1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1/94	Scientific Paper

4. TITLE AND SUBTITLE
Neural Network Solutions to Logic Programs with Geometric Constraints

5. FUNDING NUMBERS

6. AUTHOR(S)
Jo Ann Parikh
Anne Werkheiser
V.S. Subrahamanian

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Topographic Engineering Center
ATTN: CETEC-PAO
7701 Telegraph Road
Alexandria, VA 22310-3864

8. PERFORMING ORGANIZATION REPORT NUMBER
R-206

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release;
distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Hybrid knowledge bases (HKBs), proposed by Nerode and Subrahmanian, provide a uniform theoretical framework for dealing with the mixed data types and multiple reasoning modes required for solving logical deployment problems. Algorithms based on mixed integer linear programming techniques have been developed for the syntactic subset of HKBs corresponding to function-free Prolog-like logic programs. In this study, we examine the ability of neural networks to solve a more comprehensive set of problems expressed within the hybrid knowledge base framework.

The objective of this research is to design and implement a nonlinear optimization procedure for solving extended logic programs with neural networks. We focus upon two types of extensions which are typically required in the formulation of logical deployment problems. The first type of extension, which we shall refer to as a Type I extension, consists of embedding numerical and geometric constraints into logic programs. The second type of extension, which we shall call a Type II extension, consists of incorporating optimization problems into logic clauses.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Neural Networks, Logic Programs, Logical Deployment Problems, Geometric Constraints, and Optimization Problems	14
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	

Neural network solutions to logic programs with geometric constraints

Jo Ann Parikh
Computer Science Department
Southern Connecticut State University
501 Crescent Street
New Haven, CT 06515
E-mail : parikh@scsu.ctstateu.edu

Anne Werkheiser
U. S. Army Topographic Engineering Center
Fort Belvoir, VA 22060-5546
E-mail: anne@pooh.tec.army.mil

V.S. Subrahmanian
Computer Science Department &
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
E-mail : vs@cs.umd.edu

ABSTRACT

Hybrid knowledge bases (HKBs), proposed by Nerode and Subrahmanian, provide a uniform theoretical framework for dealing with the mixed data types and multiple reasoning modes required for solving logical deployment problems. Algorithms based on mixed integer linear programming techniques have been developed for the syntactic subset of HKBs corresponding to function-free Prolog-like logic programs. In this study, we examine the ability of neural networks to solve a more comprehensive set of problems expressed within the hybrid knowledge base framework.

The objective of this research is to design and implement a nonlinear optimization procedure for solving extended logic programs with neural networks. We focus upon two types of extensions which are typically required in the formulation of logical deployment problems. The first type of extension, which we shall refer to as a Type I extension, consists of embedding numerical and geometric constraints into logic programs. The second type of extension, which we shall call a Type II extension, consists of incorporating optimization problems into logic clauses.

1. INTRODUCTION

A potentially useful way of organizing the massive volume of terrain information available from multiple sensors with human expertise and decision-making criteria is through facts and rules that can be accessed by logic programs. In these systems numerical, geographical, and geometric information must be combined uniformly with logical information as in human inference to solve

logical deployment problems. The rules in the system must be able to express various types of relationships, including constraint relationships and relationships framed as optimization problems. Hybrid knowledge bases[1], proposed by Nerode and Subrahmanian, provide a general mechanism for uniformly dealing with mixed data types and multiple reasoning modes including reasoning about time and uncertainty.

The hybrid knowledge base (HKB) that was proposed[1] consists of a set of statements of the form:

$$A : [u_0, t_0] \leftarrow \Xi \parallel B_1 : [u_1, t_1] \,\&\, \ldots \,\&\, B_n : [u_n, t_n]$$

where $A, B_1, \ldots, B_n$ are atoms of an underlying logical language L, Ξ is a constraint over a constraint domain, Σ, and each u_i is a non-negative real number and t_i is a set of non-negative real numbers. Ξ is called the *constraint part* of the above clause, and $A : [u_0, t_0] \leftarrow B_1 : [u_1, t_1] \,\&\, \ldots \,\&\, B_n : [u_n, t_n]$ is called the *annotated clause part* of the above formula. The intuitive reading of the above clause is: "If Ξ is true with respect to the constraint domain Σ, and for all $1 \leq i \leq n$, B_i is true with a certainty of u_i or more at all time points in t_i, then A is true with certainty u_0 or more at all time points in t_0."

Nerode *et al.*[1] extend HKB theory to include non-monotonic modes of negation as well, but for the purposes of this paper, we will consider only positive HKBs obtained as follows: first, Σ is the domain of real numbers $\bigcup_{i=1}^{\infty} \mathrm{R}^i$. Thus, all constraints occurring in HKB clauses are over the real domain. Second, we assume that time is static, and uncertainties are not present. This means that the t_i's all refer to the same point in time, and the u_i's are all 1.

Development of efficient implementation paradigms for the full range of features encompassed by the HKB framework is a challenging problem. A mixed integer linear programming methodology has been developed by Jeroslow[2] and by Nerode and Subrahmanian[3] to compute stable models for function-free Prolog-like logic programs. In order to handle nonlinear constraints (without reducing them to linear constraints), a different methodology must be adopted.

In the class of Constraint Logic Programming (CLP) languages, due to Jaffar and Lassez[4], linear and nonlinear constraints in extended logic programs are solved by a combination of techniques that includes linear programming methods and unification. CLP(R), a constraint programming language over the domain of real numbers, solves linear constraints by generalizing unification and solves nonlinear constraints by delaying the solution until a sufficient number of variables have been solved to reduce the nonlinear constraints to linear constraints. CLP(M), a constraint programming language for solving optimization problems, uses the "ConstrainedMin" and "ConstrainedMax" routines in Mathematica to solve embedded optimization problems.

In this paper we describe a neural network approach for solving logical deployment problems expressed within the hybrid knowledge base framework. The approach concentrates on solving logic programs which contain clauses requiring satisfaction of geometric constraints and optimization problems in conjunction with logical constraints. The constraints express facts and relationships describing terrain characteristics, terrain reasoning, and user requirements.

The remainder of the paper is organized as follows. Section 2 introduces the nonlinear optimization procedure and an illustrative site identification system. Section 3 discusses the neural

network algorithm and numerical simulator. Section 4 contains experimental results obtained using the neural network approach on the site identification system. Conclusions and recommendations for future research are briefly outlined in Section 5.

2. NONLINEAR OPTIMIZATION PROCEDURE

The goal of the nonlinear optimisation procedure is to map deductive systems with embedded geometric constraints and/or embedded optimisation problems into combinatorial optimisation problems. Combinatorial optimisation problems can be defined as the class of problems which deal with maximising or minimising a function of many variables subject to inequality and equality constraints and integrality restrictions on some or all of the variables. We will be concerned with the equality-constrained optimization problem

$$\begin{aligned} &\text{minimize} && f(x), \\ &\text{subject to} && h(x) = 0, \end{aligned} \tag{1}$$

where $f : \mathbf{R}^n \rightarrow \mathbf{R}$, $h : \mathbf{R}^n \rightarrow \mathbf{R}^m$ and f and h are nonlinear functions.

The procedure to solve a logic program with recurrent neural networks consists of the following three steps:

1. Construction of Completion(Ground(P)),
2. Transformation of Completion(Ground(P)) into a Combinatorial Optimization Problem, and
3. Transformation of the Optimization Problem into an Energy Minimization Problem.

2.1. Construction of completion(ground(P))

In order to construct the completion of the ground version of a logic program P, variable-free clauses in ground(P) are converted into equivalence clauses. The ground version of P is obtained by replacing any logic clause in P that contains variables with the set of all clauses obtained by substitution of atoms for variables. In the completion of the ground version, no two clauses have the same head and the *implication* operator is replaced by the *equivalence* operator. For example, if there are two or more clauses with the same head, they can be combined as below:

$$\begin{aligned} B_{1,1} \,\&\, B_{1,2} \,\&\, \ldots \,\&\, B_{1,n} &\rightarrow A \\ &\cdots \\ B_{k,1} \,\&\, B_{k,2} \,\&\, \ldots \,\&\, B_{k,n} &\rightarrow A \end{aligned}$$

can be rewritten as

$$(B_{1,1} \,\&\, B_{1,2} \,\&\, \ldots \,\&\, B_{1,n}) \vee \ldots \vee (B_{k,1} \,\&\, B_{k,2} \,\&\, \ldots \,\&\, B_{k,n}) \leftrightarrow A$$

A more detailed description of the procedure for construction of the completion of the ground version of a logic program can be found in Bell *et al.*[3]

For the site identification system in Figure 1 below, the results of the first step of the procedure for the rules $best\_house(X,Y)$, $existing\_construction(X,Y)$, $possible\_building\_site(X,Y)$, and $possible\_restaurant\_site(X,Y)$ are respectively

$$
\begin{array}{lcl}
best\_house(a,b) & \leftrightarrow & house(a,b)\,\&\\
 & & (a,b) = arg\{max\,(distance\,from\,(R,S)\,to\,(2,2))\}.\\
existing\_construction(a,b) & \leftrightarrow & chemical\_plant(a,b) \vee house(a,b) \vee interstate(a,b).\\
possible\_building\_site(a,b) & \leftrightarrow & flat\_terrain(a,b)\,\&\,\neg\, existing\_construction(a,b).\\
possible\_restaurant\_site(a,b) & \leftrightarrow & possible\_building\_site(a,b)\,\&\,a>3\,\&\,b>3\,\&\\
 & & (distance\,from\,(a,b)\,to\,(5,5)<1.2).
\end{array}
$$

where a and b are constants obtained by substituting one of the values $1,2,3,4,5$ for X and Y.

Logic Program P:

$chemical\_plant(2,2).$

$flat\_terrain(1,1).$ $flat\_terrain(1,2).$ $flat\_terrain(1,3).$ $flat\_terrain(2,1).$ $flat\_terrain(2,2).$
$flat\_terrain(2,3).$ $flat\_terrain(2,4).$ $flat\_terrain(3,1).$ $flat\_terrain(3,2).$ $flat\_terrain(3,3).$
$flat\_terrain(4,1).$ $flat\_terrain(4,2).$ $flat\_terrain(4,4).$ $flat\_terrain(4,5).$ $flat\_terrain(5,2).$
$flat\_terrain(5,3).$ $flat\_terrain(5,4).$ $flat\_terrain(5,5).$

$house(1,4).$ $house(1,5).$ $house(2,5).$

$interstate(5,5).$

$lake(5,1).$

$$
\begin{array}{lcl}
existing\_construction(X,Y) & \leftarrow & chemical\_plant(X,Y).\\
existing\_construction(X,Y) & \leftarrow & house(X,Y).\\
existing\_construction(X,Y) & \leftarrow & interstate(X,Y).
\end{array}
$$

$$
possible\_building\_site(X,Y) \leftarrow flat\_terrain(X,Y)\,\&\,\neg existing\_construction(X,Y).
$$

TYPE I EXTENSION (Embedded Geometric Constraint):

$$
\begin{array}{lcl}
possible\_restaurant\_site(X,Y) & \leftarrow & possible\_building\_site(X,Y)\,\&\,X>3\,\&\,Y>3\,\&\\
 & & (distance\,from\,(X,Y)\,to\,(5,5)<1.2).
\end{array}
$$

TYPE II EXTENSION (Embedded Optimization Problem):

$$
\begin{array}{lcl}
best\_house(X,Y) & \leftarrow & house(X,Y)\,\&\\
 & & (X,Y) = arg\{max\,(distance\,from\,(R,S)\,to\,(2,2),\,R,S \in \{1,2,3,4,5\})\}.
\end{array}
$$

Figure 1: Site Identification System

2.2. Construction of the combinatorial optimization problem

The next step in the nonlinear optimization procedure is to construct the constraint functions and the objective function for the combinatorial optimization problem. Each logical predicate appearing in the completion of the ground version of a logic program is associated with a real variable in the range $[0, 1]$ with 0 denoting a *true* predicate and 1 a *false* predicate. Upper and lower boounds are established for any slack variables which may be added when transforming inequality constraints in the logic program into equality constraints for the combinatorial optimization problem. For logic programs with no extensions, the objective function is formulated as the negative of the sum of the associated variables. Minimizing the objective function while satisfying the system of constraints given in equation (1) ensures that any logical predicate which cannot explicitly be inferred as *true* will receive a value of *false*.

The system $h(x) = 0$ of numerical constraints represents the following categories of logical constraints:

- Fact Constraints
- Domain Constraints
- Logic Constraints
- Geometric Constraints
- Optimization Constraints

Fact constraints are represented by equating the associated variable to zero. For Logic Program P, there are 24 facts in the completion of the program, each of which has a corresponding variable associated to it. The fact *chemical_plant*$(2, 2)$, for example, is represented by a variable $C_{2,2}$. The constraint equation that requires this fact to be *true* is $C_{2,2} = 0$. $C_{2,2}$ will be zero when the corresponding predicate is *true* and non-zero when the corresponding predicate is *false*.

Domain constraints are also imposed on the values of the associated variables. In order to ensure that the value of each associated variable will be either *true* or *false*, constraints of the form $V(1 - V) = 0$ must be included for each variable associated with a logical predicate. For the site identification system without any extensions, there are a total of 175 domain constraints. Each of the seven predicates in the program, i.e. *chemical_plant*(X, Y), *flat_terrain*(X, Y), *house*(X, Y), *interstate*(X, Y), *lake*(X, Y), *existing_construction*(X, Y), and *possible_building_site*(X, Y), expands into 25 predicates in the program completion, making a total of 175 associated variables.

Logic constraints for a given clause are represented by equating to zero functions that are non-zero for a combinations of truth values that is logically invalid and which are zero for all logically valid sets of truth values. The construction of appropriate sets of functions can be completely automated. First, we will illustrate the technique for selected logic clauses in Logic Program P and then we will outline the general procedure.

The technique for the function construction is illustrated below for the two clauses in Logic Program P, *existing_construction*(a,b) and *possible_building_site*(a,b). The following notation will be used. Associated with predicates *existing_construction*(a,b), *chemical_plant*(a,b), *house*(a,b), *interstate*(a,b), *possible_building_site*(a,b), and *flat_terrain*(a,b) are the variables $E_{a,b}$, $C_{a,b}$, $H_{a,b}$, $I_{a,b}$, $P_{a,b}$, and $F_{a,b}$, respectively. The invalid sets of truth combinations for ($E_{a,b}$,$C_{a,b}$,$H_{a,b}$,$I_{a,b}$) are {t,f,f,f}, {f,t,t,t}, {f,t,t,f}, {f,t,f,t}, {f,t,f,f}, {f,f,t,t}, {f,f,t,f}, and {f,f,f,t} where *t* denotes *true* and *f* denotes *false*. The invalid sets of truth combinations for ($P_{a,b}$,$F_{a,b}$,$E_{a,b}$) are {t,t,t}, {t,f,t}, {t,f,f}, {f,t,f}. For each invalid truth combination, a function is formed by associating a value of $(1-V)$ to a truth value of *t*, a value of V to a truth value of *f*, and then multiplying together the terms corresponding to truth values appearing in an invalid combination.

The corresponding constraints for *existing_construction*(a,b) and for *possible_building_site*(a,b) are given below:

Logic Constraints for existing_construction(a,b)

$$(1-E_{a,b})C_{a,b}H_{a,b}I_{a,b}=0$$
$$E_{a,b}(1-C_{a,b})(1-H_{a,b})(1-I_{a,b})=0$$
$$E_{a,b}(1-C_{a,b})(1-H_{a,b})I_{a,b}=0$$
$$E_{a,b}(1-C_{a,b})H_{a,b}(1-I_{a,b})=0$$
$$E_{a,b}(1-C_{a,b})H_{a,b}I_{a,b}=0$$
$$E_{a,b}C_{a,b}(1-H_{a,b})(1-I_{a,b})=0$$
$$E_{a,b}C_{a,b}(1-H_{a,b})I_{a,b}=0$$
$$E_{a,b}C_{a,b}H_{a,b}(1-I_{a,b})=0$$

Logic Constraints for possible_building_site(a,b)

$$(\ -P_{a,b})(1-F_{a,b})(1-E_{a,b})=0$$
$$(1-P_{a,b})F_{a,b}(1-E_{a,b})=0$$
$$(1-P_{a,b})F_{a,b}E_{a,b}=0$$
$$P_{a,b}(1-F_{a,b})E_{a,b}=0$$

The general procedure for construction of functions for nonlinear equality constraints to represent logic constraints consists of the following steps:

1. Enumerating Invalid Truth Value Combinations for Clauses in Completion(Ground(P)),
2. Associating *true* to a term of the form $(1-V)$ and *false* to a term of the form V, and
3. Multiplying Associated Terms for Each Invalid Combination.

Numeric and geometric constraints embedded in rules are treated analogously to logical constraints. During the process of substitution of values for variables in the first stage of the procedure,

several simplifications may be directly applied. For example, in the site identification system, possible sites considered for a restaurant may be limited to those sites where X and Y are both greater than 3. This means that there are possibly four sites, sites $(4,4)$, $(4,5)$, $(5,4)$, and $(5,5)$, which need to be considered in the formulation of the optimization problem. All possible truth combinations for the rule for *possible_restaurant_site* are then considered. The inequality constraint that the distance between a possible restaurant site and the site $(5,5)$, where the interstate is located, must be less than 1.2 is turned into an equality constraint by adding a slack variable. Thus, the distance condition inequality $d((a,b),(5,5)) < 1.2$ becomes a distance condition equality of the form

$$d((a,b),(5,5)) - 1.2 + slack(a,b) = 0 \tag{2}$$

where $slack(a,b)$ is the slack variable associated with $d((a,b),(5,5))$. Let us denote the square of the expression on the left-hand side of the equality in equation 2 above as $Dist^2_{a,b}$. Whenever, in one of the invalid sets of truth combinations, the truth value for the distance condition evaluate to *false*, $Dist^2_{a,b}$ is inserted into the associated optimization constraint product. When the truth value for the distance condition is *true*, a desirable function for the associated term in the product would have the property of being close to zero when the distance condition is not satisfied and non-zero if the distance condition is satisfied. A function which we have used in our experiments is $1/(1 + exp(Dist^2_{a,b}))$. Using this function, the logic constraints for *possible_restaurant_site* are

Geometric Constraints for possible_restaurant_site(a,b)

$$(1 - R_{a,b})P_{a,b} = 0$$
$$(1 - R_{a,b})(1 - P_{a,b})Dist^2_{a,b} = 0$$
$$R_{a,b}(1 - P_{a,b})(1/(1 + exp(Dist^2_{a,b}))) = 0$$

where $R_{a,b}$ and $P_{a,b}$ are the variables associated respectively with *possible_restaurant_site* and *possible_building_site*(a,b), respectively.

Embedded optimization problems in logic clauses effect both the format of the optimization constraints and the format of the objective function. If the sample logic program consists only of logical information or logical information combined with geometric constraints, the objective function to be minimized is simply the negative of the sum of the associated variables. With an embedded optimization problem, both the negative of the sum of the associated variables and the embedded optimization condition must be simultaneously optimized. In addition to the logical constraints, a uniqueness constraint must be added to ensure that exactly one solution is found to the optimization problem. The form of the constraints for the *best_house* clause, which contains an embedded maximization problem, is illustrated below:

Optimization Constraints for best_house(a,b)

$$24 - \sum_{a,b \in \{1,2,3,4,5\}} B_{a,b} = 0$$
$$(1 - B_{a,b})H_{a,b} = 0$$

where $B_{a,b}$ and $H_{a,b}$ are the variables associated respectively with *best_house*(a,b) and *house*(a,b). The form of the objective function for the Site Identification System consisting of Logic Program P with TYPE II extension for *best_house*(a,b) is

Objective Function for Logic Program P with TYPE II Extension

$$-\sum_{a,b\in\{1,2,3,4,5\}} C_{a,b}+F_{a,b}+H_{a,b}+I_{a,b}+L_{a,b}+E_{a,b}+P_{a,b}+B_{a,b}$$

$$-\sum_{a,b\in\{1,2,3,4,5\}} (\textit{distance from } (a,b) \textit{ to } (2,2))(1-B_{a,b})$$

where $B_{a,b}$, $C_{a,b}$, $E_{a,b}$, $F_{a,b}$, $H_{a,b}$, $I_{a,b}$, $L_{a,b}$ and $P_{a,b}$ are the variables associated respectively with *best_house*(a,b), *chemical_plant*(a,b), *existing_construction*(a,b), *flat_terrain*(a,b), *house*(a,b), *interstate*(a,b), *lake*(a,b), and *possible_building_site*(a,b).

2.3. Transformation of optimization problem into energy minimization problem

The goal of this step of the nonlinear optimization procedure is to transform the combinatorial optimization problem into an energy minimization problem which can be solved by recurrent neural networks. An energy function is constructed consisting of the sum of the constraint functions and the objective function. The basic form of the resultant energy function E is

$$E=\sum_i A_i(\text{"violation of constraint i"})+B(\text{"}cost\text{"}) \tag{3}$$

where $A_i, B > 0$ and "cost" is an optimization cost function that is independent of constraint violations.[5]

3. NEURAL NETWORK MODEL

The motivation behind using neural network approaches for solving combinatorial optimization problems is to find near-optimal solutions in reasonable amounts of time to problems of realistic sizes.[5] Many combinatorial optimization problems, such as the Traveling Salesman Problem, are NP-complete problems (problems whose solution time is of order $exp(n)$). For this class of problems, which, in addition to being NP-complete, typically have associated energy functions with multiple local minima, Looi[5] points out that "optimal solutions are unattainable for problems of realistic sizes" and that "neural network models can be promising for certain optimization problems which can be readily expressed as the minimization of an energy function".

Recurrent neural networks are single-layer feedback networks which represent nonlinear dynamical systems. The state trajectories of the networks evolve over time to attractors which tend to minimize an associated energy function. Specific classes of recurrent neural networks have been identified which are asymptotically stable and which can generate optimal solutions to linear programming problems. Hopfield[6] and Cohen and Grossberg[7] have shown that discrete and analog recurrent neural networks with no auto-connections and with symmetric connection weights are asymptotically stable. Wang[8] has developed sufficient conditions for asymptotic stability of analog recurrent neural networks with time-varying interconnection weights. Wang[8] also developed sufficient conditions for the feasibility and optimality of solutions to convex programming problems generated by this class of recurrent neural networks.

A numerical simulator for recurrent neural networks with time-varying penalty parameters, which was developed by Wang[8] and modified based on suggestions by the author, was used to solve the energy minimization problems obtained by applying our nonlinear optimization procedure to logic programs. A brief overview of Wang's technique (as used in our experiments) appears below. Experimental results for the Site Identification System (Figure 1) are described in the next section.

3.1.Energy Function Design

The first task in the development of a recurrent neural network for solving an optimization problem is to design an energy function whose minimum represents the solution to the optimization problem. Let us assume that the optimization problem can be formulated as a nonlinear programming problem with a single objective function $f(\vec{v})$ in which the goal is to

$$\text{minimize } f(\vec{v}) \text{ subject to the constraint } p(\vec{v}) = 0$$

The penalty function $p(\vec{v})$ is assumed to be a non-negative, differentiable function which is equal to zero if and only if $\vec{v}$ is a feasible solution to the optimization problem. An energy function can then be defined as

$$E[\vec{v}(t), \lambda(t)] = f(\vec{v}(t)) + \lambda(t)p(\vec{v}(t))$$

The penalty parameter $\lambda(t)$ is assumed to be a positive, monotonically increasing function of t and $\vec{v}(t)$ which is initialized to a low value and increased slowly to avoid numerical instability and overpenalization. A suitable function for $\lambda(t)$ is

$$\lambda(t) = \sum_{\tau=0}^{t} \frac{\sqrt{\tau}}{p(\vec{v}(\tau))c_\lambda}$$

3.2. Algorithm for numerical simulation

The next task is to develop an update algorithm so that the system will evolve in the general direction of the negative gradient of the energy function[9] to a stable minimum. For the numerical simulation, the network was discretized based on the first-order Euler's method. In addition to selecting a function for the penalty parameter, the capacitive parameter c_μ must be determined and activation functions F must be selected for each neuron. Sigmoidal activation functions are typically used for mapping outputs back to inputs in recurrent neural networks. For a variable ranging between $[0, M]$, a sigmoidal activation function may be constructed as follows:

$$v_i(t) = \frac{M}{1 + e^{-u_i}} \quad \text{where } u_i, v_i \text{ are the input, output respectively for neuron } i$$

The following update algorithm (slightly modified from the algorithm in Wang[3]) was used for the experiments:

Step 0: Input values for the parameters Δt, c_λ, $\lambda(0)$, and ϵ (used in the termination criterion). Also input initial values for the vector $\vec{v}$ and determine the corresponding initial input vector $\vec{u}$ by taking the inverses of the sigmoidal activation functions.

Step 1: Evaluate the gradient by computing

$$\vec{g}(t) = \nabla^T f(\vec{v}(t)) + \lambda(t)\nabla^T p(\vec{v}(t))$$

Step 2: Compute the new state:

$$\vec{u}(t + \Delta t) = \vec{u}(t) - \Delta t \vec{g}(t) p(\vec{v}) c_\mu$$
$$\vec{v}(t + \Delta t) = F[\vec{u}(t + \Delta t]$$

The capacitive parameter c_μ is defined as $\frac{1}{\sqrt{t}}$ and F denotes sigmoidal transformations. The parameter c_μ should be selected sufficiently small to make the network stabilise quickly.

Step 3: Adapt the penalty parameter:

$$\text{if } E(t) \leq E(t + \Delta t) + \epsilon, \text{ then } \lambda(t + \Delta t) = \lambda(t) + \frac{\sqrt{t}}{p(\vec{v}) c_\lambda}$$

Step 4: Check for termination:

$$\text{if } p(\vec{v}) > \epsilon \text{ then STOP else go to Step 1.}$$

4. EXPERIMENTAL RESULTS

Experiments were conducted using the neural network approach on the Site Identification System for (1) Logic Program P, (2) Logic Program P with the TYPE I extension, and (3) Logic Program P with the TYPE II extension. The initial terrain configuration specified by the set of facts in Logic Program P is illustrated in Figure 2 below. All variables associated with known facts or with predicates which did not appear at the head of non-empty clauses were pre-set to 0 or 1 respectively and not allowed to change during the neural network update process.

For the first case, Logic Program P, convergence to a correct solution was obtained after only three iterations. The twenty-five variables associated with the *existing_construction* clause and the twenty-five variables associated with the *possible_building_site* clause were correctly identified as 0 for *true* and 1 for *false*. The results are illustrated in Figure 3 below.

For the second optimization problem, Logic Program P with TYPE I extension, convergence was obtained in under 10,000 iterations to the correct solutions. Figure 4 shows the solutions obtained for the four variables associated with *possible_restaurant_site* and for the fifty variables associated with the *possible_building_site* and *existing_construction* clauses. The neural network correctly deduced that a restaurant could not be built on site (5,5) because it was not a possible building site and that a restaurant could not be built on site (4,4) because it was not within a distance of 1.2 from the interstate site. Sites (4,5) and (5,4) were found to be possible sites for building a restaurant.

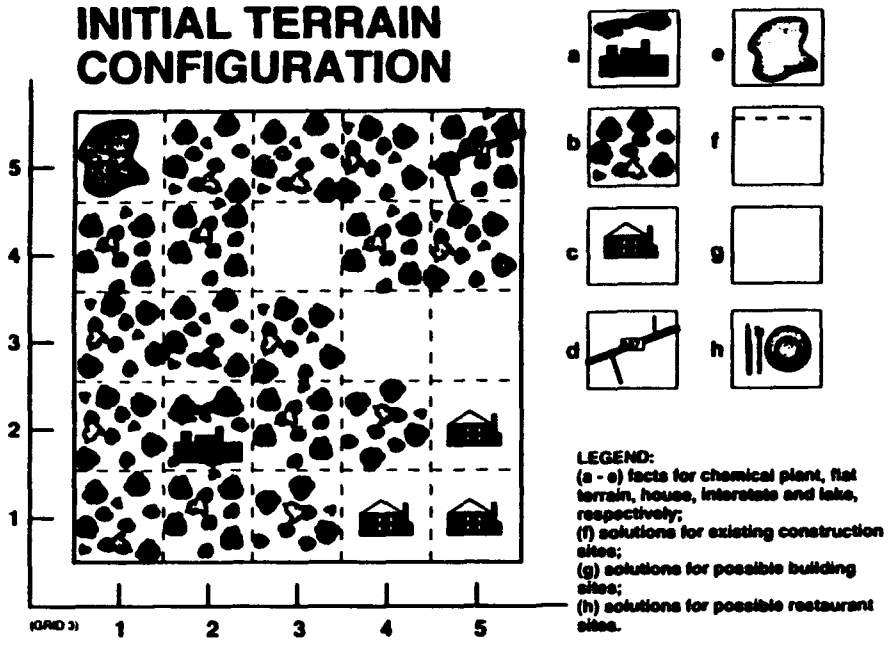

Figure 2: Initial Terrain Configuration for Site Identification System: (a-e) sites for chemical plant, flat terrain, house, interstate, and lake, respectively

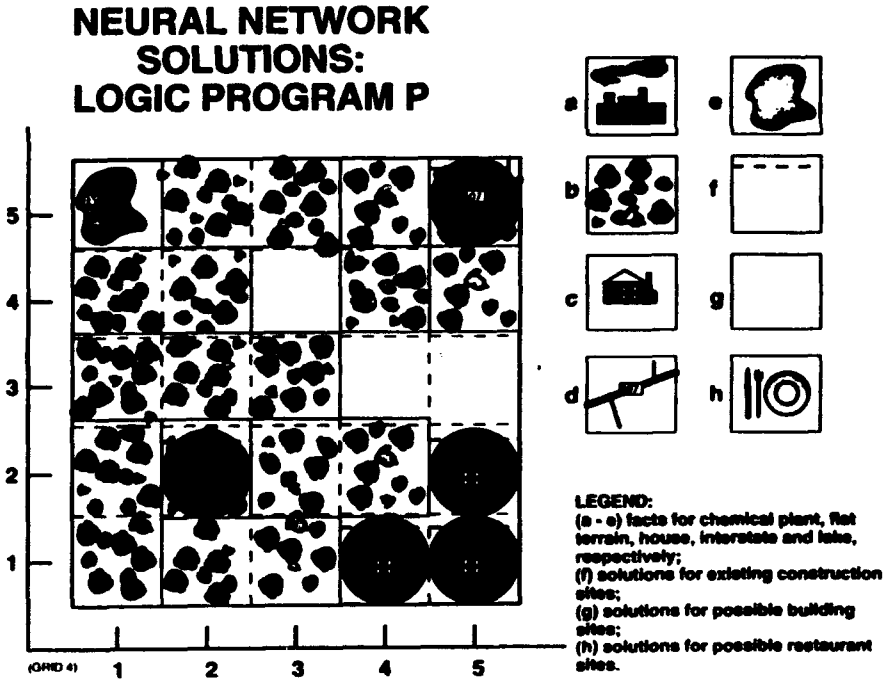

Figure 3: Solutions to Nonlinear Optimization Problem for Logic Program P: (a-e) sites for chemical plant, flat terrain, house, interstate, and lake, respectively; (f) solutions for existing construction sites; (g) solutions for possible building sites

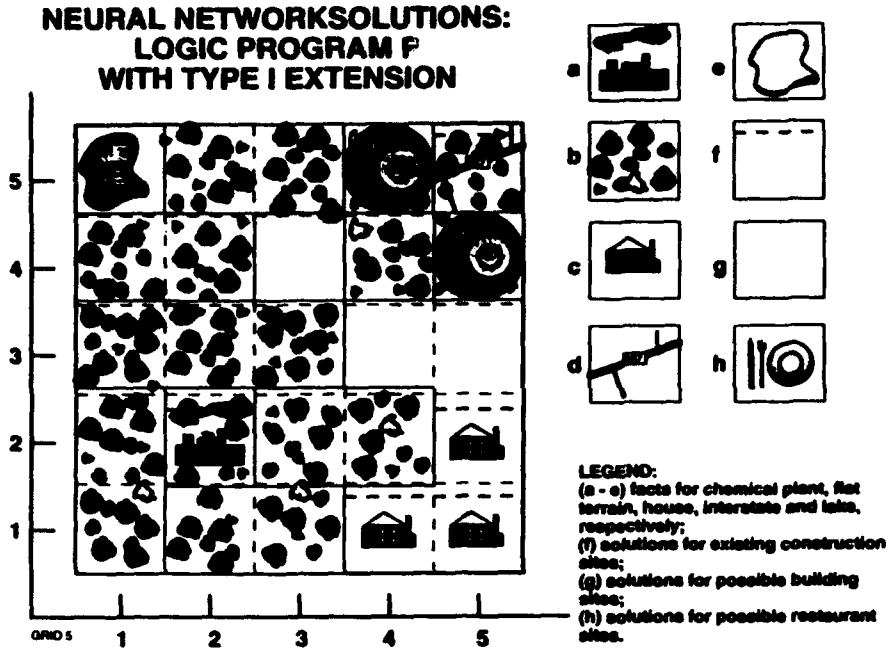

Figure 4: Solutions to Nonlinear Optimization Problem for Logic Program P with TYPE I Extension: (a-e) sites for chemical plant, flat terrain, house, interstate, and lake, respectively; (f) solutions for existing construction sites; (g) solutions for possible building sites; (h) solutions for possible restaurant sites

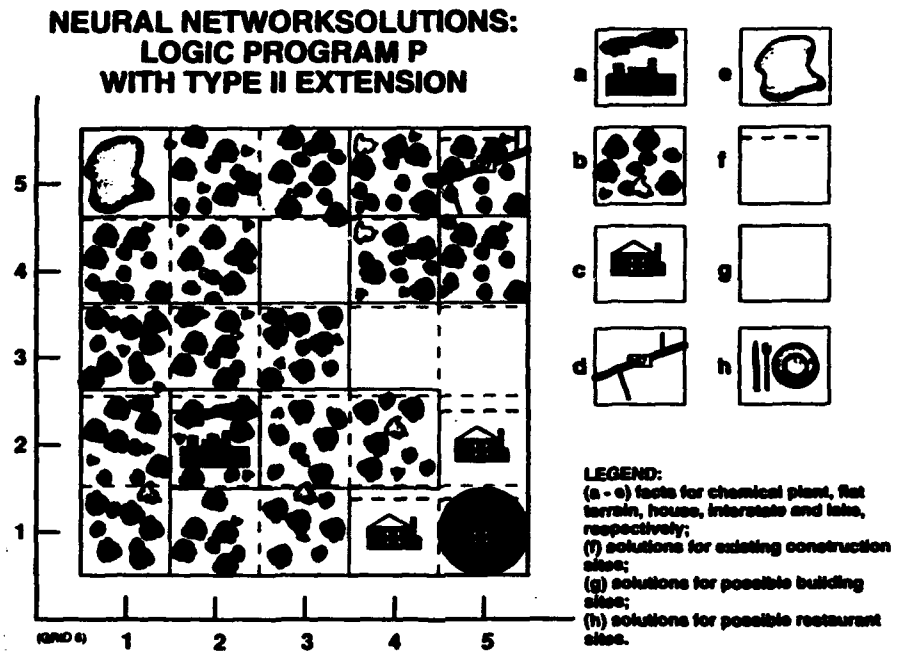

Figure 5: Solutions to Nonlinear Optimization Problems for Logic Program P with TYPE II Extension: (a-e) sites for chemical plant, flat terrain, house, interstate, and lake, respectively; (f) solutions for existing construction sites; (g) solutions for possible building sites; (h) solutions for best house site

In the experiments conducted on the network for the third optimization problem, Logic Program P with TYPE II extension, convergence was again obtained in under 10,000 iterations. The best house, i.e. the one furthest away from the chemical plant, was found to be the house at site (5,1). The solutions for this problem are illustrated in Figure 5 above.

In our experiments, parameters and initial values had to be set appropriately to prevent instability (oscillation or lack of convergence) and convergence to local minima. All variables in our experiments that were not associated with facts or with predicates that did not appear at the head of non-empty clauses were initialized to 0.5. Typical values for $\lambda(0)$ were 0.01, for Δt were 0.00001, and for c_λ were 0.001. The network first gravitated toward the unconstrained solution and then, as a result of the time-varying penalty parameter, converged toward a feasible solution.

5. CONCLUSIONS

This research has provided a design methodology for executing queries to hybrid knowledge bases with extended logic programs containing embedded geometric constraints and embedded optimization problems. A particularly innovative aspect of our approach is the encoding of logical information with geometric constraints as an energy minimization problem that can be solved with recurrent neural networks. The feasibility of the approach has been demonstrated on a small, but realistic, example. The conclusion reached is that this method has the potential to compute fast and efficient solutions to logical deployment problems and that it should be the subject of more investigation in order to realize its full potential in real-world applications.

Future research directions include:

- Investigation of Synergistic Algorithms.
  Synergistic algorithms that combine the strengths of traditional mathematical programming techniques, neural network models, and genetic algorithms need to be investigated in order to build fast, robust, and efficient systems for solving logical deployment problems. Traditional mathematical programming techniques provide a well-understood foundation for solving, in particular, linear and mixed integer programming problems. Neural networks provide a fast and efficient architecture for local search and for computing feasible solutions to nonlinear constraint problems. Genetic algorithms perform efficient global search but, when the objective function is a time-varying function which asymptotically resembles the penalty function, the mating of two distinct feasible solutions may not result in a feasible solution[13].

- Extension to Full-fledged Hybrid Knowledge Bases.
  Methods need to be developed to process queries to logic programs which handle classical and non-monotonic negation, reasoning about time, and reasoning about uncertainty. The results of our present study and the success of neural network methodology in handling reasoning about time and reasoning about uncertainty augur well for future investigations in neural network methodology for implementation paradigms for full-fledged hybrid knowledge bases.

6. ACKNOWLEDGMENTS

This work was supported by the U. S. Army Topographic Engineering Center under the auspices of the U. S. Army Research Office Scientific Services Program administered by Battelle (Delivery Orders 436 and 532, Contract No. DAAL03-91-C-0034). Subrahmanian was supported by the Army Research Office under grant number DAAL03-92-G-0225 and Parikh was supported in part by National Science Foundation grant IRI-9108638. It is a pleasure to acknowledge Barbara Jayne and Vanessa Zalegowski for illustrations and slides, and Cathy Wiley, research librarian at the Navy Center for Applied Research in Artificial Intelligence, for help with references and reprints. This paper could not easily have been completed without them.

7. REFERENCES

1. J. Lu, A. Nerode, J. Remmel and V.S. Subrahmanian, "Outline of a Theory of Hybrid Knowledge Bases", 1992 (submitted for publication).

2. R. E. Jeroslow, "Computation-oriented reductions of predicate to propositional logic", *Decision Support Systems*, Vol 4, pp. 183-187, 1988.

3. C. Bell, A. Nerode, R. Ng, and V. S. Subrahmanian, "Computation and implementation of non-monotonic deductive databases", *University of Maryland Technical Report CS-TR-2801*, 1991.

4. J. Jaffar and J.-L. Lassez, "Constraint Logic Programming", *Proc. 1986 ACM Symp. on Principles of Programming Languages*, 1986.

5. C.-K. Looi, "Neural network methods in combinatorial optimization", *Computers Ops. Res.*, Vol. 19, No. 3/4, pp. 191-208, 1992.

6. J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems", *Biolog. Cybern.*, Vol. 52, pp. 141-154, 1985.

7. M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks", *IEEE Trans. Syst. Man. Cybern.*, Vol. 13, PP. 815-826, 1983.

8. J. J. Wang, "On the asymptotic properties of recurrent neural networks for optimization", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 5, No. 4,pp. 581-601, 1991.

9. R. Shonkwiler, R. and K. R. Miller, "Genetic algorithm/neural network synergy for non-linearly constrained optimization problems", *Proc. International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, pp. 248-257, 1992.