SATyrus: A SAT-based Neuro-Symbolic Architecture for Constraint Processing

Priscila M. V. Lima
NCE/Instituto de Matemática
Universidade Federal do Rio de Janeiro, Brazil
priscila@nce.ufrj.br

M. Mariela Morveli-Espinoza, Glaucia C. Pereira and Felipe M. G. França
COPPE – Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro, Brazil
{mme,gpereira,felipe}@cos.ufrj.br

Abstract

This paper introduces SATyrus*, a neuro-symbolic architecture oriented to optimization problem solving via mapping problems specification into sets of pseudo-Boolean constraints.* SATyrus *provides a logical declarative language used to specify and compile a target problem into a particular energy function representing its space state of solutions. The resulting energy function is then mapped into a Higher-order Hopfield network of stochastic neurons in order to find its global minima. The application of* SATyrus *over three illustrative problems are given: (i) graph coloring, (ii) Traveling Salesperson Problem (TSP), and (iii) calculus of the difference between observed and hypothesized distances of two atoms, a sub-problem of the determination of a molecular structure.*

1. Introduction

SATyrus is a novel approach to the specification and solving of optimization problems. In the scope of the SATyrus architecture, a given target problem is specified, using a logical style declarative language, as a set of pseudo-Boolean constraints. Then, this set of constraints is compiled, following a SATISFIABILITY (SAT) -based mapping, into an energy function representing the space state of solutions of the target problem. Among other possible computational intelligence models that could have been adopted (e.g., genetic algorithms, artificial immune systems, etc), higher-order Hopfield networks of stochastic neurons were chosen to map and minimize the resulting energy function. This choice is justified by the linear cost of representing any energy function produced by SATyrus as symmetric neural networks.

Advantages of using logical constraints as a description language for optimization problems are discussed in this paper. Finding a *model* for a logical sentence is a problem that apparently does not involve optimizing a cost function. In *propositional logic* that corresponds to the assertion of truth-values to the propositional symbols that appear in the formula in question, in such a way that the formula as a whole becomes true. Nevertheless, it is possible to construct a function that maps the truth-value of a formula into the set $\{0, 1\}$. Some problems may be better described as a combination of logical and mathematical constraints. A subset of this combination could be seen as a sum of weighted products of boolean variables, pseudo-Boolean constraints [3].

SATyrus' Architecture is explained in the next section, where a neural engine, a constraint specification language and its compiler are introduced. Sections 3, 4 and 5 illustrate the application of SATyrus over, respectively, the Graph Coloring problem, TSP and the calculus of the module of the difference between two n-bit numbers. Section 6 presents our conclusions, draws comparisons with related work and discusses future research unfoldings.

2. SATyrus' Architecture

SATyrus' architecture is composed of a concatenation of three basic modules: (i) compiler; (ii) mapper, and (iii) (neural) engine. A problem specification, written in the customized language described in Subsection 2.3, is fed to the compiler as a set of pseudo-Boolean constraints. The object code produced by the compiler consists of a "pre-energy function", which can be fine-tuned by the user. Such a tuning, achieved through the adjustment of a "penalty scale" (modulating the whole set of constraints), has the purpose of enabling experiments over the convergence behavior of SATyrus' engine, which is described in Subsection 2.2. Be-

Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS'05)
0-7695-2457-5/05 $20.00 © 2005 IEEE

fore digging into details of the neural engine and the constraints specification language, Subsection 2.1 summarizes the artificial neural network (ANN) model adopted.

2.1. Higher-Order Hopfield Networks

For a function to be called an *energy function* it is necessary that its value decreases monotonically until the (or one of the) stable state(s) of the system is reached. The direct consequence of such interpretation is the proof of *convergence to energy minima* of ANNs composed of symmetrically connected (i. e., $w_{ij} = w_{ji}$) McCulloch-Pitts' neurons $(i, j, \ldots)$ acting as energy minimization (EM) systems, i.e., a Hopfield network [5]. The proof required the observation of a constraint: that nodes operate *asynchronously*, i.e., that no two nodes operate at the same time step. This restriction can be weakened to one where asynchronous operation is only required for *neighbouring nodes*, i.e., it is guaranteed that non-neighbouring nodes can operate at the same time and energy will still decrease monotonically [1]. Two nodes i and j are said to be *neighbours* if they are linked by a connection with weight $w_{ij} \neq 0$.

Connections allowing the mutual influence of concurrent activation among three or more neurons are known as multiplicative or *higher-order*, and the number of units pertaining to a connection is called the *arity* of the connection. Only one value (positive or negative) is associated to each higher-order connection and networks containing one or more multiplicative connections are called *higher-order networks*. Notice that higher-order connections are still considered symmetric, i.e., they take part in the activation function of all nodes involved in the connection, and have the same weight value.

A Hopfield network, even of higher-order, is only capable of finding local minima. By incorporating an stochastic component to the neurons behavior, the resulting network can find global minima through a mechanism known as *simulated annealing* [7]. In this way, consider a random variable d_i associated to each binary node $v_i \in V$, V denoting the set of random variables $v_1, v_2, \ldots, v_n$, $n = |V|$. The values of these random variables are taken from a common finite domain $D = \{0, 1\}$, so that v_i represents the state of neuron i and each element of D^n is a possible network state. Each $v_i \in V$ define a set of neighbours $Q(v_i)$ in such a way that a homogenous neighbourhood is obtained, i.e., for any two $v_i, v_j \in V$, if $v_j \in Q(v_i)$, then $v_i \in Q(v_j)$. The result of this incorporation can be described by the following equations:

$$\begin{cases} p(v_i = 1 | v_j = d_j; v_j \in Q(v_i)) = \frac{1}{1+e^{(-net_i)/T}} \\ p(v_i = 0 | v_j = d_j; v_j \in Q(v_i)) = \frac{e^{(-net_i)/T}}{1+e^{-net_i/T}} \end{cases}$$

where $net_i = (\sum w_{ij} v_j(t)) - \theta_i$, θ_i is the threshold of neuron i, and T is the parameter known as *temperature* $(T \geq 0)$.

2.2. SATyrus' Neural Engine

In order to convert SATISFIABILITY (SAT) to energy minimization (EM), consider the following mapping of logical formulae to the set $\{0, 1\}$:

$$\begin{aligned} H(true) &= 1 \\ H(false) &= 0 \\ H(\neg p) &= 1 - H(p) \\ H(p \wedge q) &= H(p) \times H(q) \\ H(p \vee q) &= H(p) + H(q) - H(p \wedge q) \end{aligned}$$

If a logical formula is converted to an equivalent in *Conjunctive Normal Form* (CNF), the result being a conjunction φ of disjunctions φ_i, it is possible to associate energy to $H(\neg\varphi)$. Nevertheless, energy calculated in this way would only have two possible values: *one*, meaning solution not found (if the network has not reached global minimum), and *zero* when a model has been found. Intuitively, it would be better to have more "clues", or degrees of "non-satisfiabililty", on whether the network is close to a solution or not.

Let $\varphi = \wedge_i \varphi_i$ where $\varphi_i = \vee_j p_{ij}$, and p_{ij} is a literal. Therefore $\varphi = \vee_i \varphi_i$ where $\varphi_i = \wedge_j \neg p_{ij}$. Instead of making $E = H(\neg\varphi)$, consider $E = H^*(\neg\varphi) = \sum_i H(\neg\varphi_i)$. So, $E = \sum_i H(\wedge_j \neg p_{ij}) = \sum_i \prod_j H(\neg p_{ij})$, where $H(p)$ will be referred to as p. Informally, E counts the number of clauses that are *not satisfied* by the interpretation represented by the network's state.

An issue to point out is that the resulting network of the above mapping may have higher-order connections, i.e., connections involving more than two neurons. This does not constitute a hindrance as has been demonstrated that, with higher-order connections, Boltzmann Machines still converge to energy minima [4]. Parallel and distributed simulation of network with higher-order connections can be done by substituting each higher-order connection by a completely-connected subgraph. Alternatively, [10] converts the higher-order network to a binarily connected one that preserves the order of energy values of the different network states.

A simple example demonstrates how SAT can be mapped to EM. Let φ be the formula, expressed as a conjunction of clauses:

$$\varphi = (p \vee \neg q) \wedge (p \vee \neg r) \wedge (r).$$

SAT (φ) can be translated to the minimum of the following energy function:

$$\begin{aligned} E &= H(\neg(p \vee \neg q)) + H(\neg(p \vee \neg r)) + H(\neg r) \\ &= H(\neg p \wedge q) + H(\neg p \wedge r) + H(\neg r) \\ &= (1-p) * q + (1-p) * r + (1-r) \\ &= q - pq - pr + 1 \end{aligned}$$

where $H(p) = p$.

2.3. SATyrus' Declarative Language

The compiler takes as input a file containing the set of restrictions specifying the target problem expressed in CNF. It includes: (i) the group of restrictions to which they belong, i.e., either *integrity* or *optimality* group, (ii) the penalty level associated to subsets of constraints of each group, (iii) definitions of constants, data structures and (iv) definition of the goal function. The set of of integrity constraints define the set of acceptable solutions while the set of optimality constraints grade the acceptable solutions according to the goal function. The compiler translates this file into an intermediate representation containing a header for the table of penalty identifiers together with corresponding suggested values, and a record for each term of the energy function. Each record contains the following information: penalty level, weight, connection arity and list of neighboring neurons. The main features of SATyrus' declarative language are:

- Specification of groups of binary neurons, each bearing a different meaning in the problem being described:

```
structures : '{' structures1 '}' ';';
structures1 : structure structures2 ;
structure : VARPROP '(' dimensions ')';
structures2 : //empty
| ';' structures1;
dimensions : INTEGER dimensions1;
dimensions1 : //empty
| ',' INTEGER dimensions1;
```

- Constructs for the association of an identifier to a group of constraints, in order to enable the attribution of a same penalty level to them, and replication constructs to allow for faster definition of similar constraints:

```
constraints : INTEGRITY GROUP constraint
';' constraints1;
constraints1 : //empty
| INTEGRITY GROUP constraint ';'
constraints1;
constraint : type indexinits indexlimits
indexdifs clauses;
type : TYPE ID ':';
indexinits : indexinit indexinits1;
indexinits1 : ';'
| indexinit indexinits1;
indexinit : FORALL '{' INDEX '}'
| EXISTS '{' INDEX '}';
indexlimits : interval intervals1;
intervals1 : //empty
| ',' interval intervals1;
interval : INDEX interval1
| INTEGER '<' equal INDEX '<' equal
limit;
interval1 : '<' equal INDEX '<' equal
limit
| IN '{' list '}';
limit : INTEGER
| INDEX operation;
operation : //empty
| sign INTEGER;
sign : '+'
| '-';
equal : //empty
| '=';
list : INTEGER ',' INTEGER list1 ;
list1 : //empty
| ',' INTEGER list1 ;
indexdifs : ':'
| ';' indexdif indexdifs1;
indexdif : INDEX '!' '=' INDEX;
indexdifs1 : //empty
| ',' indexdif indexdifs1;
```

- Constructs for the specification of propositional constraints:

```
clauses : '(' disjuncts ')';
disjuncts : disjunct disjuncts1;
disjuncts1 : //empty
| AND disjunct disjuncts1;
disjunct : literal
| '(' literal literals ')';
literals : OR literal literals1;
literals1 : //empty
| OR literal literals1;
literal : atom
| NOT atom;
atom : VARPROP '_' '{' indexes '}'
indexes : vindex indexes1;
indexes1 : //empty
| ',' vindex indexes1;
vindex : INDEX vindex1
| INTEGER
```

```
| '+'
| '-';
vindex1 : //empty
| '+' INTEGER
| '-' INTEGER;
```

- Constructs for the specification of the goal function (it is worth mentioning that this constraints may be read from a file):

```
opts : OPTIMAL GROUP optimality ';'
opts1;
opts1 : //empty
| OPTIMAL GROUP optimality ';' opts1;
optimality : constraint ';' file;
file : VARPROP READ FROM ID;
```

- Constructs for the attribution of a penalty level to a group of constraints formerly identified as belonging to a `TYPE ID`:

```
penalties : PENALTY '{' penalties1 '}';
penalties1 : penalty penalties2;
penalties2 : //empty
| ',' penalty penalties2;
penalty : ID IS LEVEL INTEGER;
```

- The general format of a problem specification is:

```
start : structures constraints opts
penalties;
```

3. Graph Coloring as Energy Minimization

Let $G = (V, A)$ be an undirected graph, where V is the graph's vertex set and A the set of G's edges. The Graph Coloring Problem consists of determining the minimum assignment of colours (positive integers) to the vertices such that each vertex has only one colour and no two neighbouring vertices have the same colour. After the mapping phase, the resulting network is mainly composed by a matrix V_{colour} having $n \times n$ binary neurons vc_{ik} and a matrix C_{olour} having $1 \times n$ binary neurons c_k, where i is a vertex in V and k represents the colour associated to vertex i. Addicionally, a matrix $neigh_{ii'}$ is used to indicate the neighbouring relationship between vertices.

Integrity constraints:

(i) Every vertex must have one color assigned to it:
$\forall i, \forall_k | 1 \leq i \leq n, 1 \leq k \leq n : \vee(vc_{ik})$. So, let $\varphi_1 = \wedge_i(\vee_k(vc_{ik}))$.

(ii) Two neighbouring vertices cannot have the same color:
$\forall i, \forall i', \forall_k | 1 \leq i \leq n, 1 \leq i' \leq n, 1 \leq k \leq n, i \neq i' : \neg(neigh_{ii'}) \vee \neg(vc_{ik} \wedge vc_{i'k})$. So, let $\varphi_2 = \wedge_i \wedge_{i'} \wedge_k (\neg(neigh_{ii'}) \vee \neg(vc_{ik} \wedge vc_{i'k}))$.

(iii) A vertex cannot have more than one color:
$\forall i, \forall_k, \forall_{k'} | 1 \leq i \leq n, 1 \leq k \leq n, 1 \leq k' \leq n, k \neq k' : \neg(vc_{ik} \wedge vc_{ik'})$. So, let $\varphi_3 = \wedge_i \wedge_k \wedge_{k'} \neg(vc_{ik} \wedge vc_{ik'})$.

(iv) If a colour k is assigned to a vertex in matrix V_{colour}, then the corresponding unit in matrix C_{olour} must be activated:
$\forall i, \forall k | 1 \leq i \leq n, 1 \leq k \leq n : \neg(vc_{ik}) \vee (c_k)$. So, let $\varphi_4 = \wedge_i \wedge_k (\neg(vc_{ik}) \vee (c_k))$.

Optimality constraints

(v) The number of activated elements in matrix C_{olour}:
$\forall k | 1 \leq k \leq n : c_k$. So, let $\varphi_5 = \wedge_k c_k$.

The multiplicative constants α and β are used to indicate the penalty strength:

$$\begin{cases} \alpha = (n * 1) + h \\ \beta = (3n * \alpha) + h \end{cases}$$

The specification of the Graph Coloring problem, as above, can be represented by the following code in SATyrus' declarative language:

```
num=6;
neigh(num,num);
vc(num,num);
colour(num);
integrity group type alfa:
forall{i,k}; 1<=i<=num,1<=k<=num:
vc[i][k];
integrity group type beta:
forall{i,l,k}; 1<=i<=num,1<=l<=num,
1<=k<=num;i!=l: (not neigh[i][l] or
not vc[i][k]or not vc[l][k]);
integrity group type beta:
forall{i,k,m}; 1<=i<=num,1<=k<=num;
1<=m<=num;k!=m: (not vc[i][k] or not
vc[i][m]);
integrity group type beta:
forall{i,k}; 1<=i<=num,1<=k<=num:
(not vc[i][k] or colour[k]);
optimality group type costo:
forall{k}; 1<=k<=num: colour[k];
PENALTY {
beta is level 2;
alfa is level 1;
costo is level 0;}
```

Figure 1 shows execution results of the higher-order Hopfield network of stochastic neurons, defined by the above specification, for a six nodes Graph Coloring problem. Geometrical cooling (0.99) was applied.

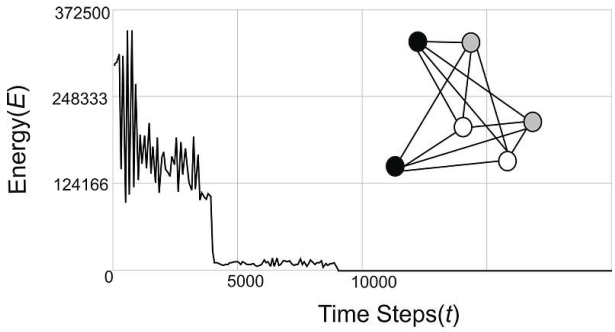

Figure 1. Energy behavior and global minimum found, after 9,967 steps, in a six nodes Graph Coloring problem.

4. TSP as Energy Minimization

Let $G = (V, A)$ be an undirected graph, where V is the graph's vertex set and A the set of G's edges. Associating each vertex $i \in V$ to a city and each edge $(i, j) \in A$ to a path between i and j, if $|V| = n \geq 3$ and $dist_{ij}$ is the cost associated to the edge $(i, j) \in A$ where $\{i, j\} \in V$, then, the Traveling Salesperson Problem (TSP) consists on determining G's Hamiltonian cycle of minimum cost. The first *ad hoc* approach to TSP as a set of mathematical constraints solvable by an ANN was introduced in [6]. In order to enable the tour to end at an initial city a, a twin name a' is given so that it will be clamped as the least city of the tour (with all traveling costs repeated), in the same way that a is clamped as the first city of the tour. In this way, a problem with m cities has to use an augmented $n \times n$ matrix, where $n = m + 1$, so that all conditions may be applied to a round tour. The resulting network, composed of an $n \times n$ matrix of binary neurons v_{ij}, where i represents a city in V and j represents the position of i in the tour, has its behavior specified by the following constraints:

Integrity constraints:

(i) All n cities must take part in the tour:
$\forall i, \forall j | 1 \leq i \leq n, 1 \leq j \leq n : \vee_j (v_{ij})$. So, let $\varphi_6 = \wedge_i (\vee_j (v_{ij}))$.

(ii) Two cities cannot occupy the same position in the tour:
$\forall i, \forall j, \forall i' | 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq i' \leq n, i \neq i' : \neg(v_{ij} \wedge v_{i'j})$. So, let $\varphi_7 = \wedge_i \wedge_{i' \neq i} \wedge_j \neg(v_{ij} \wedge v_{i'j})$.

(iii) A city cannot occupy more than one position in the tour:
$\forall i, \forall j, \forall j' | 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq j' \leq n, j \neq j' : \neg(v_{ij} \wedge v_{ij'})$. So, let $\varphi_8 = \wedge_i \wedge_j \wedge_{j' \neq j} \neg(v_{ij} \wedge v_{ij'})$.

Optimality constraints:

(iv) The cost between two consecutive cities in the tour:
$\forall i, \forall j, \forall i' | 1 \leq i \leq n, 1 \leq j \leq n-1, 1 \leq i' \leq n, i \neq i' : dist_{ii'}(v_{ij} \wedge v_{i'(j+1)})$. So, let $\varphi_9 = \vee_i \vee_{i' \neq i} \vee_{j<n} dist_{ii'}(v_{ij} \wedge v_{i'(j+1)})$.

The multiplicative constants α and β are used to indicate the penalty strength of the integrity constraints while $dist$ is applied to the optimality constraints:

$$\begin{cases} dist &= \max\{dist_{ij}\} \\ \alpha &= ((n^3 - 2n^2 + n) * dist) + h \\ \beta &= ((n^2 + 1) * \alpha) + h \end{cases}$$

The specification of TSP, as above, can be represented by the following code in SATyrus' declarative language:

```
num=4;
pos(num,num);
dist (num,num);
dist read from tsp.txt;
integrity group type alpha:
foralli,j; 1<=i<=num,1<=j<=num: pos[i][j];
integrity group type beta:
foralli,j,k;
1<=i<=num,1<=j<=num,1<=k<=num;i!=k: (not
pos[i][j] or not pos[k][j]);
integrity group type beta:
foralli,j,l;
1<=i<=num,1<=j<=num,1<=l<=num;j!=l: (not
pos[i][j] or not pos[i][l]);
optimality group type dist:
foralli,j,k;
1<=i<=num,1<=j<=num,1<=k<=num;i!=k:
dist[i][k](pos[i][j] and pos[k][j-1]);
PENALTY {
beta is level 2;
alfa is level 1;
dist is level 0;}
```

Figure 2 shows execution results of the higher-order Hopfield network of stochastic neurons, defined by the above specification, for the TSP over a six cities problem.

5. Module of N-Bit Subtraction as Energy Minimization

The methodology adopted in SATyrus requires that a problem be specified as a set of propositional constraints in CNF. However, it is possible that one feels more comfortable with defining the solution of a problem via the statement of a set of minterms. This is equivalent to a formula

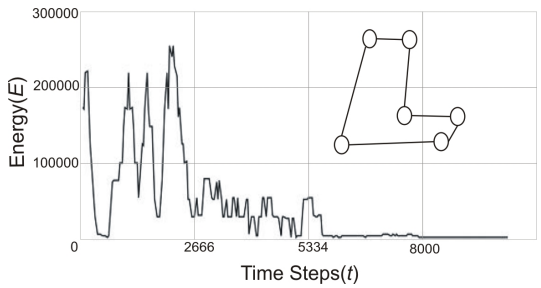

Figure 2. Energy behavior and global minimum found, after 7,960 steps, in a six cities TSP problem.

in *Disjunctive Normal Form* (DNF), that is, a disjunction of conjunctions. That does not constitute a problem, because a formula in DNF can be easily converted to CNF. In order to illustrate this possibility, the calculation of the module of the difference between two n-bit numbers, $n = 3$, was specified using SATyrus' declarative language (not shown here) in the same way as in the cases of Graph Coloring and TSP. This operation is interesting since it can define the difference between observed and hypothesized distances of two atoms, part of the determination of a molecular structure, a problem to be tackled in the near future. Figure 3 presents execution results of the higher-order Hopfield network of stochastic neurons produced by the corresponding specification. Geometrical cooling (0.99) was applied.

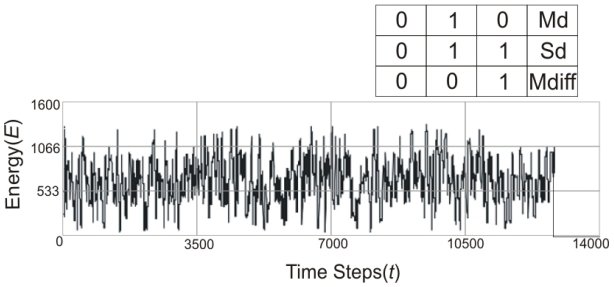

Figure 3. Energy behavior and global minimum found, after 13,249 steps, in the calculus $|Md - Sd| = Mdiff$, $Md = 2$, $Sd = 3$ (3-bit numbers).

6. Conclusion

SATyrus is a novel paradigm for the specification of problems as sets of constraints. Other constraint specification languages, such as Z notation [11] and MathSAT [2], differ from this work on (i) the use of a neural engine and on (ii) the possibility of combining different sets of constraints, representing different problems, in order to specify a new target problem, as recently demonstrated [9]. Moreover, our approach profits from the intermediate definition of an energy function, which can be minimized by any available solver, not only higher-order Hopfield networks of stochastic neurons, as considered in this work. Among the most interesting investigations for future work, we intend to develop an integration of first-order logic inferencing [8] with pseudo-boolean constraints as an alternative and natural way of processing constraint logic programming.

Acknowledgements

This work was partially supported by CNPq and CAPES, Brazilian research agencies.

References

[1] V. Barbosa and P. Lima. On the distributed parallel simulation of hopfield's neural networks. *Software-Practice and Experience*, 20(10):967–983, 1990.

[2] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani. Mathsat: Tight integration of sat and mathematical decision procedures. *Journal of Automated Reasoning*, 2005. Special Issue on SAT. To appear.

[3] H. Dixon, M. Ginsberg, and A. Parkes. Generalizing boolean satisfiability 1: Background and survey of existing work. *Journal of Artificial Intelligence Research*, 21:193–243, 2004.

[4] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[5] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences USA*, 79:2554–2558, 1982.

[6] J. Hopfield and D. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[7] S. Kirkpatrick, C. GellatJr., and M. Vecchi. Optimization via simulated annealing. *Science*, 220:671–680, 1983.

[8] P. Lima. *Resolution-Based Inference on Artificial Neural Networks*. PhD thesis, Department of Computing, Imperial College of Science, Technology and Medicine, 2000.

[9] P. Lima, G. Pereira, M. Morveli-Espinoza, and F. França. Mapping and combining combinatorial problems into energy landscapes via pseudo-boolean constraints. *Lecture Notes on Computer Science*, 2005. Proc. of BVAI '05. To appear.

[10] G. Pinkas. *Logical Inference in Symmetric Neural Networks*. PhD thesis, Sever Institute of Technology, Washington University, 1992.

[11] M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, International Series in Computer Science, 1992.