An Evolutionary Approach to the School Timetabling Problem

BARBARA KOROUŠIĆ-SELJAK
Computer Systems Department
"Jožef Stefan" Institute
Jamova 39, 1000 Ljubljana
SLOVENIA

Abstract: We present a general heuristic technique that solves a timetabling problem, such as school timetabling. It is based on a Hopfield-type neural network, whose complexity was significantly reduced by incorporating a genetic algorithm into a very first stage of the timetabling process. The goal of the GA was to initialize the network so that the number of neurons and their connections decreased, and as a consequence the computation speeded up. By using this technique, we were able to produce school timetables subject to both hard and soft constraints in a reasonable polynomial time. The technique proved to be an efficient method for solving a NP-complete combinatorial optimization problem.

Keywords: School timetabling, combinatorial optimization, Hopfield-type neural network, Liapunov function, genetic algorithm.

1 Introduction

Timetabling is a problem of assigning time slots to a set of events to satisfy several constraints. It is a NP-complete multi-objective combinatorial optimization problem, which requires considerable computational effort for its solution [1]. There are several versions of this problem, such as school timetabling, university-course timetabling, examination timetabling, employee timetabling, etc.

The literature on automated timetabling includes several surveys [2], where various formulations of the problem, as well as techniques and algorithms used for its solution are discussed. During the last few decades, many papers related to automated timetabling have been published. In addition, many applications have been developed and employed with good success to handle particular instances of the general timetabling problem.

In this paper we describe how the school timetabling problem has been solved by using an evolutionary approach. As the technique extends easily to any timetabling problem, it could be useful in general.

1.1 The School Timetabling Problem

The school timetabling problem can be briefly stated as follows: *For a certain school with T teachers, C classes and R class-rooms, it is required to schedule TR teacher-class-room triplets within a time limit of P time slots producing a feasible schedule.*

Constraints for this problem may be hard (i.e. must be satisfied) or soft (i.e. should be satisfied as far as possible). Hard constraints may include:

- Binary constraints (involve two or more events that must not overlap in time), such as:
  - A teacher cannot teach two or more classes at a time.
  - At most one teacher can teach a class at a time.
  - A classroom cannot be shared by two or more classes at a time.
- Unary constraints (involve just one event), such as:
  - Some subjects may be excluded from taking place in a given classroom, must not start at a given time, or cannot be assigned to a certain class or a teacher.
  - Some subjects must take place in a given (dedicated) classroom, must start at a given time, or must be assigned to a certain class or a teacher.
- Capacity constraints, such as:
  - A class cannot be assigned to a particular classroom unless the capacity of the classroom is greater than or equal to the number of pupils in the class.

Soft constraints may include:

- Didactic goals (e.g. balancing or spreading out the subjects over the period),
- Personal goals (e.g. keeping a certain time slot free for a given teacher),

- ❖ Organizatorial goals (e.g. keeping some time slots or classrooms fixed for a certain class or a teacher).

A feasible schedule is one that satisfies all hard constraints of the problem, and minimizes the weighted sum of costs (penalties) associated with its soft constraints.

1.2 Timetabling Techniques

It has been proved that the general timetabling problem has no exact solution, which is polynomial-time in the length of the input [3]. However, it is possible to apply an efficient heuristic algorithm that runs in polynomial time and usually, but not always, outputs an optimal solution that is guaranteed to be close to the global solution.

Most of the early timetabling techniques were based on direct heuristics. The idea was to simulate the human way of solving the problem. Later on, researchers proposed general heuristic techniques, such as simulated annealing, tabu search, constraint-logic programming and genetic algorithms [4]. The problem has also been tackled by a reduction to a well-studied problem (like the graph colouring [5]).

Neural networks [6] are alternative general heuristic techniques for solving prediction, classification and pattern recognition problems. In a recent survey on neural networks, Smith [7] clarified an extra potential of neural networks for solving combinatorial optimization problems. However, the idea of using neural networks to provide solutions to difficult NP-hard optimization problems originated in 1985 when Hopfield and Tank demonstrated that the Travelling Salesman Problem [8] could be solved using a Hopfield-type neural network (HTNN) [9].

In this paper, we present a HTNN that results most of the time in optimal solutions to the school timetabling problem, which are reasonably well in terms of solution quality and time performance. The rest of the paper is organized as follows: in Section 2 we provide a brief description a HTNN; in Section 3 we present a timetabling technique that is based on the HTNN; in Section 4 we introduce the GA as an initial routine; in Section 5 we evaluate the technique through examples; and in Section 6 we list our conclusions.

2 A Hopfield-type Neural Network

A HTNN can be described as a biologically inspired mathematical tool for solving some of the combinatorial optimization problems. The great advantage is its ability to generate a solution to the problem without necessity of training iterations.

A HTNN belongs to the class of *recurrent* neural networks. It comprises a fully interconnected system of n single layered computational units (or neurons). A strength (weight) of the connection between neuron i and neuron j is determined by $\omega_{i,j}$, which may be positive or negative depending on whether the neurons act in an excitatory or an inhibitory manner. An internal state of each neuron (u_i) is equivalent to the weighted sum of the external states of all connecting neurons. V_i gives an external state of neuron i with the range [0,1]. It is bounded by the asymptotes of a monotone-increasing function $g(u_i)$, given by the following dynamical equations:

$$\frac{du_i}{dt} = \sum_{j=1}^{n} \omega_{i,j} V_j + I_i - \frac{u_i}{\tau}, \qquad (1)$$

$$V_i = g(u_i),$$

where I_i is an external input (or bias current) to neuron i and $-u_i/\tau$ is a passive decay. In the absence of the external input and the inputs from other neurons, this term causes u_i to decay toward zero at a rate proportional to τ. A typical choice of the input-output relation g is a smooth sigmoid with asymptotes 0 and 1. This is a bounded differentiable real function that is defined for all real input values, and has a positive derivative everywhere. A frequent choice for $g(u_i)$ is

$$g(u_i) = \frac{1 + th(\lambda u_i)}{2}, \qquad (2)$$

where *th* denotes a hyperbolic function that is related to the hyperbola, and is analogous to the trigonometric function tangent. As long as g is nondecreasing, it meets the Cohen-Grossberg requirements for stability [10]. Thus, if the external inputs are maintained at a constant value, a network of neurons modeled by (1) will eventually equilibrate, regardless of the initial state.

The rapid computation power and speed of a HTNN can be obtained through a simple hardware implementation [11,12]. Yet, a problem in the silicon implementation is a vast number of connections (in a network of n neurons, there are n^2 connections). However, there are several choices of architectures, which could efficiently be used to achieve full network's parallelism. A systolic ring architecture is such an example that requires only n^2 neurons operating in parallel [13]. In addition, recent progress in the area of FPGAs has enabled speed advantages of hardware implementation to be simulated on a digital computer using a reconfigurable hardware [14].

3 A Timetabling Technique

Using the method proposed by Hopfield and Tank [9], the following steps have to be performed to obtain feasible school timetables:

1. An objective function has to be defined for the problem. By minimization of this function, an optimal solution that corresponds to a feasible school timetable can be obtained. The objective function has to be specified in terms of a Liapunov function [15]:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\omega_{i,j}V_iV_j - \sum_i I_iV_i \qquad (3)$$

   because it guarantees convergence of a corresponding HTNN to a stable state.
2. Parameters of the corresponding HTNN, such as connection weights and external inputs, have to be defined by considering both hard and soft constraints of the problem. These parameters determine the network's energy function.
3. Initial conditions and updating rules have to be defined to start and run the network.
4. A method for interpreting the final state of the network as a problem's solution has to be established.

3.1 The Objective Function

The objective function, which satisfies a set of *m* hard and soft constraints of the school timetabling problem, is expressed in the terms of (3):

$$F = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\sum_{c=1}^{m}\omega_{i,j}^{c}V_iV_j - \sum_{i=1}^{n}\sum_{c=1}^{m}I_i^cV_i, \qquad (4)$$

Hard constraints can be expressed mathematically as equalities or inequalities, such as:

$$\begin{aligned}
&\forall i\in\{1,...,T\},\forall j\in\{1,...,C\}:\sum_{t=1}^{TR}\sum_{p=1}^{P}t_{i,t}c_{j,t}V_{t,p} = n_{<i,j>},\\
&\forall p\in\{1,...,P\},\forall i\in\{1,...,T\}:\sum_{t=1}^{TR}V_{t,p}t_{i,t}\leq 1,\\
&\forall p\in\{1,...,P\},\forall i\in\{1,...,C\}:\sum_{t=1}^{TR}V_{t,p}c_{j,t}\leq 1,\\
&\forall p\in\{1,...,P\},\forall k\in\{1,...,R\}:\sum_{t=1}^{TR}V_{t,p}r_{k,t}\leq n_k.
\end{aligned} \qquad (5)$$

The first equation of (5) ensures that a teacher *i* and a class *j* meet exactly $n_{<i,j>}$ times in the final schedule, $N_{T\times C}$ being a non-negative integer matrix. However, there exists possibly an optimal solution if and only if $\sum_{i=1}^{T}\sum_{j=1}^{C}n_{<i,j>}\leq T\cdot P$. Next three equations of (5) prevent the teacher, the class and the classroom conflicts, respectively (i.e. binary constraints). We are given three binary matrices $T_{T\times TR}$, $C_{C\times TR}$ and $R_{R\times TR}$, such that

$$\begin{aligned}
&t_{i,t} = \begin{cases}1, if\ \cdot\ i\in t\\ 0, otherwise\end{cases}, c_{j,t} = \begin{cases}1, if\ \cdot\ j\in t\\ 0, otherwise\end{cases},\\
&r_{k,t} = \begin{cases}1, if\ \cdot\ k\in t\\ 0, otherwise\end{cases}.
\end{aligned} \qquad (6)$$

N_R is a non-negative vector, where $n_k\leq R$ defines the number of classrooms in which a subject *k* may be given (i.e. the capacity constraint). If some unary constraint is required, the corresponding equation of (5) expressed in terms of inequality should be rewritten in terms of equality, such as:

$$\sum_{t=1}^{TR}V_{t,x}t_{y,t} = 1, \qquad (7)$$

where a teacher *y* is required to give a subject in a period *x*.

By reduction to the 3-SAT problem [8] the school timetabling problem, which considers the hard (binary, capacity and unary) constraints, was proved to be NP-complete.

Soft constraints are defined in a similar way as the hard constraints, but are beyond the scope of this paper.

3.2 The Parameters and the Energy Function

The network's energy function, which considers the above-defined hard constraints of the problem, is expressed with penalty terms:

$$\begin{aligned}
E = &-\frac{1}{2}\sum_{t=1}^{TR'}\sum_{p=1}^{P}\sum_{t'=1}^{TR'}\sum_{\substack{p'=1 \\ (t,p)\neq(t',p')}}^{P}\omega_{(t,p),(t',p')}V_{t,p}V_{t',p'}\\
&-\sum_{t=1}^{TR'}\sum_{p=1}^{P}I_{t,p}V_{t,p},
\end{aligned} \qquad (8)$$

where $TR' = TR + T + C + \sum_{k=1}^{R}n_k$.

The network consists of $TR\cdot P$ "representation" neurons plus $(T + C + \sum_{k=1}^{R}n_k)\cdot P$ "slack" neurons. The "slack" neurons are required to inforce constraints expressed as inequalities [10]. As a consequence, also the binary matrices $T_{T\times TR}$, $C_{C\times TR}$ and $R_{R\times TR}$ need to be extended into $T_{T\times TR+T}$, $C_{C\times TR+T+C}$ and $R_{R\times TR+T+C+}\sum_{k=1}^{R}n_k$, respectively, such that

$$\forall i\in\{1,...,T\},\forall y\in\{1,...T\}:t_{i,TR+y} = \begin{cases}1, if\ \cdot\ i = y\\ 0, otherwise\end{cases}$$

$$\forall j \in \{1,...,C\}, \forall y \in \{1,...T\}: t_{j,TR+y} = 0,$$
$$\forall j \in \{1,...,C\}, \forall y \in \{T+1,...,T+C\}: \quad (9)$$
$$c_{j,TR+y} = \begin{cases} 1, if \cdot j = y \\ 0, otherwise \end{cases}$$
$$\forall k \in \{1,...,R\}, \forall y \in \left\{1,...T + C + \sum_{i=1}^{k-1} n_i\right\}: r_{k,TR+y} = 0,$$
$$\forall k \in \{1,...,R\},$$
$$\forall y \in \left\{T + C + \sum_{i=1}^{k-1} n_i + 1,...,T + C + \sum_{i=1}^{k} n_i\right\}: r_{k,TR+y} = 1,$$
$$\forall k \in \{1,...,R\},$$
$$\forall y \in \left\{T + C + \sum_{i=1}^{k} n_i + 1,...,T + C + \sum_{i=1}^{R} n_i\right\}: r_{k,TR+y} = 0.$$

The network's energy function of (8) is made equivalent to the problem's objective function of (4) if the connection weights and the external inputs take the following values:

$$\omega_{(t,p),(t',p')} = \sum_{i=1}^{T}\sum_{j=1}^{C} \omega^{1,<i,j>}_{(t,p),(t',p')}$$
$$+ \begin{cases} 0, if \cdot p \neq p' \\ \sum_{i=1}^{T} \omega^{2,i}_{(t,p),(t',p)} + \sum_{j=1}^{C} \omega^{3,j}_{(t,p),(t',p)} + \sum_{k=1}^{R} \omega^{4,k}_{(t,p),(t',p)}, otherwise \end{cases}$$
$$I_{t,p} = \sum_{i=1}^{T}\sum_{j=1}^{P} I^{1,<i,j>}_{t,p} + \sum_{i=1}^{T} I^{2,i}_{t,p} + \sum_{j=1}^{C} I^{3,j}_{t,p} + \sum_{k=1}^{R} I^{4,k}_{t,p},$$
$$\omega^{1,<i,j>}_{(t,p),(t',p')} = \begin{cases} -2t_{i,t}c_{j,t}t_{i,t'}c_{j,t'}, if \cdot (t,p) \neq (t',p') \\ 0, otherwise \end{cases},$$
$$I^{1,<i,j>}_{t,p} = (2n_{<i,j>} - 1)t_{i,t}c_{j,t},$$

$$\omega^{2,i}_{(t,p),(t',p)} = \begin{cases} -2t_{i,t}t_{i,t'}, if \cdot t \neq t' \\ 0, t = t' \end{cases}, \quad (10)$$
$$I^{2,i}_{t,p} = t_{i,t},$$
$$\omega^{3,j}_{(t,p),(t',p)} = \begin{cases} -2c_{j,t}c_{j,t'}, if \cdot t \neq t' \\ 0, t = t' \end{cases},$$
$$I^{3,j}_{t,p} = c_{j,t},$$
$$\omega^{4,k}_{(t,p),(t',p)} = \begin{cases} -2r_{k,t}r_{k,t'}, if \cdot t \neq t' \\ 0, t = t' \end{cases},$$
$$I^{4,k}_{t,p} = (2n_k - 1)r_{k,t}.$$

The values of the connection weights and the external inputs are saved in a matrix $\Omega_{TR'\cdot P \times TR'\cdot P}$ and a vector $I_{TR'\cdot P}$, respectively.

3.3 The Initial Conditions and the Updating Rules

The HTNN that is defined by the energy function of (8) can start its computation when an internal states of each neuron *i* is initiated, such as:

$$u_i = u_{init} + R,$$
$$u_{init} = -\frac{ath(2\frac{1}{N} - 1)}{\lambda}, N = \sum_{i=1}^{T}\sum_{j=1}^{C} n_{<i,j>}, \quad (11)$$

where *R* is a random value that is distributed linearly on $\pm 0.1u_{init}$. This is required to prevent the network become trapped in an unstable equilibrium.

The computation of the above-defined network can be performed by executing the following updating rules:

$$u^{(t+1)}_{t,p} = u^{(t)}_{t,p} + \Delta t(\sum_{p=1}^{P}\sum_{p'=1}^{P} \omega_{(t,p),(t',p')})V_{t',p'} + I_{t,p} - u^{(t)}_{t,p},$$
$$V_{t,p} = g(u_{t,p}) = \frac{1 + th(\lambda u_{t,p})}{2}, \quad (12)$$

3.4 The Interpretation Method

A school timetable can be represented as a permutation matrix $S_{TR\times P}$, where $s_{t,p}$ corresponds to $V_{t,p}$. A matrix row *t* corresponds to the t^{th} triplet teacher-class-classroom (<*i*,*j*,*k*>), while a column *p* corresponds to the p^{th} period. The final schedule generated by the network is interpreted by replacing each matrix entry with either 1 or 0, depending on whether the corresponding neuron state is 'high' or 'low'. If $s_{t,p}$=1, a teacher *i* can teach a class *j* in a classroom *k* in a period *p*.

4 The Genetic Algorithm

We applied a genetic algorithm (GA) [4] to reduce the number of "slack" neurons. Following the assumption that a classroom with the adequate capacity can be selected for a pair teacher-class before the network's run-time, the last inequality constraint of (5) could be simplified, such as

$$\forall p \in \{1,...,P\}, \forall k \in \{1,...,R\}: \sum_{k=1}^{R} V_{t,p} r_{k,t} \leq 1, \quad (13)$$

The GA codes parameters of the problem's search space as finite-length strings over some finite alphabet. It works with a coding of the parameter set, not the parameters themselves. The algorithm employs an initial population of strings, which evolve into the next generation under the control of probabilistic transition rules—known as randomized genetic operators—such as selection, crossover and

mutation. The fitness function evaluates the quality of solutions coded as strings. This information is then used to perform an effective search for better solutions. There is no need of other auxiliary knowledge. The GA tends to take advantage of the fittest solutions by giving them greater weight, and concentrating the search in the regions of the search space that show likely improvement.

The GA is different from traditional techniques because of its intrinsic parallelism (in evaluation function, selections) that allows working from a broad database of solutions in the search space simultaneously, climbing many peaks in parallel. Thus, the risk of converging on a local optimum is low. The random decisions made in the GA can be modeled using Markov chain analysis to show that each finite GA will always converge to its global optimum region [16].

In spite of its simplicity, the GA has proved to be an efficient method for solving various optimization and classification problems, in areas ranging from economics and game-theory to control-system design.

4.1 Encoding

In our case, the parameters (i.e. the indexes of classrooms) were coded as strings over the alphabet $\mathbb{N}^+$ of non-negative integer values. Using a symbolic presentation of a string with $T \cdot C$ parameters (i.e. teacher-class pairs) gives:

$$S = s_{<1,1>} \ldots s_{<i,j>} \ldots s_{<T,C>}$$

4.2 Fitness Evaluation

Following the genetic operators the new population had to be evaluated. Here, each string *s* (solution candidate) of the population was decoded into a set of classroom indexes. Its fitness was defined by the following quantity

$$fitness_S = \sum_{i=1}^{T}\sum_{j=1}^{C} f_{<i,j>} \cdot \max_{k \in \{1,\ldots,R\}} \sum_{i=1}^{T}\sum_{j=1}^{C} n_{<i,j>} f_{<i,j,k>}$$

$$f_{<i,j>} = \begin{cases} 1, if \cdot s_{<i,j>} \cdot is \cdot dedicated \cdot to \cdot <i,j> \\ 0, otherwise \end{cases},$$

$$f_{<i,j,k>} = \begin{cases} 1, if \cdot s_{<i,j>} = k \\ 0, otherwise \end{cases}. \quad (14)$$

This information was used to perform an effective search for feasible solutions, i.e. strings with the shortest schedule length. A solution was considered to be feasible if and only if its fitness value was less than or equal to *P*. Feasible solutions including more appropriate classrooms for given subjects (teacher-class pairs) were considered to be better solutions.

4.3 The Modifications of the Network

By incorporating the GA into the timetabling technique, several modifications are required, such as

$$E = -\frac{1}{2}\sum_{t=1}^{TR'}\sum_{p=1}^{P}\sum_{t'=1}^{TR'}\sum_{\substack{p'=1 \\ (t,p)\neq(t',p')}}^{P} \omega_{(t,p),(t',p')} V_{t,p} V_{t',p'}$$

$$-\sum_{t=1}^{TR'}\sum_{p=1}^{P} I_{t,p} V_{t,p},$$

where $TR' = TR + T + C + R$.

$$\omega_{(t,p),(t',p')} = \sum_{i=1}^{T}\sum_{j=1}^{C} \omega^{1,<i,j>}_{(t,p),(t',p')}$$

$$+ \begin{cases} 0, if \cdot p \neq p' \\ \sum_{i=1}^{T} \omega^{2,i}_{(t,p),(t',p)} + \sum_{j=1}^{C} \omega^{3,j}_{(t,p),(t',p)} + \sum_{k=1}^{R} \omega^{4,k}_{(t,p),(t',p)}, \\ otherwise \end{cases} \quad (15)$$

$$\omega^{4,k}_{(t,p),(t',p)} = \begin{cases} -2 r_{k,t} r_{k,t'}, if \cdot t \neq t' \\ 0, t = t' \end{cases},$$

$$I_{t,p} = \sum_{i=1}^{T}\sum_{j=1}^{P} I^{1,<i,j>}_{t,p} + \sum_{i=1}^{T} I^{2,i}_{t,p} + \sum_{j=1}^{C} I^{3,j}_{t,p} + \sum_{k=1}^{R} I^{4,k}_{t,p}, \quad (16)$$

$$I^{4,k}_{t,p} = r_{k,t}.$$

The matrix *R* reduces into $R_{R \times TR+T+C+R}$, such that

$$\forall k \in \{1,\ldots,R\}, \forall y \in \{1,\ldots T+C\}: r_{k,TR+y} = 0,$$

$$\forall k \in \{1,\ldots,R\}, \forall y \in \{T+C+1,\ldots,T+C+R\}: \quad (17)$$

$$r_{k,TR+y} = \begin{cases} 1, if \cdot k = y \\ 0, otherwise \end{cases}.$$

As *TR'* also reduces, dimensions of the matrix $\Omega_{TR' \cdot P \times TR' \cdot P}$ and the vector $I_{TR' \cdot P}$ become suitably smaller. In addition, the number of triplets *TR* divides for factor *R*.

5 The Evaluation of the Technique

In order to evaluate the timetabling technique, we selected two specific school timetabling problems, one simplified (artificially constructed) and one taken from real life (Table 1). The real-life problem had to satisfy the hard (binary, unary and capacity) constraints as well as several soft constraints.

We developed software by using the Borland Delphi programming tool to set the network's parameters (i.e. connection weights and external inputs) and select a set of classrooms for each pair

teacher-class. This software is an implementation of the GA.

Otherwise, we applied the Wolfram Research Mathematica [17] prototyping tool for experimentation with the neural network. The results presented in Table 1 demonstrate that the complexity of the neural network increases with the number of teachers, classes and classrooms.

233 MHz Pentium with 64MB RAM	*T|C|R|P TR (TR')*	Parameters setting	Neurons	Iterations	Feasible solutions
Simplified problem	2|2|2|2 4(10)	5 secs.	70	1000 60	90 %
Real-life problem	7|2|9|21 12(30)	180 secs.	630	5000 900	75 %

Table 1. Results for two school timetabling examples.

We tried to improve the results by applying two modifications of the HTNN:

1. During the state evolution, we were increasing the gain parameter (λ, see (2)) exponentially. As a consequence, the state evolution ended much quicker as before (in the real-life problem, in less than 500 iterations).
2. We collected feasible problem solutions generated by using the network, and applied a GA to improve the final result. We used the elitism strategy [18] to select the best-ranked solutions, and applied the multi-point crossover and the mutation genetic operators to generate new populations of feasible schedules. However, this approach has not contributed much improvement in the performance of the method.

6 Conclusions

In this paper we have presented an evolutionary approach to the school timetabling problem. Our approach uses the HTNN to perform the timetabling and the GA to simultaneously initialize and optimize the network. By using this approach we were able to produce feasible schedules in a polynomial time.

References:

[1] T.B. Cooper and J.H. Kingston, "The Complexity of Timetable Construction Problems".

[2] A. Schaerf, "A Survey of Automated Timetabling", *Artificial Intelligence Review*, 13:87-127, Kluwer Academic Publishers, 1999.

[3] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989.

[5] T.R. Jensen and B. Toft, *Graph Coloring Problems*, Wiley, John&Sons, Inc., 1994.

[6] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley Publishing Company, 1989.

[7] K.A. Smith, "Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research", *INFORMS Journal on Computing*, 11(1):15-34, 1999.

[8] M.R. Garey and D.S. Johnson, *A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[9] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biol.Cybern.*, 52:141-152, 1985.

[10] G.A. Tagliarini, J.F. Christ and E.W. Page, "Optimization Using Neural Networks", *IEEE Transactions on Computers*, 40(12):1347-1358, 1991.

[11] B. Mueller, J. Reinhardt and M.T. Strickland, *Neural Networks, An Introduction*. Springer-Verlag, 1995.

[12] V. Beiu, "Neural Network Hardware Implementations".In *Handbook of Neural Computation*. IOP Publishing Ltd and Oxford University Press, 1997.

[13] A. Johannet et al, "Specification and Implementation of a Digital Hopfield-Type Associative Memory with On-Chip Training".

[14] N. Botros and M. Abdul-Aziz, *Hardware Implementation of an Artificial Neural Network Using Field Programmable Gate Arrays (FPGA's)*. IEEE Transactions on Industrial Electronics, 41:665-667, 1994.

[15] E. Beltrami, Mathematics for Dynamic Modeling. Academic Press, Boston, 1987.

[16] C.L. Karr et al, *Solving inverse initial-value, boundary-value problems via genetic algorithm*, Engineering Applications of Artificial Intelligence, 13(6):625-633, 2000.

[17] S. Wolfram, Mathematica, *A System for Doing Mathematics by Computer*. Addison-Wesley, 1996.

[18] K.A. de Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", *Ph.D. Thesis*, University of Michigan, 1975.