x} - \bar{\mathbf{x}}^{\text{old}} - \bar{s}\bar{\mathbf{w}}||^2/2 && \text{(update x).} \end{aligned} \tag{18}$$

Furthermore, clocked objective functions make new algebraic transformations possible. For example, equation (11) may be implemented for X-expressions (assuming only that we can implement it for O/l-valued variables) by introducing new variables η as follows:

$$E[\mathbf{x}\left\{\pi(\xi)\right\}] \rightarrow \sum_i \left[\eta_i(\pi_i(\bar{\xi}) - 1/2) + \phi_{0/1}(\eta_i)\right] \oplus E[\mathbf{x}\{\eta\}]. \tag{19}$$

Here $\phi_{0/1}$ is a two-sided barrier function which limits its argument to values between zero and one.

2.1.4 Experiments

The clocked objective function notation has been used to derive and express a number of experimentally validated relaxation-based neural networks, including networks for multiscale image segmentation [Tsi97], visual pose estimation [LM94], point matching [GLR+95], and invariant learning of point-set and graph models of visual objects [RGM96]. In these applications, the problem variables were divided into an exhaustive collection of subsets each of which received an exclusive clock phase. During the clock phase for any subset of the variables, all other variables were clamped and the optimization of the free subset was relatively easy or even analytically solvable. This situation is described by equation (3), which may be rewritten as a clocked objective function using *(13)*. It occurs sufficiently often that we provide another notation for it:

$$E\langle \mathcal{X}_1^{\text{free}}, \mathcal{X}_2^{\text{free}}, \ldots \mathcal{X}_A^{\text{free}} \rangle_\oplus \equiv \bigoplus_\alpha E[\mathcal{X}_\alpha^{\text{free}} | \mathcal{X}_\alpha^{\text{fixed}}] = \sum_\alpha \psi_\alpha(t) E[\mathcal{X}_\alpha^{\text{free}} | \mathcal{X}_\alpha^{\text{fixed}}] \tag{20}$$

2.1.5 Clocked Circuits

Clocked objective functions can also be used to specify circuits at the analog level. The simplest way to do this is to assign to each clock phase the dynamics of an analog neural network in which some variables have been clamped. The clamping is under the control of the clock signals and/or other variables. That is the effect of equation (6), either under the original definition of clocked objective (5) or under the more powerful and convenient notation defined in equations (8), (11), and (1 3); it is also a basic idea behind the design of clocked pipelines of combinatorial logic in the data paths of simple CPU chips [MC80] where clamping is determined only by the clock signals. We take it as clear, then, that such clocked objective functions can be implemented as analog circuits provided that each phase can be so implemented, and provided that the objective includes $\bar{x}$ expressions (cf. (3)) but does not include $x\{y\}$ expressions (cf. (8)). For example, the line minimization clocked objective of equation (18) can be implemented this way, as can the multiscale optimization objective found in [MGM91].

In the next subsection we show another such example: a clocked objective function which incorporates one or more general feed-forward neural networks inside a relaxation-based neural net, in a hybrid that may be of use for combining relatively efficient learning algorithms (from feed-forward nets) with expressive power (from relaxation nets).

Later, we will discuss a set of applications that require the more powerful $x\{y\}$ notation, without speculating on the hidden circuit- level implementation of the switching mechanism. Thus the

problem of eliminating $x\{y\}$ expressions in favor of x expressions remains for future work; it is related to the "neural network routing problem" discussed in [MG90], section 2.6. A further open problem is to replace global clock signals in a Lagrangian circuit formulation with a system of self-timed subcircuits in which the ψ_α control functions are replaced by relatively local variables with independent dynamics. Solutions to analogous problems are implicit in the design of many distributed computer systems but not within a circuit-level Lagrangian framework. The $x\{y\}$ notation represents a substantial escalation in expressive power, and section 4 is devoted to some of its uses in designing computational attention mechanisms.

2.1.6 Feed-Forward Networks as Constraint Projection

A feed-forward network inside of a relaxation network can be regarded as a set of *constraints* on the relaxation network:

$$E_{\mathrm{FF/relax}}[\mathbf{x}] = E_{\mathrm{relax}}[\mathbf{x}] + \sum_{l\,(\mathrm{layers})} \mathrm{FF}[\mathbf{v}^l, T^l, \mathbf{v}^{l-1}], \tag{21}$$

where FF is the functional dependency constraint of a layer's output neurons on its input neurons (here taken to be in the previous layer, though neurons in any previous layer may be inputs without causing problems for the following algorithm). Various methods are available for enforcing constraints within a neural network optimization [PB87, MG90, PS89], but the feed-forward network constraints have a natural ordering determined by the feed-forward pattern of connections. So in this special-case we can use a nonlinear *projection* method to enforce all the constraints. As mentioned in section 2.3.1 of Part I, related algorithms are discussed in [BT89], for example, under the name of "gradient projection algorithms" or "scaled gradient projection algorithms".

Any incremental relaxation of the objective E_{relax} is followed by a series of projections which reestablish the feed-forward constraints, layer by layer (i.e. from earlier to later neurons in the feed-forward neuron order), in preparation for further relaxation. The clocked objective is

$$E_{\mathrm{FF-projection}}[\mathbf{x}] = \bigoplus_{l\,(\mathrm{layers})} \left\{ \sum_i \left\{ -v_i^l \sum_j T_{ij}^l \bar{v}_j^{l-1} + \phi_i(v_i^l) \right\} \right\} \oplus E_{\mathrm{relax}}[\mathbf{x}]. \tag{22}$$

Note the especially simple form of each layer's objective:

$$\sum_i \left\{ -v_i^l \sum_j T_{ij}^l \bar{v}_j^{l-1} + \phi_i(v_i^l) \right\}. \tag{23}$$

Every neuron v_i^l in layer 1 is independent of every other in this objective, and the minimization of this objective is best achieved just by assigning values to all layer-l variables in parallel:

$$v_i^l = g_i\left(\sum_j T_{ij}^l v_j^{l-1}\right), \text{ where } g_i^{-1}(v) = \phi_i'(v). \tag{24}$$

This is the projection operation which immediately enforces the layer-l constraints. Later layers' projection operations do not disrupt earlier ones. So, at the beginning of the relaxation phase of every cycle, all the FF constraints will have been consistently satisfied.

3 FOCUS OF ATTENTION THEORY

A particular kind of clocked objective function formalizes the idea of a computational focus of attention. We will derive this clocked objective by first considering the functionality and cost terms of a coarse-scale greedy Lagrangian, and then developing the associated fine-scale greedy Lagrangian which specifies circuit-level dynamics,

3.1 Formulation of the Lagrangian at the Coarse Scale

Let $\mathcal{X}$ be a set of discrete-valued variables which determine, directly or indirectly, which components of the neuron vector v are actively updated at any given time. In other words, $\mathcal{X}$ determines a

characteristic function $\pi_i(\chi)$ for the *focus of attention* or active set of v_i's. Thus

$$\pi_i(\chi) = \begin{cases} 1 \text{ if } v_i \text{ is active, i.e. in the focus of attention,} \\ 0 \quad \text{otherwise,} \end{cases} \tag{25}$$

with

$$\sum_i \pi_i(\chi) = n. \tag{26}$$

For example, we could have as many components of χ as of v and set $\pi_i(\chi) = \chi_i$. Or instead, we could introduce a partition of the components of v into blocks indexed by cc, with a 0/1 partition matrix B_{ia}; this is a form of aggregation applied to χ. (For now we will take n to be constant, though a variable n is sometimes useful) Then we would have one component of χ to switch each block of the partition, and $\pi_i(\chi) = \sum_a B_{ia}\chi_a$. (That is, a variable v_i isin the focus of attention if and only if its course-scale block a is in the focus of attention as determined by χ_a.) Usually $\pi_i(\chi)$ can be made linear in χ.

Regardless of the actual formula for $\pi_i(\chi)$, there will be some sparseness constraint on χ to ensure that only a small fraction of the neurons v are in the focus of attention at any one time. For example one might impose $\sum_i \pi_i(\chi) = n$, where n is the optimal size of the focus of attention (and n<< N = the total number of neurons vi). In the case of a partition matrix B with blocks of roughly equal size b (so $\sum_i B_{ia} \approx b$), the sparseness constraint would become $\sum_a \chi_a = n/b$.

Whatever the sparseness constraint on χ is, we will express it as a summand $\hat{\Phi}(\chi)$ in an objective function. Φ may be a penalty function, a barrier function, a Lagrange multiplier times the constraint, orsomecornbinatio nof these possibilities. Thus, we could choose froma variety of "k-winner" objective functions(k winners allowed in a competitive group). Assuming $\hat{\Phi}(\chi) = \Phi(e)$ where $e \equiv \sum_i \pi_i(\chi) - n$, we can enforce or at least favor satisfaction of the constraint e $\leq$ O with

$$\Phi(e) = \left| \begin{array}{rl} (c/2)e^2 & \text{(a penalty term), or} \\ \lambda e + \text{cue} - (c/2)\sigma^2 & \text{(Lagrange multiplier+ effective penalty [MG90],} \\ & \text{with } \sigma \text{ an appropriate auxiliary variable), or} \\ c\int_{-\infty}^{e} g(x)dx & g \text{ monotonic and odd (a barrier term), or} \\ e\sigma - \int_{\min g(y)}^{\sigma} g^{(-1)}(x)dx, & \text{(effective barrier, linear in e),} \end{array} \right. \tag{27}$$

Stricter sparseness terms are also permissible, such as a sum of many k-winner terms on different sets of variables. And for a variable-size focus of attention, in which *71* is variable, one would also need a cost term for n.

All components of v will be assumed to take continuous values, even if they are ultimately supposed to converge to discrete values, Then the coarse-time-scale update rule implied by the action S will be of the form

$$\text{v}' = \mathbf{v}'(\mathbf{v}, \chi). \tag{28}$$

For example

$$v_i' - v_i = \pi_i(\chi) G_i(\mathbf{v}), \tag{29}$$

where G is the cumulative effect determined by the fine-scale dynamics within an active-v clock phase. 'l'his update rule is to be derived from the greedy variation of a multiphase dynamical objective of the form

$$S = \sum_{\substack{\text{coarse scale} \\ \text{decision times } t, \\ L_{\text{cycle}} \geq 0}} L(t) = \sum_{t \mid \sum_\alpha L_\alpha \geq 0} \sum_{\alpha \in \left\{ \begin{array}{c} \text{coarse-}\mathbf{v}, \\ \text{coarse-}\chi \end{array} \right\}} \psi_\alpha(t) \left[C_\alpha(t) + F_\alpha(t) \right], \tag{30}$$

where ψ_α is defined as in section 2.1. *The principle feature of equation* (30) *is that it has two c1ocIc phases, one* during which the v variables are free to move and the χ variables are clamped, and one in which the roles are reversed, During the active-χ phase the focus of attention is determined for the next active-v phase of the cycle.

Notice also that we have assumed a simple stopping criterion, $\sum_\alpha L_\alpha < 0$, which means that the coarse-scale dyuarnics continues only as long as its benefits (decrease in F) outweigh the costs (given by C'), and this decision is made at the end of each complete cycle. We must now find suitable functions $C_{\text{coarse-v}}$, $F_{\text{coarse-v}}$, $C_{\text{coarse-}\chi}$, and $F_{\text{coarse-}\chi}$.

3.2 Coarse-Scale F

To find the F terms, we must decompose $F_{\text{total}} = \Delta E$ into a sum of roars(~-scale causal terms. We would like F_{coarse} to measure the improvement in E due to choosing a configuration $\boldsymbol{\chi}$ and then updating v accordingly:

$$F(t) = F_{\text{coarse-v}} + F_{\text{coarse-}\chi} = E[\mathbf{v}'(\mathbf{v}, \boldsymbol{\chi})] - E[\mathbf{v}] + \Phi(\boldsymbol{\chi}) \tag{31}$$

How can wc decompose this combined effect of v and $\boldsymbol{\chi}$ into separate F terms for each coarse-scale decision? As previously mentioned, the difficulty is that the coarse-scale decision step which chooses values for $\boldsymbol{\chi}$ cannot be made simultaneously with the decision of v values whose presence in the focus of attention is determined by that particular $\boldsymbol{\chi}$. One obvious way to accomplish this is to stage alternating coarse-scale decision phases, updating the two sets of variables, each based on the most recent value of the other:

$$\begin{aligned} \chi' &= \boldsymbol{\chi}'(\boldsymbol{\chi}, \mathbf{v}) \\ v' &= \mathbf{v}'(\mathbf{v}, \boldsymbol{\chi}'). \end{aligned} \tag{32}$$

Then, to decompose $F_{\mathbf{v}} + F_{\chi} = E[\mathbf{v}'] - E[\mathbf{v}]$, we may interpose some especially low cost *estimate* $\tilde{\mathbf{v}}$ of v' which could even be computed analytically given any candidate $\boldsymbol{\chi}'$:

$$\begin{aligned} F_{\text{coarse}\,\chi'}[\mathrm{X'}\,|\mathrm{v}] &= E[\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')] - E[\mathbf{v}] + \Phi(\boldsymbol{\chi}) \\ F_{\text{coarse}\,\mathbf{v}'}[\mathbf{v}'|\mathbf{v}, \mathrm{X'}] &= E[\mathbf{v}'|\boldsymbol{\chi}'] - E[\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')|\mathbf{v}, \boldsymbol{\chi}']. \end{aligned} \tag{33}$$

The optima of these two expressions with respect to their free arguments then determine the functions in equation (32). Note that $F_{\text{coarse}\,\mathbf{v}'}[\mathbf{v}' | \ldots]$ is independent oft, though the constant $E[\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')]$ is subtracted off to satisfy equation (31).

The F functions of equation (33) may be understood in the terminology of section 2.1.1 as a control term $(\Delta E)_{\text{est}}[\boldsymbol{\chi}|\mathbf{v}] = E[\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')] - E[\mathbf{v}]$, a transient term $\Phi(\boldsymbol{\chi})$, and a subspace term $E[\mathbf{v}'|\boldsymbol{\chi}']$. However, the subspace term is carefully normalized by subtracting the constant $E[\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')]$ in order to apportion credit for a given ΔE (equation (31)) between the $\boldsymbol{\chi}$ and v phases of the dynamics. By equations (9) and (25), the subspace term $E[\mathbf{v}'|\boldsymbol{\chi}']$ may be written as $E[\mathbf{v}'\{\boldsymbol{\pi}(\boldsymbol{\chi})\}]$. So the objective function of equation (33) is equivalent to the clocked objective function

$$E_{\text{atten}} = (\Delta E)_{\text{est}}[\boldsymbol{\chi}|\mathbf{v}] + \Phi(\boldsymbol{\chi}) \oplus E[\mathbf{v}\{\boldsymbol{\pi}(\boldsymbol{\chi})\}]. \tag{34}$$

It remains to specify the parameterization $\pi(\chi)$ of the focus of attention, the cost $\Phi(\boldsymbol{\chi})$ for a given focus of attention, and the estimation formula for the ΔE that would accrue from a given focus of attention $\pi(\chi)$. Each can be specified in a variety of ways. $\Phi(\boldsymbol{\chi})$ may be a k-winner constraint. Also the estimation formula $(\Delta E)_{\text{est}}$ may be meta-optimized to provide more accurate estimations as judged by their effect on the performance of the attention algorithm.

In summary, once we are given the function $\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')$ and the cost terms C_α, there is a Lagrangian (the sum of cost and functionality terms) and an associated optimization principle ($\delta_G L = 0$, as in section 2.2 of Part I) that determines the discrete-time dynamics of v and $\boldsymbol{\chi}$. The action is given by (30) for S and (33) for F.

3.2.1 **Criteria for Estimating the Effects of a Focus**

It remains to find suitable expressions or dynamics for $\tilde{\mathbf{v}}(\mathbf{v}, \boldsymbol{\chi}')$. These have the function of estimating the influence of alternative $\boldsymbol{\chi}$ vectors (hence of different foci of attention) on v without actually performing the minimization of $F_{\text{coarse}\,\mathbf{v}'}[\mathbf{v}' | \ldots]$. This problem is closely analogous to the recta-optimization problem posed in section 3.2 of Part I . 'l'here we sought a functional form $K(\dot{\mathbf{v}}, \mathrm{v})$ for the kinetic energy which resulted in the "optimal" dynamical system, where optimality was defined to depend on behavior in many different trials of the network. Likewise we must first define meta-optimality and then seek it, in the determination of a formula for $\tilde{\mathbf{v}}$ which will be used in many different trials of the network.

For any such functional $\tilde{\mathbf{v}}$, the required network computation must be very *inexpensive* compared to that of v' for this reason: the cost of optimizing $F_{\text{coarse}\,\chi}$ is expected to be some large number of fine-scale iterations times the cost of finding $\tilde{\mathbf{v}}$ and is to be added to (and therefore balanced with) the cost of finding v',

As always we must weigh functionality against cost. What makes an estimator $\tilde{\mathbf{v}}(\mathbf{v}, X')$ effective? For a single neural network trajectory, the obvious choice is to consider the $\tilde{\mathbf{v}}$ function effective to the extent that the resulting $\mathbf{v}(t)$ trajectory minimizes the action S in (30). After all, the Lagrangian already contains the correct balance of cost and benefit terms for judging the $\mathbf{v}$ dynamics, complete with a stopping criterion. The only remaining question is how to aggregate over many trials of the network which share the same formula for $\tilde{\mathbf{v}}$, i.e. many starting points, inputs, and possibly connection matrices. One could attempt a worst-case analysis as in the determinat ion of $K(\dot{\mathbf{v}}, \mathrm{v})$, but we have not succeeded in that, Alternatively we consider an average case measure of action, averaged just over some probability distribution on starting points.

We have already proposed a recta-objective, (35), for this type of problem, Here we are averaging over starting points (and perhaps also over inputs h and connection matrices T):

$$\mathcal{M}[\tilde{\mathbf{v}}] = \langle S \rangle = \Big\langle \sum_{t|L(t)\geq 0} L(t) \Big\rangle_{\mathbf{v}(0)} \approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t|L(t)\geq 0} L(t|\mathbf{v}_p(0)) = \mathcal{M}_p[\tilde{\mathbf{v}}], \tag{35}$$

where {VP(0)} are P starting points sampled from the same random distribution over initial conditions.

Generally, predictive accuracy in $\tilde{\mathbf{v}}$ is rewarded by this objective because of the term $E[\mathbf{v}'|\chi']$ in (33): x' is optimized for $E[\tilde{\mathbf{v}}(\mathbf{v}, \mathrm{x})]$ and then used as a constraint in optimizing $E[\mathbf{v}'|\chi']$ with respect to v'.

The sampling procedure converts the infinite sum into a computable and optimizable quantity $\mathcal{M}_p$ at the expense of introducing a *learning* and *generalization* problem. As in theoretical approaches to learning [Vap82, BH89], we must ensure a sample size sufficient not only to approximate the infinite sum, but to continue to do so even after the sampled objective has been optimized (by tuning $\tilde{\mathbf{v}}$) to that particular sample (so that it is no longer a random sample of the infinite sum). In this way, a nontrivial predictive learning problem enters into the design of the switched neural network dynamics.

$\mathcal{M}_\infty$ may also be regarded as an average over all configurations along a trajectory, rather than just over the starting points, since every decision point along the trajectory contributes to the summed action. But to do this we must define a suitable probability distribution of configurations, and the distribution itself is a function of $\tilde{\mathbf{v}}$. This may limit its usefulness for simplifying the objective.

The connection between the optimization of $\tilde{\mathbf{v}}$ and a learning problem demonstrates one advantage of the derivation in section 3.2 of Part I of optimal kinetic energy terms from a worst-case meta-objective (equation (60) in Part I) rather than an average-case meta-objective (equation (35)): by this means analysis could be substituted for a large and (in general) recurring training computation.

3.2.2 Candidate $\tilde{\mathbf{v}}$ Estimators

We now present several possible forms for $\tilde{\mathbf{v}}(\mathbf{v}, \chi)$, which are to be optimized and evaluated according to the criteria of the previous section. In the simplest form, $\tilde{\mathbf{v}}$ is to be computed by hypothesizing a small, constant time At between course scale decisions, during which $\tilde{\mathbf{v}}$ and therefore $E[\mathbf{v}]$ change according to Taylor's formula:

$$\tilde{v}_i = v_i + \Delta t \frac{dv_i}{d\tau_v} \tag{36}$$

(cf. (29)) where $\tau_v = \int \psi_{\mathbf{v}}(t)dt$ as in Figure 2.1.

We may also introduce, for each variable v_i, a hypothetical time axis τ_i which increases linearly with real time t when neuron vi is in the focus of attention (equivalently, when $\psi_{\mathbf{v}}(t) = 1$ and x allows vi to be actively updated, i.e. when $\psi_{\mathbf{v}}(t)\pi_i(\chi) = 1$) and stays constant otherwise. So

$$\tau_i(t) = \int dt \psi_{\mathbf{v}}(t)\pi_i(\chi), \text{ and } d\tau_i/d\tau_{\mathbf{v}} = \pi(\chi). \tag{37}$$

Then

$$\tilde{v}_i = v_i + \Delta t \frac{dv_i}{d\tau_i} \frac{d\tau_i}{d\tau_{\mathbf{v}}} \tag{38}$$

and

$$F_{\text{coarse}-\chi'}[\chi|\mathbf{v}] = E[\tilde{\mathbf{v}}(\mathbf{v},\chi')] - E[\mathbf{v}] + \Phi(\chi) \simeq (\Delta E)_{\text{est}}[\chi|\mathbf{v}] + \Phi(\chi), \tag{39}$$

where

$$(\Delta E)_{\text{est}}[\chi|\mathbf{v}] = \Delta t \sum_i \left(\frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i}\frac{d\tau_i}{d\tau_{\mathbf{v}}}\right)[\mathbf{v}(t_{\text{beginning of v phase}})|\chi]. \tag{40}$$

We introduce the useful quantity

$$E_{;i}[\mathbf{v}] \equiv \frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i}, \tag{41}$$

which for Hopfield/Grossberg dynamics becomes (cf. equation (30) of Part I)

$$E_{;i}[\mathbf{v}] = -g_i'(g_i^{-1}(v_i))\left(\frac{\partial E}{\partial v_i}\right)^2 \equiv -g_i'(u_i)(E_{,i})^2, \tag{42}$$

first proposed as an objective function for driving a focus of attention in [Mjo87]. With these definitions, $(\Delta E)_{\text{est}}$ becomes

$$(\Delta E)_{\text{est}}[\chi|\mathbf{v}] = \Delta t \sum \pi_i(\chi) E_{;i}[\mathbf{v}] + \Phi(\chi), \tag{43}$$

and the associated $\tilde{\mathbf{v}}$ becomes, from (38),

$$\tilde{v}_i = v_i + \Delta t \pi_i(\chi)\dot{v}_i, \tag{44}$$

where now $\dot{v}_i \equiv dv_i/d\tau_i$ and $\dot{v}_i$ will take bounded values determined by the v-phase Lagrangian.

'l'he optimizing parameter here (for the prediction objective *M) is* At, which will also enter into the coarse-scale cost term, since the cost of switching can be amortized only over the time At. Note that the variables χ_a are still discrete, and the cost of partly or completely minimizing $F_{\text{coarse}\,\chi'}$ depends on the relation between $\pi_i(\chi)$ and χ_a to be specified.

Naturally the partial relaxation cost associated with $\pi_i(\chi)$ will only increase if we take the natural step of expanding $\tilde{\mathbf{v}}$ and E to second order in At. One good reason for doing this second-order expansion is that the optimal At will not be small if switching costs are sufficiently high, so a second order approximation may be more accurate. The second-order expansion proceeds as before:

$$\tilde{v}_i = v_i + \Delta t \pi_i(\chi)\dot{v}_i + \frac{\Delta t^2}{2}\pi_i(\chi)\ddot{v}_i \tag{45}$$

and

$$(\Delta E)_{\text{est}}[\chi|\mathbf{v}] = \text{At} \sum_i \pi_i(\chi) E_{;i}[\mathbf{v}] + \frac{\Delta t^2}{2}\sum_{ij}\pi_i(\chi)\pi_j(\chi)E_{;ij}[\mathbf{v}] + \Phi(\chi), \tag{46}$$

where $E_{;i}$ has been defined in equation (43) and where $E_{;ij}[\mathbf{v}]$ is the quadratic form given by

$$\begin{aligned} E_{;ij}[\mathbf{v}] &\equiv \frac{\partial^2 E}{\partial\tau_i\tau_j} \\ &= \frac{\partial^2 E}{\partial v_i v_j}\frac{dv_i}{d\tau_i}\frac{dv_j}{d\tau_j} + \delta_{ij}\frac{\partial E}{\partial v_i}\frac{d^2 v_i}{d\tau_i^2} \\ &= E_{,ij}\dot{v}_i\dot{v}_j + \delta_{ij}E_{,i}\ddot{v}_i. \end{aligned} \tag{47}$$

For example under Hopfield/Grossberg dynamics, $E_{;ij}$ can be calculated as

$$\tau_H^2 E_{;ij}[\mathbf{v}] = g'(u_i)g'(u_j)E_{,i}E_{,j}E_{,ij} + \delta_{ij}g'(u_i)E_{,i}\left(\sum_k g'(u_k)E_{,ik} + \frac{g''(u_i)}{g'(u_i)}(E_{,i})^2\right) \tag{48}$$

Because $\pi_i(\chi)^2 = \pi_i(\chi)$, any diagonal terms in the quadratic form $\sum_{ij}E_{;ij}\pi_i(\chi)\pi_j(\chi)$ (cf. (46)), in particular all those terms with δ_{ij} factors as in (48), can be absorbed into the π-linear part of $F_{\text{coarse}\,\chi'}$ For example, in a quadratic neural net objective $E[v] = -(1/2)\sum_{ij}T_{ij}v_iv_j - \sum_i h_iv_i + \sum_i\phi(v_i)$, the coefficient of the quadratic form for χ could be taken as

$$E_{;ij}[\mathbf{v}] = -T_{ij}g'(u_i)g'(u_j)E_{,i}E_{,j}. \tag{49}$$

In this case the π-quadratic part of (46) becomes

$$(\Delta E)_{\text{estimate-quadratic}} = -\sum_{ij} \pi_i(\chi)\pi_j(\chi)g'(u_i)g'(u_j)E_{,i}E_{,j}T_{ij}, \tag{50}$$

and a corresponding connection matrix would have the opposite sign.

The essential ncw feature of objective (46) is that it involves quadratic infraction.s between the χ expressions corresponding to different neurons This introduces a nontrivial *scheduling* problem as part of the determination of the next focus of attention: separate neurons must not only be capable of making progress individually, but also those neurons likely to cooperate should be scheduled into the same focus of attention. This point will be elaborated in section 4.2.

3.2.3 Cost Terms

At the coarse scale, the cost of one cycle of computation is the cost of running the v network for time $\Delta t_{\mathbf{v}}$, plus the cost of switching to the χ network, plus the cost of running the x network for a period At$_x$, plus the cost of switching back to the v network to start the next cycle.

These considerations may be expressed in the following cost terms for a coarse-scale clocked Lagrangian:

$$C_{\text{coarse-v}} = C_{\text{switch}} + N_1(n)\Delta t_{\mathbf{v}} + \text{Clamp}(\Delta\chi, \{\Delta v_{i|\pi_i(\chi)=0}\}) \tag{51}$$

and

$$C_{\text{coarse-}\chi} = C_{\text{switch}} + N_2(n)\Delta t_{\chi} + \text{Clamp}(\Delta\mathbf{v}), \tag{52}$$

where "Clamp" is a penalty or barrier function which enforces the constancy of v or x as needed. Both of the cost terms here are constant if we regard $n, \Delta t_{\mathbf{v}}$, and At$_x$ as constant within a run, although in that case the constant values of the n and the At's probably should be chosen by a meta-optimization procedure using the same action, averaged over many trials, as the meta-objective.

Such a meta-optimization procedure could also be generalized to produce a simple rule, rather than a constant value, for each At and for 71; when such a rule produces the result $\Delta t_{\mathbf{v}}$ = At$_x$ = O, the computation stops, In that way the cornrnon problem of choosing a stopping criterion, as well as the more specialized problem of switching between optimization of v and of χ, fall naturally in the purview of meta-optimization. Of course such a rule could be given in the form of a Lagrangian for Δt_α, or equivalently for ψ_α, but we will not pursue this case here.

3.3 L at the Fine Scale

Since the v are analog variables, finding fine-scale C and F terms which act to minimize the coarse-scale ones is *now easy*. We proceed as in sections 2.1.2 of Part I and 3 of Part I, except that the Lagrangian functional of equation (22) in Part I is generalized to integrate each variable v_i according to its own internal time variable $\tau_i = \int \psi_{\mathbf{v}}(t)\pi_i(\chi)(t)dt$ as in Figure 1:

$$S^{(1)}_{\text{fine-v}} = \sum_i \int d\tau_i \left(K[\frac{dv_i}{d\tau_i}, v_i] + \frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i} \right) \tag{53}$$

We may convert this into an integral of a single Lagrangian over a single time variable by using the formula for τ_i and the fact that $\psi_{\mathbf{v}}(t)$ and $\pi_i(\chi)(t)$ are each approximately zero or one almost all the time:

$$\begin{aligned}
S^{(1)}_{\text{fine-v}} &= \sum_i \int dt \frac{d\tau_i}{dt} \left(K[\frac{dv_i}{d\tau_i}, v_i] + \frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i} \right) \\
&\approx \int dt \sum_i (\frac{d\tau_i}{dt})^2 \left(K[\frac{dv_i}{d\tau_i}, v_i] + \frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i} \right) \\
&= \int dt \sum_i \psi_{\mathbf{v}}(t)\pi_i(\chi) \left(\frac{d\tau_i}{dt} K[\frac{dv_i}{d\tau_i}, v_i] + \frac{\partial E}{\partial v_i}\frac{dv_i}{d\tau_i}\frac{d\tau_i}{dt} \right) \\
&\approx \int dt \psi_{\mathbf{v}}(t) \sum_i \pi_i(\chi) \left(K[\frac{dv_i}{d\tau_i}\frac{d\tau_i}{dt}, v_i] + \frac{\partial E}{\partial v_i}\frac{dv_i}{dt} \right) \\
&\qquad \text{(using } K[0, v] = 0 \text{ and } d\tau_i/dt \approx 0 \text{ or } 1) \\
&= \int dt \psi_{\mathbf{v}}(t) \left(\sum_i \pi_i(\chi) K[\frac{dv_i}{dt}, v_i] + \sum_i \pi_i(\chi)\frac{\partial E}{\partial v_i}\frac{dv_i}{dt} \right).
\end{aligned} \tag{54}$$

But this is not quite the whole fine-scale Lagrangian for the active-v clock phase, because of the coarse cost terms of equation (51). The "Clamp" terms may be refined by adding appropriate cost-of-movement terms $K[\dot{x}, x]$ (where K is minimal at $\dot{x} = 0$) for each of the clamped variables:

$$S^{(2)}_{\text{fine-v}} = \int dt \psi_{\mathbf{v}}(t) \left(\sum_{\text{all non-v variables } x} K[\dot{x}, x] + \sum_i (1 - \pi_i(\chi)) K[\frac{dv_i}{dt}, v_i] \right) \tag{55}$$

Adding $S^{(1)}$ and $S^{(2)}$ together, we get the part of the action that pertains to the active-v phase:

$$S_{\text{fine-v}} = \int dt \psi_{\mathbf{v}}(t) \left(\sum_{\text{all variables } x} K[\dot{x}, x] + \sum_i \pi_i(\chi) \frac{\partial F}{\partial v_i} \frac{dv_i}{dt} \right) \tag{56}$$

Comparing this action to the Lagrangian in equation (6), we see that the fine-scale dynamics is that of a clocked objective function governed by the focus of attention characteristic function $\pi_i(\chi)$.

Note that, as far as the Lagrangian is concerned, this refinement amounts to an algebraic substitution

$$\psi_{\mathbf{v}}(t) \; [C_{\mathbf{v}} + F[\mathbf{v}]] \rightarrow \psi_{\mathbf{v}}(t) \left(\sum_{\text{all variables } x} K[\dot{x}, x] + \sum_i \pi_i(\chi) \frac{\partial F}{\partial v_i} \dot{v}_i \right), \tag{57}$$

which is justified since at the end of a coarse-scale step, F is just a constant starting value plus a coarse-scale change $\Delta_{\text{coarse}} F$, and the coarse-scale change is equal to a sum of fine-scale changes $\int dt \sum_i (\partial F / \partial v_i) \dot{v}_i$. Also, the K terms for the clamped variables (some v_i and all other variables) serve as penalty terms which, in the absence of other $\dot{x}$ terms, enforce $\dot{x}$ = O when $\psi_{\mathbf{v}}$ = 1 and thereby refine the "Clamp" terms of $C_{\mathbf{v}}$.

The hard part of refining a focus-of-attention Lagrangian is to find fine-scale C and F terms for the variable-z phase, because our coarse-scale terms assume discrete-valued χ variables and the previous refinement techniques don't apply to that case. Indeed, a general, N variable, discrete-valued optimization may be the goal of the entire neural computation (at the coarsest time scale of all) so we surely can't assume that much capability at the tine time scale. On the other hand we have already accepted an approximation in $F_{\text{coarse-}\chi}$ on the grounds that it is not global convergence but merely the order of neural updates that is at stake. Additional simplifying approximations may also be acceptable if optimized through training and verified through testing.

Unless $F_{\text{coarse-}\chi}$ is linear in χ_a, (for example by being linear in At with $\pi_i(\chi)$ linear in x), this F is a nonlinear objective which will require many steps of analog relaxation dynamics, implying an uncertain time to convergence to a nearly discrete- valued χ. Since we only have an intermediate, fixed time At available for relaxation, some additional mechanism will be required to find discrete values for χ after a possibly incomplete analog optimization of $F[\xi]$, where ξ_a are continuous-valued versions of χ_a.

3.3.1 Two Phases of Switching

The computational **savings we** seek accrues through the actual switching from one active set of neurons to the next. For switching to occur, however, we need a "digital restoration phase" in which the χ variables are restored to definite 0/1 values. This phase could be left implicit in our modeling, as part of the unspecified switching hardware, but then we would be unable to analyze possible failures of the mechanism such as too little time to converge to discrete values, or too many $\pi_i(\chi) = 1$. By contrast it is easy to leave purely digital circuit switching details unspecified, since accumulated experience makes it relatively easy to engineer such circuit mechanisms outside of our methodology. We will however explicitly model a third phase, in which analog variables $\hat{\chi}_a$ are restored to nearly discrete values χ_a, as close to O or 1 as any physical circuit quantity ever gets.

Then we will have a global cycle through one phase that relaxes the analog v variables and two phases that optimize the discrete $0/1\chi$ variables by first optimizing analog variables ξ and then restoring them to nearly discrete values $\hat{\chi}$ which can substitute for actual discrete values χ in any circuit implementation. Of course in a digital implementation medium (such as a general-purpose software environment) which exists as an abstraction of some analog physical system, one should instead move directly from ξ to χ.

With this addition the fine-scale Lagrangian becomes

$$L_{\text{fine}} = \sum_{\text{all variables } x} K[\dot{x}, x] + \sum_{\text{phases } \alpha} \psi_\alpha(t) \sum_{\alpha\text{-variables, } \mathbf{x}_\alpha} \left(\frac{\partial E_\alpha}{\partial \mathbf{x}_\alpha} \dot{\mathbf{x}}_\alpha \right) \tag{63}$$

which, as we showed with equation (6), is exactly the Lagrangian corresponding to a clocked objective function

$$E_{\text{fine}} = \sum_{\alpha} \psi_\alpha(t) E_\alpha[\mathbf{x}_\alpha | \ .] = \bigoplus_{\alpha} E_\alpha[\mathbf{x}_\alpha, \bar{\mathbf{x}}_{\beta \neq \alpha}]. \tag{64}$$

More particularly (substituting from equations (57) and (60)) we get the clocked objective function for three-phase attentive dynamics:

$$\begin{aligned} E_{\text{3-phase}} = & \sum_i \pi_i[\xi] E_{;i}[\bar{\mathbf{v}}] + \Phi\left(\sum_i \pi_i(\xi) - n\right) + \sum \phi_{0/1}(\xi_a) && \text{(control terms)} \\ & \oplus - \sum_a \chi_a(\bar{\xi}_a - \theta) + \sum_a \phi_{0/1}(\chi_a) && \text{(transient terms)} \\ & \oplus \ E[\mathbf{v}\{\pi(\chi)\}]. && \text{(subspace term)} \end{aligned} \tag{65}$$

This clocked objective function for a focus of attention is a more elaborated version of equation (34). Note that, from equation (57), we have

$$\frac{\partial E[\mathbf{v}\{\pi(\chi)\}]}{\partial v_i} = \pi_i(\chi) \frac{\partial F}{\partial v_i} = \pi_i(\chi) \frac{\partial E}{\partial v_i}, \tag{66}$$

which is the essential feature of a clocked objective function, as derived in (5).

Various special case expressions for $\pi_i(\chi)$ will be explored in the next section. In the resulting networks we will often omit the digital resetting phase for a simple kWTA network, on the understanding that it should be restored as part of an analog circuit design.

4 APPLICATIONS TO COMPUTATIONAL ATTENTION

Here we present several possible applications of the forgoing computational attention mechanisms and notation. The first two (sections 4.1 and 4.2) have been employed to good effect in [Tsi97] where substantial savings in computational cost are documented. The rest of the applications below may be considered as design examples.

4.1 Priority Queue Attention

The simplest possible expression for $\pi_i(\chi)$ is the identity function, in which each variable v_i has its own attention indicator χ_i:

$$\pi_i(\chi) = \chi_i \in \{0, 1\}, \text{ where } \sum_i \chi_i = n << N. \tag{67}$$

We have previously reported on this case in [MM91]. "The objective function for χ would be transformed into a clocked objective, as in (30) (again using the notation of section 2. 1.3):

$$E[\mathbf{v}] \rightarrow \left(\text{kWTA}(\chi, n) + \sum_i \chi_i E_{;i}[\bar{\mathbf{v}}]\right) \oplus E[\mathbf{v}\{\chi\}]. \tag{68}$$

This representation of $\pi_i(\chi)$ looks expensive, since any savings obtained by leaving most v_i's out of the focus of attention could be lost by updating all the χ_i variables each iteration. From equation (65) this update would also require computing $E_{;i}$ for every i, in the focus or not. But in fact $E_{;i}$ is unchanged unless v_i is in the focus of attention, or has a network neighbor in the focus; so for efficiency we can store this gradient information in a variable w_i which is only updated in those

These considerations can be formalized as a slight modification of the Lagrangian transformation point of view used in section 2.1 of Part I to derive a fine-scale Lagrangian for v. Now we are required to *partially* optimize an objective $E_{\text{coarse-}\chi}[\chi|\mathbf{v}]$, while guaranteeing the discreteness of χ. We will adapt the same three transformations as before. First we switch from discrete to constrained continuous optimization, accomplished in two successive phases using clocked objective function notation (2.1):

$$\psi_\chi(t)\left[C_\chi + F[\chi] + \Phi\left(\sum \pi_i(\chi) - n\right)\right] \rightarrow \psi_\xi(t)\left[\sum_{\text{all variables } x} K[\dot{x}, x] + F[\xi] + \Phi\left(\sum_i \pi_i(\xi) - n\right)\right] + \psi_{\hat{\chi}}(t)\left[\sum_{\text{all variables } x} K[\dot{x}, x] + \sum_a \hat{\chi}_a(\xi_a - \theta)\right], \tag{58}$$

where $\xi_i \in [0, 1]$, $\hat{\chi}_i \in [0, 1]$, θ is a threshold, and Φ is a sparseness term such as those of equation (27). Second, replace all constraints with penalty functions added to the objectives:

$$\begin{aligned} F[\xi] &\rightarrow E_{\chi\text{-opt}}[\xi] \equiv F[\xi] + \Phi\left(\textstyle\sum_i \pi_i(\xi) - n\right) + \textstyle\sum_a \phi(\xi_a), \\ \textstyle\sum_a \hat{\chi}_a(\xi_a - \theta) &\rightarrow E_{\text{restore}}[\hat{\chi}] \equiv \textstyle\sum_a \hat{\chi}_a(\xi_a - \theta) + \textstyle\sum_a \phi(\hat{\chi}_a). \end{aligned} \tag{59}$$

Here the threshold θ is usually taken to be 1/2, but other values may be used if the analog $\hat{\chi}$ dynamics would thereby be sped up without losing accuracy. Also $\xi(i) = \pi_i(\xi)$, as in equation (25). Note that the objective $E_{\text{restore}}[\hat{\chi}]$ is especially well-behaved among those we have considered, since the only way a large condition number or delay can arise is through the potential terms. The third transformation is to refine these coarse-scale objectives, and the usual volumetric cost terms, into fine-scale Lagrangians (cf. (57)):

$$\begin{aligned} C_\xi + F[\xi] + \sum_a \phi(\xi_a) &\rightarrow \sum_{\text{all variables } x} K[\dot{x}, x] + \nabla_\xi\left[F[\xi] + \Phi\left(\sum_i \pi_i(\xi) - n\right) + \sum_a \phi(\xi_a)\right]\cdot\dot{\xi} \\ C_{\hat{\chi}} + \textstyle\sum_a \hat{\chi}_a(\xi_a - \theta) + \textstyle\sum_a \phi(\hat{\chi}_a) &\rightarrow \sum_{\text{all variables } x} K[\dot{x}, x] + \nabla_{\hat{\chi}}\left[\sum_a \hat{\chi}_a(\xi_a - \theta) + \sum_a \phi(\hat{\chi}_a)\right]\cdot\dot{\hat{\chi}} \end{aligned} \tag{60}$$

These two Lagrangians, along with the usual one for v, must be reassembled into a full three-phase Lagrangian by multiplying by nonoverlapping clocks $\psi_\alpha(t)$ and summing over α as in section 2.1; that is the only way to express the action as a sum over algorithm time t (some $\int \cdot dt$ or some $\sum_t \cdot$) rather than over the intra-phase time variables τ_α.

3.3.2 Complete Multiphase Dynamics

We now have a 3-phase dynamics: First, choose the focus of attention using analog χ variables so as to optimize their estimated effect on ΔE subject to resource limitations. Second, discretize x. l'bird, relax $E[\mathbf{v}|\chi]$, using the chosen focus of attention. The analog χ phase includes a global k-winner constraint for $\pi(\chi)$. We will assemble the previously derived fine-scale cost and functionality terms for this net into an action functional and an associated clocked objective function.

Adding the partial Lagrangians of equations (57) and (60), we get a preliminary Lagrangian

$$\hat{L}_{\text{fine}} = \sum_{\text{phases } \alpha} \psi_\alpha(t)\left\{\sum_{\text{all variables } x} K[\dot{x}, x] + \sum_{\text{o-variables, } \mathbf{x}_\alpha} \frac{\partial E}{\partial \mathbf{x}_\alpha}\frac{\partial E_\alpha}{\partial \mathbf{x}_\alpha}\dot{\mathbf{x}}_\alpha\right\} \tag{61}$$

This Lagrangian presents a problem for times t between a-phases, when $\sum_\alpha \#_\alpha(t) = 0$, because at such times no dynamics is specified. The desired dynamics between phases is that all variables should be clamped. This can be ensured by adding a penalty term for movement of any variable between phases, in the form of a kinetic energy term K:

$$\hat{L}_{\text{extra}} = \left(1 - \sum_\alpha \psi_\alpha(t)\right) \sum_{\text{all variables } x} K[\dot{x}, x]. \tag{62}$$

Note that in physics, a Lagrangian consisting only of a kinetic energy term corresponds to a particle moving along a geodesic such as a straight line ($\ddot{x} = 0$), whereas here it corresponds to a variable clamped to a particular value.

circumstances. Also, the n-winner circuit can be implemented digitally as an incremental priority queue of w_i values. SO the clocked objective function becomes

$$
\begin{aligned}
E_{\text{queue}} &= \sum_i \left(w_i \left\{ \text{start} + \chi_i + \sum_j \text{Nbr}_{ij}\chi_j \right\} - E_{,i}[\bar{\mathbf{v}}] \right)^2 / 2 && \text{(transient terms)} \\
&\oplus\ \text{start}^2/2 + \sum_i \chi_i \bar{w}_i + \Phi\left(\sum_i \chi_i - n \right) + \sum_i \phi_{0/1}(\chi_i) && \text{(transient + control terms)} \\
&\oplus\ E[\mathbf{v}\{\boldsymbol{\chi}\}]. && \text{(subspace terms)}
\end{aligned}
\tag{69}
$$

Here "start" is initialized to unity and almost immediately changed to zero (in the second phase of the first clock cycle), and Nbr_{ij} is a constant 0/1 matrix recording whether neurons v_i and v_j are adjacent in the network or not:

$$
\text{Nbr}_{ij} = \begin{cases} 0 & \text{if } \max_{\mathbf{v}} \left| \frac{\partial^2 E}{\partial v_i \partial v_j}[\mathbf{v}] \right| = \mathbf{0}, \\ & \text{i.e. if } \max_{\mathbf{v}}(|\phi_i''(v)|) + |T_{ij}| + \sum_k |T_{ijk}| = 0; \\ 1 & \text{otherwise.} \end{cases}
\tag{70}
$$

Note that at the end of the first phase, $w_i = -E_{,i}[\bar{\mathbf{v}}]$. That's because (a) in the first cycle, $\text{start}_i = 1$, and every variable w_i is initialized to $-E_{,i}$; and (b) in subsequent cycles, either w_i is again set to the proper value, or else $\chi_i = 0$ and $\sum_j \text{Nbr}_{ij}\chi_j = 0$. In the latter case we know that w_i is unchanged from the previous cycle (since it is only changed in the first phase of any cycle), and also that $E_{,i}$ is unchanged from the previous cycle because it is unchanged by the dynamics of $E[\mathbf{v}\{\boldsymbol{\chi}\}]$'s relaxation:

$$
\begin{aligned}
\left| \frac{d}{dt} E_{,i} \right| &= \left| \frac{d}{dt} \left(\frac{\partial E}{\partial v_i} \hat{K}\left(-\frac{\partial E}{\partial v_i}, v_i\right) \right) \right| \\
&= \left| \frac{d}{dt} \left(\frac{\partial E}{\partial v_i} \right) \left(\hat{K} - \frac{\partial E}{\partial v_i} \hat{K}_{,w} \right) + \frac{\partial E}{\partial v_i} \frac{dv_i}{dt} \hat{K}_{,v} \right| \\
&= \left| \left(\sum_j \frac{\partial^2 E}{\partial v_i \partial v_j} \frac{dv_j}{d\tau_j} \frac{d\tau_j}{dt} \right) \left(\hat{K} - \frac{\partial E}{\partial v_i} \hat{K}_{,w} \right) + \frac{dv_i}{d\tau_i} \frac{d\tau_i}{dt} \frac{\partial E}{\partial v_i} \hat{K}_{,v} \right| \\
&\le \left(\sum_i \left| \frac{\partial^2 E}{\partial v_j \partial v_j} \right| \chi_j \left| \frac{dv_j}{d\tau_j} \right| \right) \left| \hat{K} - \frac{\partial E}{\partial v_i} \hat{K}_{,w} \right| + \chi_i \left| \frac{dv_i}{d\tau_i} \right| \left| \frac{\partial E}{\partial v_i} \hat{K}_{,v} \right| \\
&= 0 \ \left(\text{since } \chi_i + \sum_j \text{Nbr}_{ij}\chi_j = 0\right).
\end{aligned}
\tag{71}
$$

So throughout the second phase when χ is being determined, $\bar{w}_i = -E_{,i}[\bar{\mathbf{v}}]$.

Also note that in accordance with the definition in equation (11), the expression that controls the clamping of a variable such as w_i is implicitly held constant and need not be explicitly clamped. Only the second phase of equation (69) above has O(N) variables, and it can be replaced by a priority queue data structure with update cost $O(n \log N + cN)$, where k depends on digital hardware details and where c <<1 reflects the cost of storing w_i in inactive memory for future use, presumed to be relatively small.

Equation (69) assumes that n is constant. This assumption may be removed, if the coarse-scale cost of each n is modeled explicitly as mentioned in section 3.2.3. To a first approximation we may take the cost of a focus of attention to be proportional to its size, n, and ignore the effects of various different border shapes on the actual cost (these effects would tend to favor a focus with a small-boundary.) But what should the proportionality factor be between cost and benefit (ΔE) terms? To get sensible results we'll answer this question in an ad hoc way, not (yet) derived from fundamental considerations. Suppose that the cost of updating a neuron is dominated, not by space and time costs, but by the A E benefit *foregone* by not saving those same space-time resources to update some other neuron in the following iteration. To estimate that cost, per focal neuron, we multiply the average available ΔE per neuron by a constant f which must be meta-optimized. Then we have the following functionality expression.

$$
F[\boldsymbol{\chi}, n] = \sum_i \chi_i E_{,i}[\mathbf{v}] + \text{kWTA}(\boldsymbol{\chi}, n) + \frac{nf}{N} \sum_i |E_{,i}[\bar{\mathbf{v}}]| + \phi_{0/1}(n/N).
\tag{72}
$$

Optimizing this F may be achieved by (a) *sorting* i according to $E_{,i}$, for example incrementally with a priority queue data structure, and **(b)** turning on all χ_i for which $|E_{,i}|/(N^{-1}\sum_i |E_{,i}[\mathbf{v}]|) \geq f$. The focus of attention then consists of neurons whose single-neuron estimated contribution to ΔE is more than f times the average; it can range from none to all of the neurons. The potential function $\phi_{0/1}(n/N)$ can also be chosen so that the minimum focus size is one, rat her than none, of the neurons.

The focus of attention equation (67) provides maximal flexibility, since any subset of n out of N neurons in the network can be in the focus at one time. however, efficiency requires a hidden priority queue representation of $\pi(\chi)$, so that χ can be represented with only a marginal increment of space to encode this focus over that required by the n actual neurons in the focus at any time.

Generally such a representation is based on the binary addressing capabilities of a general-purpose computer. In fact the number of bits required in χ to specify such a focus is $\log_2 \binom{N}{71}$" For large N and n << N, this is approximately $n \log_2 N$ bits. \Ve can easily encode χ with this many bits, for example using the binary addresses of the n neurons in the unrestricted focus of attention. (Other efficient addressing schemes, such as Gray codes, would work too.) In radix (e.g. binary) notation for which $i = i_1 \ldots i_l$:

$$\chi(i) = \sum_a \prod_{b=1}^{l} \delta^K(\chi_{ab} - i_b) \tag{73}$$

(where χ_{ab} are binary-valued and δ^K is the Kronecker delta), or equivalently,

$$\chi(i) = \sum_a \prod_{b=1}^{l} \chi_{abi_b}, \quad \text{with} \quad \sum_{i_b} \chi_{abi_b} = 1. \tag{74}$$

If such a representation is substituted directly into a neural network objective function, rather than used in a hidden digital implementation of a stereotyped objective function such as the priority queue, then we get relatively intractable high-order objectives for χ (see [MG90] for an example of a sorting network using a similar high-order representation). Until this problem is solved by expressing some special- or general-purpose addressing and communication algorithms with simple clocked objective functions, we must appeal to non-neural switching circuits as necessary, taking care to estimate their costs. The clocked objective with brace notation $v\{\chi\}$ still specifies the *use* we make of such switching hardware, and would remain a useful notation even if we knew how to eliminate it in terms of clocked objectives without brace notation.

4.2 Multiscale Attention

The π_i (x) = χ_i representation of a focus of attention has the disadvantages of requiring a hidden, digital implementation (e.g. a priority queue) in order to be efficient, ancl of allowing foci without any coherent structure that might decrease the number of border neurons that are outside the focus but involved in the computational decision to move the focus, Both of these problems may be eliminated by restricting the focus of attention to a choice of one or several blocks of neurons, from a fixed partition of all the neurons into equal-sized blocks with low connectivity between the Mocks. An example of such a partition would be the division of the 2-d grid of the region-segmentation network (equation (19) in Part I) into A << N uniform rectangular sub-grids. Any such partition can be represented by a sparse, non-square 0/1 matrix B for which $\sum_a B_{ia} = 1$. Given such a partition, only one focus indication neuron χ_a is needed for each block $a \in \{1, \ldots . A << N\}$, rather than one per neuron index $i \in \{1, \ldots . N\}$. In return for increased efficiency in the attention mechanism as compared with the previous case, one gives up flexibility in the shape of the focus of attention. Some of that flexibility can be reacquired by generalizing the partition scheme described below to many levels in a recursive algorithm.

For a single level of partitioning, in which neurons v_i are grouped into fixed blocks a which enter or leave the focus together according to indicator neurons χ_a,

$$\pi_i(\chi) = \sum_a B_{ia}\chi_a, \tag{75}$$

where B is the constant partition matrix.

We could just substitute this expression for χ_i (or $\pi_i(\chi)$) into equation (69) (or (65)), in which case the most active blocks of the partition B would be the focus of attention. Attention would be a very affordable computation, a k-wi[lncr-take-all (kWTA) network. One clocked objective is simply

$$\begin{aligned} E_{\text{block}} &= \sum_a \chi_a \sum_i B_{ia} E_{;i}[\bar{\mathbf{v}}] + \Phi\left(\sum_a \chi_a - n(A/N)\right) + \sum_a \phi_{0/1}(\chi_a) \\ &\oplus \quad E[\mathbf{v}\{\sum_a B_{ia}\chi_a\}], \end{aligned} \tag{76}$$

which can again be improved by storing $E_{;i}$ as w_i, to be recalculated only as necessary, and which can be further improved by storing $\omega_a = \sum_i B_{ia} w_i$.

But here we will push the method a little farther, by choosing the k blocks not only based on their internal gradients but also on their predicted synergies with each other. The synergy is predicted by using the second order expansion for E, equation (46), which may be affordable now that we have only A focus-control neurons:

$$E[\chi] = \Delta\tau_{\mathbf{v}}(\frac{dE}{d\tau_{\mathbf{v}}})[\mathbf{v}\{B\chi\}] + \frac{\Delta\tau_{\mathbf{v}}^2}{2}(\frac{d^2E}{d\tau_{\mathbf{v}}^2})[\mathbf{v}\{B\chi\}]. \tag{77}$$

Then the clocked objective analogous to (69) is

$$\begin{aligned} E_{\text{block}} &= \sum_i \left(w_i\left\{\text{start} + \sum_a B_{ia}\chi_a + \sum_b \text{Nbr}_{ib}\chi_b\right\} - E_{;i}[\bar{\mathbf{v}}]\right)^2/2 \\ &\quad + \sum_{ij} \left(w_{ij}\left\{\text{start} + \sum_a (B_{ia}+B_{ja})\chi_a + \sum_c (\text{Nbr}_{ic}+\text{Nbr}_{ic})\chi_c\right\} - E_{;ij}[\bar{\mathbf{v}}]\right)^2/2 \\ &\oplus \quad \sum_a \left(\omega_a\left\{\text{start} + \chi_a + \sum_b \text{Nbr}_{ab}\chi_b\right\} - \sum_i B_{ia}\bar{w}_i\right)^2/2 \\ &\quad + \sum_{ab} \left(\omega_{ab}\left\{\text{start} + \chi_a + \chi_b + \sum_c (\text{Nbr}_{ac}+\text{Nbr}_{bc})\chi_c\right\} - \sum_{ij} B_{ia}B_{jb}\bar{w}_{ij}\right)^2/2 \\ &\oplus \quad \text{start}^2/2 + \sum_a \xi_a\bar{\omega}_a - \frac{\Delta\tau_{\mathbf{v}}}{2}\sum_{ab}\xi_a\xi_b\bar{\omega}_{ab} + \Phi\left(\sum_a \xi_a - n(A/N)\right) + \sum_a \phi_{0/1}(\xi_a) \\ &\oplus \quad -\sum_a \eta_a\bar{\chi}_a + \Phi\left(\sum_a \eta_a - n(A/N)\right) + \sum_a \phi_{0/1}(\eta_a) \\ &\oplus \quad -\sum_a \chi_a(\bar{\eta}_a - \theta) + \sum_a \phi_{0/1}(\chi_a) \\ &\oplus \quad E[\mathbf{v}\{\sum_a B_{ia}\chi_a\}], \end{aligned} \tag{78}$$

where we have introduced constant sparse matrices

$$\text{Nbr}_{ib} = \Theta\Big(\sum_j B_{jb}\text{Nbr}_{ij} - 1/2\Big) \tag{79}$$

ant]

$$\text{Nbr}_{ab} = \Theta\Big(\sum_{ij} B_{ia}B_{jb}\text{Nbr}_{ij} - 1/2\Big). \tag{80}$$

In (78), as in its prototype (46), the main departure from other clocked objective functions for attention is the quadratic objective function for ξ which expresses a nontrivial scheduling problem: which k neuron-blocks should be active simultaneously in order to maximize the expected sum of single-block and block-pair contributions to $|\Delta E|$? This quadratic optimization could be as hard as the original optimization problem E, were it not for the fact that it involves far fewer variables ξ_a. So it is crucial to have a separate restoration phase for χ in case the ξ analog scheduling optimization does not finish within its clock phase. In fact if the convergence time of the scheduling network isn't known well enough, we may need two restoration phases: one which restores ξ to an analog kWTA solution η, and a subsequent phase to ensure discrete 0/1 values χ for the attention control variables. This conservative approach to restoration is incorporated in equation (78).

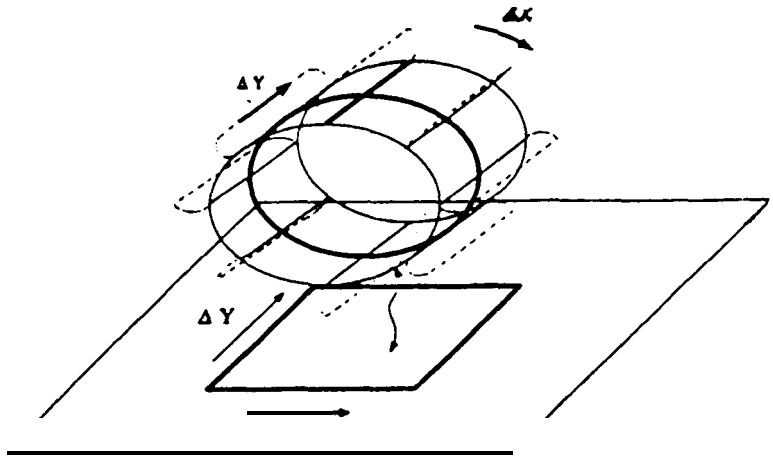

Figure 2: A rolling window of attention.

The scheduling network is a kind of auxiliary, coarse-scale network which controls attention at the level of blocks. Its connection matrix is surprisingly similar to part of a previously studied multiscale optimization neural network [MGM91], which also had an auxiliary coarse-scale network at the level of blocks of neurons. In that case the coarse-scale network was not for the purpose of control, but rather to accelerate the convergence of the much more expensive fine-scale network (which was simulated without any attention mechanism). In this regard the coarse-scale attention-control connection matrix ω_{ab} may be taken (as discussed in section 3.2.2) to be the negative of equation (50) after substituting (75) for $\pi_i(\chi)$; then it becomes identical to the coarse-scale acceleration connection matrix from [MGM91],

$$\hat{T}_{ab} = \sum_{ij} B_{ia} B_{jb} g'(u_i) g'(u_j) E_{,i} E_{,j} T_{ij}. \tag{81}$$

4.3 Jumping and Rolling Windows of Attention

The block-attentive neural network algorithm of equation (78) is equipped with a focus of attention that *jumps* from one block or combination of blocks to another in successive clock cycles. These jumps are rather expensive, since they involve storing the values of whole blocks of neurons which used to be in the focus of attention but no longer are, and retrieving from static memory the blocks of neurons which are newly promoted to the focus. A more gradual migration of neurons to and from the focus of attention is studied in this section, for networks with such a regular topology that the focus of attention can *roll* (i.e. move incrementally) from one region to another as well as jump.

A rolling focus of attention is one which moves incrementally, keeping most of its neurons assigned to the same implementation hardware. For example, consider a two-cl imensional mesh of neurons with local connectivity, as occurs for example in the region-segmentation objective function (19) of Part I. A small piece of such a mesh could be implemented by a two-dimensional VLSI chip in which a fraction of the chip area is devoted to end-around connections, giving the circuit the topology of a torus, together with some form of secondary storage for the many neuron values which are clamped and stored off-chip. The torus can roll in any direction. The situation is illustrated in figure 2. Consider also the assignment of physical (chip-implemented) neurons to the much larger set of *virtual neurons* comprising the neural network. A rolling motion allows this assignment to remain unchanged everywhere except at the boundaries of the chip, or equivalently the boundaries of the focus of attention. This minimizes the need for off-chip communication and on-chip analog shifting circuitry everywhere in the chip, at the expense of requiring dynamic boundary circuitry (probably digital) throughout the chip. An alter-native would be to allow the focus of attention to "slide" around the neural net instead, in which case the dynamic boundary circuitry may be eliminated in favor of the analog shifting circuitry. Our clocked objective function can be implemented either way. For clarity we will discuss the rolling case.

'1'0 describe the focus of attention mathematically, we just need $\pi(\chi)$. We want to use a set of blocks of neurons as in section 4.2, so that they can jump under the control of $\{y_a\}$, except that the blocks also roll (or slide) around the mesh. Each block's position can be characterized by its center. Block a has center $\mathbf{c}_a + \mathbf{x}_a$, in which $\mathbf{c}_a$ is a home position for block a *defined* by a fixed coarst'-scalc grid, and $\mathbf{x}_a$ is a dynamical displacement variable. The reason for including the home positions is to allow unused blocks to stay near their home positions, providing coverage of the alternative locations that the focus of attention can jump to. (This capability would not be necessary if blocks were only allowed to roll, but that would introduce spurious local minima into the attention mechanism, for example when a rolling window encounters its own or another window's path.) Then $\pi(\chi)$ is as in section 4.2, with $B_{ia} = b_i(\mathbf{c}_a + \mathbf{x}_a)$:

$$\pi_i(\chi) = \sum_a b_i(\mathbf{c}_a + \mathbf{x}_a)\chi_a. \tag{82}$$

We may scale our two-dimensional coordinates so that a block is a unit square, and we may assign addresses c_i in this coordinate system to each neuron i. We take $\mathbf{c}_a$ and $\mathbf{x}_a$ to be measured in this coordinate system also. Then the window boundary function b_i becomes

$$b_i(\mathbf{c}_a + \mathbf{x}_a) = b(\mathbf{c}_a + \mathbf{x}_a - c_i), \tag{83}$$

where

$$b(x) = \prod_{\alpha=1}^{\dim x} \Theta(1/2 - |x_\alpha|). \tag{84}$$

We will also have occasion to use a soft (differentiable) version of this window boundary function,

$$\tilde{b}_i(\mathbf{c}_a + \mathbf{x}_a) = \tilde{b}(\mathbf{c}_a + \mathbf{x}_a - \mathbf{c}_i), \tag{85}$$

where

$$\tilde{b}(\mathbf{x}) = \prod_{\alpha=1}^{\dim x} \tilde{\Theta}(1/2 - |x_\alpha|) \tag{86}$$

and

$$\tilde{\Theta}(x) = \begin{cases} 0, & x \le -w/2 \\ x/w + 1/2, & -1111/2 \le x \le w/2 \\ 1, & x \le w/2 \end{cases} \tag{87}$$

Then a clocked objective function for the rolling and jumping window of attention is

$$\begin{aligned} E_{j\,\&\,r}[\mathbf{v}, \boldsymbol{\xi}, \chi] \;=\; & \sum_i \Big(w_i \Big\{ \text{start} + \sum_a b_i(\mathbf{c}_a + \mathbf{x}_a)\chi_a + \sum_b \text{Nbr}_{ib}\chi_b \Big\} - E_i[\bar{\mathbf{v}}] \Big)^2 /2 \\ & \text{(compute the gradients)} \\ \oplus\; & \sum_a \Big[-\eta_a \Big(\text{start} + \bar{\chi}_a + \sum_b \text{Nbr}_{ab}\bar{\chi}_b - 1/2 \Big) + \phi_{0/1}(\eta_a) \Big] \\ & \text{(clamp unaffected windows)} \\ \oplus\; & \sum_a \Big(\omega_a\{\eta_a\} - \sum_i b_i(\mathbf{c}_a + \mathbf{x}_a)\bar{w}_i \Big)^2 /2 \\ & \text{(aggregate the gradients)} \\ \oplus\; & \text{start}^2/2 + \sum_a \Big[H(x_a\{\eta_a\}) + \sum_j \text{Nbr}_{aj}\tilde{b}_j(x_a\{\eta_a\})\bar{w}_j \Big] \\ & \text{(roll unclamped windows)} \\ \oplus\; & \sum_a \chi_a \Big[H(\bar{x}_a) + \sum_j b_j(\mathbf{c}_a + \bar{\mathbf{x}}_a)\bar{w}_j \Big] + \text{kWTA}(\chi, nA/N) \\ & \text{(select } k \text{ best windows \& jump there)}, \\ \oplus\; & E[\mathbf{v}\Big\{ \sum_a \chi_a b_i(\mathbf{c}_a + \mathbf{x}_a) \Big\}] \qquad \text{(descent within windows)}, \end{aligned} \tag{88}$$

where as before

$$\text{kWTA}(\chi, k) = \Phi\left(\sum_a \chi_a - k \right) + \sum_a \phi_a(\chi_a). \tag{89}$$

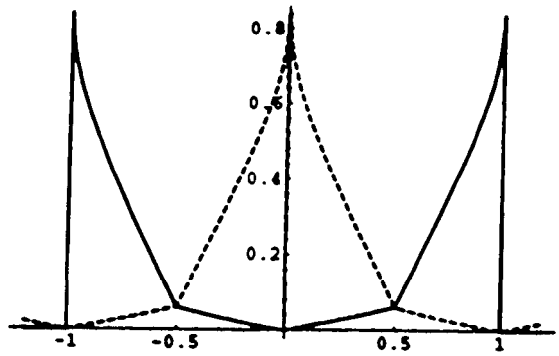

Figure 3: Spring function $\hat{H}(x) = \epsilon|x| + c_1\rho(|x| - 1/2) + \phi_{\pm 1}(\frac{x}{1-w/2})$, solid curve. First term restores $|x|$ to zero when block is out of the focus of attention. Second term favors hand-of to a neighboring block (neighboring block spring functions shown in dotted curves.) The third term is a barrier term, limiting the number of blocks that can be attracted to an attractive focal region of the network.

A crucial ingredient is the spring potential function H which allows a block o to move freely away from its home position until it is more than halfway into another block's territory, then to *hand* off the rolling window to a neighboring block 6 by turning off χ_a and turning on χ_b, and then to return to the borne position $\mathbf{x}_a$ = O to compute its expected ΔE and compete for another chance in the focus of attention.

A spring function that makes this possible is illustrated in figure 3. An explicit expression for H is

$$H(\mathbf{x}) = \sum_{\alpha=1}^{\dim \mathbf{x}} \hat{H}(x_\alpha), \tag{90}$$

where

$$\hat{H}(x) = \epsilon|x| + c_1\rho(|x| - 1/2) + 4 * 1(\frac{x}{1-w/2}), \tag{91}$$

and where

$$p(r) = \int_{-\infty}^{x} \Theta(x)dx = \begin{cases} o, & x \leq 0 \\ a, & 0 \leq x \end{cases} \tag{92}$$

4.4 Sparse Networks and Spreading Activation

'l'he attention mechanisms of the previous sections are designed to limit the number of active variables at any time, including both problem variables v and attention-control variables $\mathcal{X}$. However there is no attempt to limit the number of inactive variables whose values must still be stored and which therefore still occupy some hardware at all times. By imposing such a limit, we may be able to achieve far greater efficiency for optimization problems whose solutions are constrained to be sparse. What is required is that most of the variables outside the focus of attention should take on default values, such as zero, which need not be stored at all. The strategy is to enforce sparseness of v at every phase in every cycle, not just at the end of the computation. To achieve this we will allow mild expansions in the number of active neurons at some phases within a cycle, and enforce counterbalancing contractions in the number of active neurons at other phases in the cycle.

Suppose v is a set of N variables, constrained to be sparse in the sense that all but $n << N$ of them take (possibly identical) default values default(i) at any valid configuration. The default values may be zero or any number easily computed from the index r' alone, without the use of a large table of values (which would have to be stored). Let E(v) be an objective which includes penalty terms for sufficient sparseness constraints on at least some of the variables v, and which has the property that at any sparse configuration in which cn variables are unclamped in a focus of attention, all but n of the variables must approximate their default values at any local minimum. (Here c $\geq$ 1 is a constant,) Also suppose it is possible to initialize the network so that the focus of

attention contains all non-default variables (of which there are $\leq n$) and also all neighbors of such variables (of which there are $\leq cn$).

Then at the beginning of a relaxation phase for $E[\mathbf{v}\{\chi\}]$, all $\leq n$ non-default variables and all their $\leq cn$ neighbors are included in the focus of attention At the end of the relaxation phase, some new set of $\leq n$ variables have non-default values; the rest have near default values which can be reset to their default values without introducing much error, and which therefore do not need to be stored explicitly. In this way a limited front of activation relaxation, will propagate through the network of possible neurons which we shall refer to as *latent* neurons. The dynamics is reminiscent spreading activation or "marker propagation" algorithms in artificial intelligence [Fah79, '1ou86], and could perhaps be developed in that direction by using objective functions proposed in [MGA89]. Latent neurons are to be distinguished from the virtual neurons of previous sections (e.g. section 4.1), the latter requiring storage even when out of the focus of attention.

A suitable clocked objective function for such a spreading activation network, with many latent neurons, is

$$
\begin{aligned}
E_{\text{spread}} = & \sum_i \chi_i\{\chi_i\} + \sum_i \phi_{0/1}(\chi_i\{\chi_i\}) \\
& \oplus - \sum_i \chi_i\{s_i + \mathrm{Nbr}_{ij} s_j\} + \sum_i \phi_{0/1}(\chi_i\{s_i + \mathrm{Nbr}_{ij} s_j\}) \\
& \oplus E[\mathbf{v}\{\chi\}; 71, \epsilon] \\
& \oplus - \sum_i s_i\{\chi_i\}(\bar{v}_i - \mathrm{default}(i))^2/\epsilon^2 - 1) + \sum_i \phi_{0/1}(s_i\{\chi_i\}) \\
& \quad + \gamma\Phi(\sum_i s_i\{\chi_i\} - n) + \gamma \sum_i s_i\{\chi_i\} E_{,i}[\bar{\mathbf{v}}; n] \\
& \oplus -(1/2) \sum_i (v_i\{\chi_i(s_i - 1)\} - \mathrm{default}(i))^2 + \sum_i \phi_{0/1}(v_i\{\chi_i(s_i - 1)\}).
\end{aligned}
\tag{93}
$$

Here the first phase serves simply to find all nonzero x's and to set their values to zero. The second phase sets the focus of attention to include all non-default v_i's (for which $s_i = 1$) and their neighbors in the network topology. The third phase relaxes the network within the focus of attention, which we assume produces a new set of $\leq n$ variables v_i's which are not close to their default values. The fourth phase finds these variables and updates s_i to record them. Optionally, we can set $\gamma > 0$ to ensure what is already supposed to be guaranteed by E, that $s = 1$ for nonzero gradients and that $\sum_i s_i \leq n$. The fifth phase truncates near default values to exact default values, because neurons taking their default values do not need to be stored. (So in an implementation the fifth phase would not physically perform a truncation; it would simply de-allocate the hardware used to support the affected neurons.) The five phases together constitute one iteration of sparsity-preserving dynamics.

As an example of a suitable objective function E, we discuss a simple network for finding roots of a continuous function $f(x)$ of one variable $x \in [0, 1]$, by the bisection method. This network dynamically constructs a tree of at most n nonzero indicator neurons a_i, taken from an infinitely large tree of latent neurons. The network seeks large negative values of $f(x)f(x+\epsilon)$, and then bisects the interval $[x, x+\epsilon]$. Using multiple index notation $i \equiv i_1 i_2 i_l$, the search tree consists of all the latent 0/1 neurons ail.. i_l which take a value close to one if the search currently includes that node of the tree; also each node has a census neuron $m_{i_1 \ldots i_l} \in [0, n]$ which counts the number of neurons (including a's and m's) active at or below that node in the tree. These sets of variables would include the $l = 0$ versions, a and m without any indices, which are associated with the root of the search tree. The bisection search interval boundaries arc $x_0 = 0$, $x_1 = 1$, $x_{00} = 0$, $x_{01} = x_{10} = .5$, $x_{11} = 1$, and in general, $x_{i_1 \ldots i_n b} = \sum_{p=1}^{l} i_p 2^{-p} + b2^{-l}$.

Then a sparse objective function for this problem is

$$
\begin{aligned}
E_{\text{tree}} = & \sum_{l=1}^{\infty} \sum_{i_1 \ldots i_l = 0,1} a_{i_1 \ldots i_{l-1}} a_{i_1 \ldots i_l} g_{\pm}(C^l f(x_{i_1 \ldots i_l 0}) f(x_{i_1 \ldots i_l 1})) \\
& + (A/2) \sum_{l=0}^{\infty} \sum_{i_1 \ldots i_l = 0,1} (a_{i_1 \ldots i_l} + g_{\pm 1}(m_{i_1 \ldots i_l}) + m_{i_1 \ldots i_l 0} + m_{i_1 \ldots i_l 1} - m_{i_1 \ldots i_l})^2 \\
& + (A/2)(m/n - 1)^2 + \sum_{l=0}^{\infty} \sum_{i_1 \ldots i_l = 0,1} \phi_{0/1}(a_{i_1 \ldots i_l}) + \sum_{l=0}^{\infty} \sum_{i_1 \ldots i_l = 0,1} \phi_{0/1}(m_{i_1 \ldots i_l}/n),
\end{aligned}
\tag{94}
$$

where $g_{\pm}$ is an odd monotonic function with slow asymptotic growth, e.g. logarithmic growth. The network could be initialized with all a, m and s variables taking near-zero ($O(\epsilon) << 1$) values, except at the root where s = 1. At initialization all the non-zero gradients of E (which arise from the k-winner-take-all terms) are concentrated at the root and its immediate children $i = 0$ and $i = 1$.

A noteworthy property of the objective (94) is that the sparseness constraints are not global, but rather distributed over the topology of the network in such a way that an actual neuron a is involved in every term of the sparseness constraint. This prevents many census variables m from being given non-zero values in an effort to find one non-zero a variable. Instead, only as many census variables will be activated as needed. The $a_i + \hat{g}_{\pm 1}(m_i)$ summand in the k-winner term serves to include both a_i and m_i in the count of activated variables: $\hat{g}_{\pm 1}(m)$ is a sigmoid with values $\approx$ 771 for $m \ll 1$, and ≈ 1 for $m \geq 1$. The $\hat{g}_{\pm 1}(m)$ expression could be replaced by another O/l-valued neuron whose sole connection is to m.

We speculate that it may be possible to give a similar treatment of the conventional objective functions for inexact graph matching, such as [11'1'86]

$$\begin{aligned} E_{\text{match}}[M] &= -\sum_{ijab} G_{ij} g_{ab} M_{ia} M_{jb} + (A/2) \sum_i (\sum_a M_{ia} - 1)^2 + (A/2) \sum_a (\sum_i M_{ia} - 1)^2 \\ &\quad + B \sum_{ia} M_{ia}(1 - M_{ia}) + \sum_{ia} \phi_{0/1}(M_{ia}). \end{aligned} \tag{95}$$

However it is again necessary to localize the winner-take-all constraints, for example by embedding them in spanning trees for both G and g, in which each variable M_{ia} enters into each WTA constraint at its own location in the spanning tree. An additional attraction of such a sparse graph-matching network is that the E-relaxation phase of the clocked objective could actually be a nested loop which performs deterministic annealing in order to avoid local minima, Since successive cycles would have different foci of attention, the successive annealing procedures would be different - the high-temperature part of an annealing relaxation would not erase the progress towards a solution encoded in the focus of attention. A related technique for accelerating the convergence of matching networks by exploiting their sparseness was used in [LM94, GLR+95].

4.5 Orthogonal Windows

As suggested in [Mjo87], we can take advantage of the fact that some or all of the neurons in many hand-designed neural nets fall into natural cross-products, e.g. $v_i \equiv v_{i_1, i_2}$. An example is the graph-matching objective function of equation (95). In such cases we can greatly decrease the cost term by decomposing $\boldsymbol{\chi}$ and hope to retain functionality since it is only $\boldsymbol{\chi}$, not v, whose information content is thereby reduced. An obvious decomposition to try is:

$$\pi_i(\boldsymbol{\chi}) = \chi^{(1)}_{i_1} \chi^{(2)}_{i_2}, \tag{96}$$

i.e.

$$\boldsymbol{\pi}(\boldsymbol{\chi}) = \boldsymbol{\chi}^{(1)}_{i_1} \otimes \boldsymbol{\chi}^{(2)}_{i_2}, \tag{97}$$

where

$$\sum_i \pi_i(\boldsymbol{\chi}) = (\sum_{i_1=1}^{N_1} \chi^{(1)}_{i_1})(\sum_{i_2=1}^{N_2} \chi^{(2)}_{i_2}) \leq n. \tag{98}$$

'l'he last may be ensured by constraining

$$\sum_{i_1=1}^{N_b} \chi^{(b)}_{i_b} \leq n_b \quad (b \in \{1, 2\} \text{ and } n_1 n_2 \leq n). \tag{99}$$

For more than two terms in the cross product, all this generalizes to

$$\pi_i(\boldsymbol{\chi}) = \prod_b \chi^{(b)}_{i_b}, \tag{100}$$

where

$$\sum_{i_b=1}^{N_b} \chi^{(b)}_{i_b} \leq n_b \text{ and } \sum_i \pi_i(\boldsymbol{\chi}) = \prod_b n_b \leq n. \tag{101}$$

Following equation (68), we can use the clocked objective function

$$E_{\text{orthog}} = E_\chi[\boldsymbol{\chi}, \mathbf{v}] \oplus E[\mathbf{v}\{\boldsymbol{\chi}^{(1)} \otimes \boldsymbol{\chi}^{(2)}\}], \tag{102}$$

where

$$\begin{aligned} E_\chi[\boldsymbol{\chi}, \bar{\mathbf{v}}] \equiv & \sum_{\mathbf{i}} \chi^{(1)}_{i_1}\chi^{(2)}_{i_2} E_{;i}[\bar{\mathbf{v}}] + \Phi(\sum \chi^{(1)}_{i_1} - n_1) + \Phi(\sum_{i_2} \chi^{(2)}_{i_2} - n_2) \\ & + \sum_{i_1} \phi_{0/1}(\chi^{(1)}_{i_1}) + \sum_{i_2} \phi_{0/1}(\chi^{(2)}_{i_2}). \end{aligned} \tag{103}$$

A major problem with this scheme is that all the $E_{;i}[\bar{\mathbf{v}}]$ derivatives must be calculated, even though we want a small window of attention. A simple solution is to window the control variables $\boldsymbol{\chi}$ also, and only calculate the few that are necessary. There may be only $O(N_1 + N_2)$ of those, rather than $O(N)$. One possibility is the disjoint union focus of attention $\pi(\boldsymbol{\chi}) = (\boldsymbol{\eta}^{(1)}, \boldsymbol{\eta}^{(2)})$ for $\boldsymbol{\chi}$. We will apply transformation (68) twice: first to v, substituting $\pi_i(\boldsymbol{\chi}) = \chi^{(1)}_{i_1}\chi^{(2)}_{i_2}$ for χ_i, and then to X itself, using a straightforward focus of attention:

$$\pi_{(b,i_b)}(\boldsymbol{\eta}) = \eta^{(b)}_{i_b}, \text{ where } \sum_{i_b} \eta^{(b)}_{i_b} \leq cn_b. \tag{104}$$

¿From equations (41) and (42), we can calculate

$$E^{(1)}_{\chi;i_1}[\boldsymbol{\chi}, \bar{\mathbf{v}}] = E_{\chi,i}[\boldsymbol{\chi}, \bar{\mathbf{v}}]\hat{\chi}^{(1)}_{i_1} = -g'_\chi(g^{-1}_\chi(\chi^{(1)}_{i_1}))\Big(\sum_{i_2} \chi^{(2)}_{i_2} E_{;i}[\bar{\mathbf{v}}] + \ldots\Big) \tag{105}$$

and

$$E^{(2)}_{\chi;i_2}[X, \ldots] = E_{\chi,i}[\boldsymbol{\chi}, \bar{\mathbf{v}}]\hat{\chi}^{(2)}_{i_2} = -g'_\chi(g^{-1}_\chi(\chi^{(2)}_{i_2}))\Big(\sum_{i_2} \chi^{(1)}_{i_1} E_{;i}[\bar{\mathbf{v}}] + \ldots\Big) \tag{106}$$

Then the doubly attentive clocked objective function becomes

$$\begin{aligned} E_{\text{orthog}} = & E^{(1)}_{\chi;i_1}[\bar{\boldsymbol{\chi}}, \bar{\mathbf{v}}] + E^{(2)}_{\chi;i_2}[\bar{\boldsymbol{\chi}}, \bar{\mathbf{v}}] + \Phi(\sum_{i_1} \nu^{(1)}_{i_1} - cn_1) + \Phi(\sum_{i_2} \nu^{(2)}_{i_2} - cn_2) \\ & + \sum_{i_1} \phi_\nu(\nu^{(1)}_{i_1}) + \sum_{i_2} \phi_\nu(\nu^{(2)}_{i_2}) \\ \oplus & \sum_{i} \chi^{(1)}_{i_1}\{\nu^{(1)}_{i_1} + \chi^{(1)}_{i_1}\}\chi^{(2)}_{i_2}\{\nu^{(2)}_{i_2} + \chi^{(2)}_{i_2}\} E_{;i}[\bar{\mathbf{v}}] \\ & + \Phi(\sum_{i_1} \chi^{(1)}_{i_1}\{\nu^{(1)}_{i_1} + \chi^{(1)}_{i_1}\} - n_1) + \Phi(\sum_{i_2} \chi^{(2)}_{i_2}\{\nu^{(2)}_{i_2} + \chi^{(2)}_{i_2}\} - n_2) \\ & + \sum_{i_1} \phi_{0/1}(\chi^{(1)}_{i_1}\{\nu^{(1)}_{i_1} + \chi^{(1)}_{i_1}\}) + \sum_{i_2} \phi_{0/1}(\chi^{(2)}_{i_2}\{\nu^{(2)}_{i_2} + \chi^{(2)}_{i_2}\}) \\ \oplus & E[\mathbf{v}\{\boldsymbol{\chi}^{(1)} \otimes \boldsymbol{\chi}^{(2)}\}]. \end{aligned} \tag{107}$$

The first phase may be traded in as before for a priority queue implementation; but the space cost of the default circuit implementation is already so small ($O(n_1 + n_2)$ for the kWTA network) that the priority queue is not necessary. In the second phase at most $(c+1)^2 n^2$ gradients $E_{;i}$ must be calculated, As in previous networks, one could make the efficient calculation of all gradients explicit by adding extra phases and variables.

The focus of attention introduced in this section applies when the neuron index i takes values in some domain which is a cross product of other domains, domain(i) $=$ domain$(i_1) \times$ domain(i_2). 'T'his is of interest for building complex network architectures by composing simpler elements. Another natural operation on index domains is the disjoint union $i = (b, i_b)$. The E_χ example above showed how to compose a focus of attention for this case as well (see equation (104), with $\sum_b cn_b \leq$ the number $\hat{n}$ of active neurons allowed), though that case is much simpler than for the cross product.

5 DISCUSSION AND CONCLUSIONS

In part I of this work wc introduced a Lagrangian formulation of the relaxation dynamics of neural networks which compute try optimizing an objective function in a standard neural network

form. The Lagrangian formulation makes novel use of *a greedy functional derivative*, which we defined and computed. With these tools we demonstrated the use of three levels of optimization in the design of relaxation neural network dynamics: the original objective E, the Lagrangian L, and a Illcta-objective $\mathcal{M}$ which measures cost and functionality over many trials of the network.

In part II here we deal with a second group of more ramified applications. For these we introduced a clocked objective function and an associated notation. These constructs have the capability to clamp or unclamp net variables depending on the values of other of the net variables. This notation and tile stepwise refinement strategy for designing clocked objective functions sufficed to obtain computational attention *mechanisms*. Analogous to virtual memory or virtual processors in digital computers, such computational attention mechanisms have a focus of attention quality which can take a variety of forms. These include a priority queue, a set of coarse-scale blocks of neurons which could be scheduled according to their expected synergies in optimization, a set of jumping and rolling rectangular windows in a two-dimensional network, a sparse set of active neurons for which the excluded *latent* **neurons** require no memory, and the cartesian product of several simpler foci of attention. Each of these cases was concisely expressed using simple analytic notation with clocked objective functions. Reference was made to a number of experiments, application aucl computation, which employ the greedy variational and clocking calculus which we have introduced here.

Acknowledgements

We wish to acknowledge Charles Garrett and K. Srinivas for unpublished simulations; also discussions with Roger Smith, Chien-Ping Lu, Anand Rangarajan, Paul Cooper, Stan Eisenstat and Alain Martin; also the hospitality of the Institute for Theoretical Physics at Santa Barbara. This research was supported in part by AFOSR grant AFOSR-88-0240 and by ONR grant N00014-92-J-4048.

References

[111189] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151-160, Spring 1989.

[BSB+91] Bernhard E. Boser, Eduard Sackinger, Jane Bromley, Yann LeCun Richard E. Howard, and Larry D. Jackel. An analog neural network processor and its application to high-speed character recongnition. In *International Joint Conference on Neural Networks*, pages I- 415 to 1-421. IEEE, July 1991.

[BT89] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation*, chapter 3, pages 210-217. Prentice Hall, 1989.

[Coo89] Paul R. Cooper. *Parallel Object Recognition from Structure (The Tinkertoy Project)*. PhD thesis, University of Rochester Department of Computer Science, July 1989. Technical Report 301.

[Fah79] S. E. Fahlman. *NETL: A System for Representing and Using Real- World Knowledge*. MIT Press, 1979.

[GLR+95] Steven Gold, Chien-Ping Lu, Anand Rangarajan, Suguna Pappu, and Eric Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. In G. Tesauro, D. Touretzky, arid T. Leen, editors, *Advances in Neural Information Processing System 7*. MIT Press, 1995.

[HT86] J. J. Hopfield and D. W. Tank. Collective computation with continuous variables. In *Disordered Systems and Biological Organization*, pages 155--170. Springer-Verlag, 1986.

[LM94] Chien Ping Lu and Eric Mjolsness. Two-dimensional object localizat ion by coarse-to-fine correlation matching. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan -Kaufmann, 1994.

[MC80] Carver Mead and Lynn Conway. *introduction to VLSI Systems*. Addison-Wesley, 1980.

[MG90] Eric Mjolsness and Charles Garrett. Algebraic transformations of objective functions Neural **Networks, 3:651-669,** 1990.

[MGA89] Eric Mjolsness, Gene Gindi, and P. Anandan. Optimization in model matching and perceptual organization. *Neural Computation*, 1, 1989.

[MGM91] Eric Mjolsness, Charles D. Garrett, and Willard L. Miranker. Multiscale optimization in neural nets. *IEEE Transactions* on *Neural Networks*, 2(2), March 1991.

[Mjo87] Eric Mjolsness. Control of attention in neural networks. In *Proc. of First International Conference on Neural Networks, volume vol.* 11, pages *567-574.* IEEE, 1987.

[MM] Eric Mjolsness and Willard L. Miranker. A Lagrangian formulation of neural networks I: Theory and analog dynamics. Part I of this paper.

[MM91] Eric Mjolsness and Willard 1,. Miranker. A Lagrangian approach to fixed points. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Neural Information Processing System 3.* Morgan Kaufmann, 1991.

[MW66] W.L. Miranker and B. Weiss. 'l'he Feynman operator calculus. *SIAM Review*, 8:224-232, 1966.

[PB87] John C. Platt and Alan H. Barr. Constrained differential optimization. In Dana Z. Anderson, editor, *Neural information Processing Systems.* American Institute of Physics, 1987.

[1'S89] C. Peterson and B. Soderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems,* 1 (3), 1989.

[RGM96] Anand Rangarajan, Steven Gold, and Eric Mjolsness. A novel optimizing network architecture with applications. *Neural Computation, 8(5), 1996.*

[Tou86] David S Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann Pulishers, 1986.

[Tsi97] Dimitris Ioannis Tsioutsias. *Multiscale Attention as a Gloabally Convergent Framework for Large-Scale* Nonlinear *Optimization.* Phi) thesis, Yale University Computer Science Department, May 1997. See chapters 3 and 4.

[Vap82] V. N. Vapnik. *Estimation of Dependences Based* on *Empirical Data.* Springer Verlag, 1982.