HIGHER ORDER NEURAL NETS FOR TASK PLANNING IN DECOMPOSED STATE SPACES

GURSEL SERPEN and DAVID L. LIVINGSTON
Electrical Engineering and Computer Science Department
The University of Toledo
Toledo, OH 43606

Abstract:
A higher-order single-layer relaxation-type recurrent neural network is employed to plan a task in a decomposed state space. The finite state machine model of the state space of the complex task is decomposed into parallel/series combinations of component machines, each of which represents a subtask, using lattice theoretic techniques. Planning the complex task is realized by identifying the transfer sequence, an ordered set of inputs, of the component state machines for the subtasks. A fourth-order stochastic optimization neural network, single-layer relaxation-type recurrent network with simulated annealing, is utilized to perform the search needed to specify the transfer sequence in the state space of the complex task. The effect of decomposition on two essential areas is highlighted: significant reduction in computational resources required to implement the neural algorithm and the need to employ a high order neural network are emphasized.

INTRODUCTION

Given a complex task, the tendency of intelligent beings is to decompose the task into a set of workable subtasks. The divide-and-conquer approach has traditionally been applied to design problems such as software development in the form of structured programming and computer design where the basic computing system is composed of many subsystems. The ultimate goal is to be able to plan a task in an automated fashion such that the original high level task can be decomposed into lower level subtasks in the process.

The decomposition method requires a task to be represented as an algebraic structure. If the task can be represented as an algebraic structure, then the lattice-theoretic methods can be used to obtain a decomposition of the structure and thus the task. Of particular interest is the hierarchical relationship of subtasks; that is, the performance of a complex subtask may be accomplished by the execution of other, less complex, subtasks. These subtasks can be partially ordered into a hierarchical structure with the main task residing at the top of the hierarchy and subtasks, which are so simple as to be considered primitive residing at the bottom. Given the main task and the primitive operations, the objective now becomes one of finding the intermediate tasks and their relationships with each other, the main task, and the primitive operations.

As previously stated, once we have represented a task as an algebraic structure, the theories of lattices and universal algebras can be applied to examine the underlying structure of the task. The technique incorporates the following steps. The complete state space for the task under consideration is defined. A state machine[1] model of the task is developed with primitive operations as inputs to the machine. The state machine serves as an abstract tool to find the lattice of substitution property partitions from which the structure of the decomposition is completely identified. The underlying theory of algebraic structure decomposition, i.e., lattices, partitions, algebras and state machines, can be found in references (Hartmanis, 1966; Kohavi, 1978).

The foremost implication of task decomposition in terms of planning is that the state space search necessary to identify the transfer sequence can now be conducted in a decomposed state space rather than the composite state space, which is generally much larger. The main action of task planners is the searching of a very large state space in the presence of constraints. An important concern for real-time environments is that conducting this search in a serial fashion may not meet the time constraints of some tasks indicating a need for parallel methods. Artificial neural networks (ANN) are examples of implementations of algorithms that perform emergent computations in a parallel and distributed manner.

There is a large array of ANN algorithms suitable for application to optimization problems in the literature (Cichocki, 1993). The Hopfield network (HN) (Hopfield, 1986) and its derivatives are perhaps the most widely used ANN algorithms that address static optimization problems; they topologically belong to the class of single-layer, relaxation-type recurrent ANNs. The HN derivatives rely on gain scheduling as in simulated annealing (Sejnowski, 1986), network with nodes modeled by lossless integrators, network of nodes with unipolar activation functions, network with additive uncorrelated noise with zero mean and a variance gradually decreasing in time, mean field theory network, and mean field annealing network, among others. In practice, the HN and its derivatives offer a computationally simple way to address a class of optimization problems and effectively search for a local minimum of a energy (performance) function the definition of which is based on the network interconnection topology. In the Boltzmann machine (BM) case, each local minimum can be visited with a certain probability, which becomes larger as the minimum gets deeper: largest probability is associated with the global minimum in the energy function space.

HIGHER ORDER BOLTZMANN MACHINE AND TASK PLANNING

This research effort seeks to demonstrate the use of higher order Boltzmann machine to search for the shortest path in a decomposed state space. Analogously, we are looking for a transfer sequence of a decomposed state machine for a given initial and final state pair. A directed graph with non-weighted edges is a functional definition for a state machine and it is used as an abstract model for the transfer sequence search problem, where vertices of the graph represent the states and non-

[1] A finite state machine is defined as a three-tuple: $M = \langle I, S; f_s \rangle$, where I and S are finite, nonempty sets of inputs and states respectively. The mapping f_s is the state transition function with domain $I \times S$ and range S.

weighted directed edges stand for the transitions between states. The goal is to determine the shortest path between given initial and final vertices/states of the digraph/state machine.

As the result of a simple serial or parallel decomposition, two component digraphs with appropriate dependency information are generated by breaking down a composite digraph. Hence, the search for the shortest path now needs to be conducted over both component digraphs simultaneously. Another feature of searching a decomposed architecture is that one needs to define the initial and final nodes for each component digraph. Now, a path between any two vertex pairs of the composite digraph is equivalently two paths for the serially or parallel decomposed topology such that one path is associated with each component digraph. The shortest path between two vertices of a given decomposed directed graph can be defined using the following constraints:

1. the state transitions included in the path specification for each component machine must be valid, which translates into including only those edges that already exist in the associated digraph of the component machine,
2. the dependencies between component machines must be observed, equivalently the edges chosen from each digraph must be compatible with each other, and
3. the length of the path is equal to that power of the adjacency matrix of the composite digraph, which has the first nonzero entry in the row and column locations, defined by the initial and final vertices, respectively.

The *n*-th order Boltzmann machine is a stochastic single-layer recurrent network and will be employed in relaxation mode to search for the shortest path (Albizuri et al., 1995; DeGloria et al., 1993; Sejnowski, 1986). The performance function of an *n*-th order Boltzmann machine is defined as

$$E = -\frac{1}{N}\sum_{i_1}\sum_{i_2}\cdots\sum_{i_N} w_{i_1 i_2 \cdots i_N} s_{i_1} s_{i_2} \cdots s_{i_N} - \sum_{i_k=1}^{N} s_{i_k} b_{i_k}$$

where $w_{i_1 i_2 \cdots i_N}$ is an *N*-dimensional weight matrix symmetric on all pairs of indices, b_{i_k} is the external input to and s_{i_k} is the output of the computation node i_k with $k = 1 \ldots N$. Each computation node output is binary valued, zero or one, and the activation function is defined as

$$P\left(s_{i_k} = 1\right) = \frac{1}{1 + e^{-net_{i_k}/T}}$$

where P_{i_k} is the probability that s_{i_k}, the output of unit i_k, is equal to 1, T is a time-varying computational parameter analogous to annealing temperature and net_{i_k} is the input sum to unit i_k. The term net_{i_k} is defined by

$$net_{i_k} = \sum_{j_1} \sum_{j_2} \cdots \sum_{j_{N-1}} w_{j_1 j_2 \cdots j_{N-1}} s_{j_1} s_{j_2} \cdots s_{j_{N-1}},$$

where $i_k \notin \{j_1 j_2 \cdots j_{N-1}\}$.

A suitable Boltzmann machine network topology for a composite machine is a two-dimensional array of computation nodes with dimensions *M* and *N* where *M* is the number of states and *N* is the length of the longest path connecting any two states in the composite machine ($M \times N$ for the composite machine). Rows of the array are labeled by the states of the machine and the columns stand for the temporal position for the ordering of the states in the transfer sequence. This specific topology will accommodate all possible solutions for the transfer sequence search between any given pair of initial and final states. Decomposition of the state machine into component machines will also decompose the neural network array such that for a composite machine the decomposition process will reduce the row dimension *M*. For a two-component decomposition, the $M \times N$ topology can be partitioned into two arrays with dimensions $M_1 \times N$ and $M_2 \times N$ where M_1 is the number of states in the first component machine and M_2 is the number of states in the second component machine with $M_1 + M_2 < M$. This structure can be utilized to represent both parallel and serial two-component decompositions of a state machine. An example neural network topology for a serially decomposed two-component machine is shown in Figure 1 where $\{S_1, S_2, S_3\}$ and $\{S_4, S_5, S_6\}$ are states of M_1 and M_2, respectively. In the case of an *h*-component decomposition of an *M*-state machine, $h < M$, there will be *h* neural network arrays with dimensions $M_1 \times N, M_2 \times N, \cdots, M_h \times N$. Each computation node represents the hypothesis that if the node is active then the state associated with its row index is chosen as the element of the transfer sequence with its order in the sequence given by its column index while an inactive node simply implies that the unit takes no part in the transfer sequence specification. Task planning in a serially decomposed state machine using a higher order Boltzmann machine will be demonstrated in the next section.

TASK PLANNING IN A SERIALLY DECOMPOSED STATE MACHINE

A serially decomposed state machine will require a fourth order neural network architecture. The general form of the performance function for a fourth-order Boltzmann machine is

$$E = -\frac{1}{2} \sum_i \sum_j \sum_k \sum_l w_{ijkl} s_i s_j s_k s_l - \sum_i s_i b_i$$

where w_{ijkl} is the weight between computation nodes s_i, s_j, s_k, and s_l. The fourth-order weight term is defined by

$$w_{ijkl} = \sum_{\alpha} g_{\alpha} \delta^{\alpha}_{ijkl}$$

where α is the index over the set of constraints and $g_\alpha \in R^+$ if the hypothesis all nodes represent for constraint α are mutually supporting and $g_\alpha \in R^-$ if the hypothesis are conflicting. The term δ^{α}_{ijkl} is equal to 1 if the hypotheses represented by s_i, s_j, s_k, and s_l are compatible under constraint α and is equal to 0 otherwise.

A feasible transfer sequence for a given two-component serially decomposed state machine requires that,

1. state transitions for each component machine to be valid and
2. state transitions in each component machine to be compatible with respect to transitions in other interacting component machine for which a dependency relationship exists.

Given the definition of the constraints a transfer sequence specification has to satisfy, the next step is to map these constraints to the domain of the topology of the neural network. The constraint that imposes valid state transitions within each component state machine can be mapped by a second order connection by employing the following conditions:

1. $S_j = f_s^n(S_i, i)$ exists in the component machine M_n, $n \in \{head, tail\}$, for some input $i \in I$,
2. $S_i, S_j \in M_n$, and
3. $\left| col(S_i) - col(S_j) \right| = 1$,

where $col(S_i)$ is the column index of the state S_i. Then, following observations can be made:

1. $\delta^1_{ij} = 1$ and $g_1 \in R^+$ (excitatory interaction between nodes in the neighboring columns of the neural network array given that there is a transition between machine states) if all three conditions are satisfied,
2. $\delta^2_{ij} = 1$ and $g_2 \in R^-$ (inhibitory interaction between nodes in the neighboring columns of the neural net array given that there does not exist a transition between machine states) if conditions 2 and 3 are satisfied but condition 1 is not, and
3. $\delta^1_{ij} = \delta^2_{ij} = 0$ (no interaction between two nodes in the neural net topology given that nodes represent states belonging to different component machines or nodes are not in the neighboring columns) if condition 2 or 3 is not satisfied.

The dependency between component machines requires that corresponding state transitions in component machines must be compatible. For the cases of simple serial and parallel decomposition there are two component machines and thus a state transition in each component machine can be represented by a state pair implying that four states for two machines need to interact to verify the compatibility of state transitions. This constraint requires a fourth order connection for implementation. The conditions that map the dependencies between component machines are as follows:

1. The state transition $S_j = f_s^m(S_i, i)$ in component machine M_m, $m \in \{head, tail\}$, is compatible with state transition $S_k = f_s^n(S_l, i)$ in component machine M_n, $n \in \{head, tail\}$, such that $n \neq m$,
2. State transition $S_j = f_s^m(S_i, i)$ in component machine M_m, $m \in \{head, tail\}$, and state transition $S_k = f_s^n(S_l, i)$ in component machine M_n, $n \in \{head, tail\}$, such that $n \neq m$, are realized by the same input;
3. $S_i, S_j \in M_m$ and $S_k, S_l \in M_n$ such that $n \neq m$,
4. $\left|col(S_i) - col(S_j)\right| = 1 \ \wedge \ \left|col(S_k) - col(S_l)\right| = 1$ $\wedge \ \left[col(S_i) = col(S_l)\right] \ \wedge \ \left[col(S_j) = col(S_k)\right]$.

In the case of a serial decomposition, these constraints can be mapped to the neural network topology as follows:

1. $\delta^3_{ijkl} = 1$ and $g_3 \in R^+$ (excitatory interaction between compatible pairs of nodes from both component machines) if conditions 1, 3 and 4 are met,
2. $\delta^4_{ijkl} = 1$ and $g_4 \in R^-$ (inhibitory interaction between incompatible pairs of nodes from both component machines) if conditions 3 and 4 are met but condition 1 is not,
3. $\delta^3_{ijkl} = \delta^4_{ijkl} = 0$, otherwise.

As implied by the definitions of the connections between computation nodes there are two kinds of interaction between nodes; pairwise and quadwise. Pairwise connections are local to each component machine and impose the component machine structure, on the other hand quadwise connections are global and implement the compatibility of state transitions in a component machine with respect to other component machine. A weighting parameter is associated with each constraint. Values of these constraint weight parameters, namely g_1, g_2, g_3 and g_4, will be defined by employing the procedure presented in (Serpen, 1992) as follows.

Consider an active node in a column other than the initial and final columns of a component machine in the neural network solution array: e.g., the active node in row B_2 and column 2 of Figure 1. The active node in row B_2 and column 2 will receive second-order inputs from two other active nodes in the previous and next columns (neighboring columns) under the constraint that imposes the component state machine structure. In this case, these second-order inputs are associated with the constraint g_1. The same active node will also receive inputs under the fourth-order connection topology that imposes the compatibility of component machine state transitions. The active node in the neural network solution array will receive a total of two fourth-order excitatory inputs (related to the constraint g_3): one due to the transition from the previous state to current state (the triplet consisting of nodes at following locations: row B_1 and column 1, row B_4 and column 1, and row B_4 and column 2) and the other due to the transition from the current state to next state (the triplet consisting of nodes at following locations: row B_4 and column 2, row B_2 and column 3, and row B_6 and column 3). Noting that Boltzmann Machine node dynamics approximate the discrete Hopfield network node dynamics for sufficiently small computational temperature values as typical for convergence, the active node dynamics implies the following inequality on the constraint weight parameters:

$$net_{i_k} = 2g_1 + 2g_3 > 0$$

This inequality is satisfied for any values of the constraint weight parameters g_1 and g_3.

Similarly, an inactive node, e.g., the node at row B_4 and column 3 of Figure 1, will receive at most two excitatory inputs from two active nodes in the previous and next columns under the second-order connection topology, which imposes the component machine structure. The same inactive node may also receive at most two inhibitory inputs from two active nodes in the previous and next columns under the constraint, which imposes the component machine structure. However, total number of excitatory (related to the constraint g_1) and inhibitory (related to the constraint g_2) inputs for the inactive node must be equal to two since there are exactly two active nodes, one active node in the previous and the other active node in the next columns (with respect to the column of the inactive node) of the component machine topology. That inactive node in the neural network solution array will receive a total of two fourth-order inputs: one due to the transition from the previous state to current state and the other due to the transition from the current state to next state. Again, the total number of fourth-order excitatory (related to the constraint g_3) and inhibitory (related to the constraint g_4) inputs for the inactive node is equal to two: one triplet of active nodes representing the transition from previous state to current state and a second triplet of active nodes representing the transition from current state to next state. Then the total network input to the inactive node in the solution array is defined by

$$net_{i_k} = n_1 g_1 + n_2 g_2 + n_3 g_3 + n_4 g_4 \leq 0$$

where $n_1 + n_2 = 2$ and $n_3 + n_4 = 2$ with n_1, n_2, n_3 and $n_4 \in [0,2]$. This inequality can be rewritten as follows:

$$n_1 g_1 + n_3 g_3 \leq n_2 |g_2| + n_4 |g_4|$$

Note that there are seven feasible inequalities implied by this inequality. Two inequalities derived using (1,1,2,0) and (2,0,1,1) for (n_1, n_2, n_3, n_4) imply the remaining set of inequalities and establish the most constraining bounds as follows:

$$g_1 + 2g_3 \leq |g_2| \text{ and } 2g_1 + g_3 \leq |g_4|$$

These two inequalities indicate the set of possible values for constraint weight parameters to establish stability of solutions in the state space of the neural network dynamics.

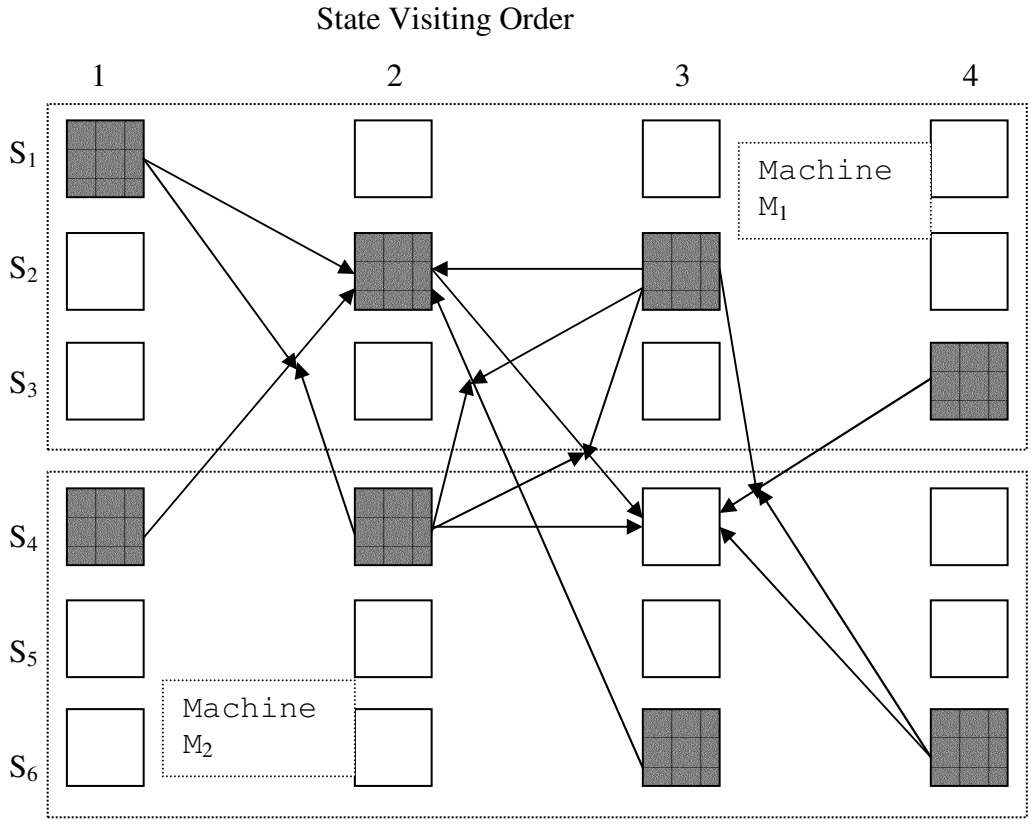

Figure 1. Neural Network Array for a Solution for Path Planning in a Serially Decomposed State Machine.

We have performed preliminary simulation based analysis of the performance of the high-order Boltzmann Machine for task planning in serially decomposed state spaces using the 3-disk Towers of Hanoi problem (Livingston & Serpen, 1989).

Simulation results indicate that the high order Boltzmann machine tends to converge to solutions in a large percentage of cases. However, the stable points network relaxes to include some non-solution points as well. We are currently investigating the hypothesis that the stable point set is a proper super set for the set of solution points in order to explain the convergence to non-solution points. It is also worth noting that we were able to easily determine a "good" initial value and schedule for the computational annealing parameter during the course of simulations.

CONCLUSIONS

We have demonstrated the use of a high order Boltzmann machine for task planning problem in decomposed state spaces. By using the decomposed form of the state machines that are models of the tasks, the number of processing elements in the Boltzmann machine can be reduced at the expense of high order connections.

We have demonstrated the need for high order machines in a class of problems, which may be of practical significance to task planning and execution. In the case of task planning in decomposed state spaces, it was observed that a high-order neural network was necessary to implement the compatibility relations between subtasks. It is still an open research question if the proposed task planning neural network will scale well with the large size state spaces.

We have employed a procedure to define the values of constraint weight parameters to establish the stability of problem solutions in the state space of the neural network dynamics. Preliminary simulation results indicated that the high order Boltzmann machine converged to a solution after majority of relaxations. Further research is needed to firmly establish the performance of the high order Boltzmann machine performance.

ACKNOWLEDGEMENT

This work was supported in part by NASA under contract NAS1-18584-12.

REFERENCES

F. X. Albizuri, A. d'Anjou, M. Grana, J. Torrealdea, and M. C. Hernandez, "The High-Order Boltzmann Machine: Learned Distribution and Topology," IEEE Transactions on Neural Networks, Vol. 6, No. 3, pp. 767-770, 1995.

A. Cichocki and R. Unbehauen, Neural Networks for Optimization and Signal Processing, Wiley, New York, 1993.

A. DeGloria, P. Faraboschi, and M. Olivieri, "Efficient Implementation of the Boltzmann Machine Algorithm," IEEE Transactions on Neural Networks, Vol. 4, No. 1, pp. 159-163, 1993.

J. Hartmanis and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1966.

J. J. Hopfield and D. W. Tank, "Computing with Neural Networks: A Model", Science 233, pp. 625-632, 1986.

Z. Kohavi, Switching and Finite Automata Theory. New York: McGraw-Hill, 1978.

D. L. Livingston and G. Serpen, "Evaluation of Lattice Theoretic Techniques for Task Decomposition Formalism," Proceedings of IEEE SOUTHEASTCON, pp. 1114-1119, 1989.

T. J. Sejnowski, "Higher Order Boltzmann Machines," AIP Conference Proceedings 151, Snowbird, UT, 1986.

G. Serpen, Bounds on Constraint Weight Parameters of Hopfield Networks for Stability of Optimization Problem Solutions, Dissertation, Old Dominion University, 1992.