Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 153 (2004) 65–79
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem

M.P. Carrasco [a,b,*], M.V. Pato [b,c]

[a] Escola Superior de Gestão, Hotelaria e Turismo, University of Algarve, Largo Eng. Sárrea Prado, n.21, 8501 859 Portimão, Portugal
[b] Centro de Investigação Operacional, University of Lisbon, Bloco C2, Sala 2.2.57, Campo Grande, 1749 Lisbon, Portugal
[c] Instituto Superior de Economia e Gestão, Technical University of Lisbon, Rua do Quelhas n.6, 1200 781 Lisbon, Portugal

Abstract

This study explores the application of neural network-based heuristics to the class/teacher timetabling problem (CTTP). The paper begins by presenting the problem characteristics in terms of hard and soft constraints and proposing a formulation for the energy function required to map the issue within the artificial neural network model. There follow two distinct approaches to simulating neural network evolution. The first uses a Potts mean-field annealing simulation based on continuous Potts neurons, which has obtained favorable results in various combinatorial optimization problems. Afterwards, a discrete neural network simulation, with discrete winner-takes-all neurons, is proposed. The paper concludes with a comparison of the computational results taken from the application of both heuristics to hard hypothetical and real CTTP instances. This experiment demonstrates that the discrete approach performs better, in terms of solution quality as well as execution time. By extending the comparison, the neural discrete solutions are also compared with those obtained from a multiobjective genetic algorithm, which is already being successfully used for this problem within a timetabling software application.

Keywords: Timetabling; Metaheuristics; Neural networks

1. Introduction

Timetabling is a typical real world scheduling activity that arises at least once a year at every educational institution. The three most common educational timetabling problem categories are examination timetabling, course timetabling and class/teacher timetabling, as they are commonly called. The basic distinction occurs in terms of the elements to be scheduled, that is, exams, course options or regular lessons. The examination timetabling problem consists in scheduling the exams for a set of courses, over a limited time

* Corresponding author. Address: Escola Superior de Gestão, Hotelaria e Turismo, University of Algarve, Largo Eng. Sárrea Prado, n.21, 8500 Portimão, Portugal.
E-mail addresses: pcarras@ualg.pt (M.P. Carrasco), mpato@iseg.utl.pt (M.V. Pato).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(03)00099-7

period, while avoiding the overlapping of exams for each student [22]. For a detailed description of this problem see the surveys made by Carter and Laporte [8] or Burke et al. [3]. In course timetabling problems, for each student a set of lectures is previously defined. Then all lectures included within the institution's set of courses must be scheduled in such a way that the overlapping of lectures for courses with students in common is minimized (vide, for instance, Downsland [13]; Kiaer and Yellen [20]). This problem arises in universities or other educational institutions with flexible curricula.

The current paper focuses on the last type, the class/teacher timetabling problem, CTTP for short. It considers the scheduling of a set of lessons (class/teacher assignments) subject to hard and soft constraints, and can be modelled in the context of combinatorial optimization. In fact, several methodologies have been proposed to tackle different combinatorial formulations for this type of timetabling problem, from graph theory based methods [32–34], to binary programming [2,29] and constraint-based approaches [31], simulated annealing [1], tabu search [12,26], genetic algorithms [10,11], and neural networks [6,16,17].

In practice, the timetabling task is often performed manually, through a slow trial-and-error procedure. Automatically tackling real world situations often requires the application of problem-specific heuristics to overcome the difficulties of structure complexity or large dimension often present in real instances. In recent decades, several heuristics have been proposed to tackle this task [9,11,25]. Among these, following the work of Hopfield and Tank [18], artificial neural networks have proven to be relatively successful in solving complex combinatorial optimization problems [19,27], including some timetabling issues similar to the CTTP [16,17], and other general timetabling problems [21,15,28]. From the practical perspective, some successful results were presented in [17], through a neural approach for a real CTTP instance based on a medium real timetabling issue from a Swedish high school of about 1000 students. In Ref. [2], using binary programming a small size real example (about 200 students) was favourably addressed. Within genetic algorithms for real CTTPs, [11] reports comparative results with both simulated annealing and tabu search heuristics, for a small instance of 10 classes and 30 teachers. Also, comparative results of a tabu search heuristic with the manual solution, are described in [26] for three real CTTPs from schools with up to 38 classes and 61 teachers.

There were major points of motivation for developing the study concerning specific heuristics for the CTTP that has led to the present report. The first was to investigate the practicability of the classic neural approaches mentioned above, to solve medium dimensioned real CTTPs. The second was to compare the neural timetabling solutions with those obtained by a multiobjective genetic algorithm successfully applied by the authors to this problem.

Hence, the paper begins with a description of the characteristics of the CTTP, in Section 2. In Section 3, the Hopfield and Tank neural network model is summarized and a specific energy function for the CTTP is proposed. Then, in Section 4, two heuristic methods for tackling the problem, based on the above neural network formulation, are investigated, namely, a continuous Potts mean-field annealing approach and a discrete winner-takes-all neuron approach. Section 5 introduces a set of hard hypothetical and real CTTP instances and the results obtained from the application of the two neural heuristics are both compared and at the end evaluated against a genetic multiobjective heuristic. Finally, Section 6 concludes with some considerations.

2. The class/teacher timetabling problem

The class/teacher timetabling problem, referred to as CTTP, is the subject of this paper. It is usually found in schools with less flexible curricula, as is the case of most secondary schools and some universities, where the majority of the predefined lessons for each course are compulsory for the classes in question and can therefore in no way overlap.

The specific class/teacher timetabling problem addressed in this work, already described in [7], can be defined as an optimization problem relative to the scheduling of a set of lessons (prior assignments of one or more rigid classes of students to one teacher and one subject) over a weekly set of time periods, using suitable rooms, while satisfying a broad spectrum of constraints. Due to the distinct nature of each constraint, two levels of importance can be identified, as follows.

The hard constraints assure the aspects related to the feasibility of a CTTP solution. For this reason all hard constraints must always be satisfied in order to obtain legal timetabling solutions. The hard constraints mentioned are described below.

(h1) The classes' curricula must be respected, i.e. all lessons assigned to each class are fully scheduled.
(h2) Each teacher and class is assigned to no more than one lesson and one room within one time period. This is a hard constraint common to all timetabling problems.
(h3) Lessons can be of different but fixed durations, given by a multiple of the time period.
(h4) For pedagogical reasons, each subject is scheduled to be given no more than once a day.
(h5) Rooms are suitable for the lessons assigned. Each lesson may require a particular type of room, in terms of seating or special resources. For example, as a rule, computer-related lessons require a room containing computers and specific software.
(h6) The teaching shift for each class is respected. In some schools there is a practicable teaching set of time periods (shift) that must always be satisfied for each class. Such is the case when scheduling lessons for a worker class which is usually performed over an evening teaching shift.
(h7) Each class and teacher must have a lunch break. This constraint imposes a free period of at least one time period, to be used for the lunch activity.
(h8) Each class's and teacher's unavailability period is respected. In fact, classes and, far more often, teachers have periods of unavailability arising from periodic professional or academic commitments.

At a secondary level of importance one finds soft constraints, representing the optional aspects that both teachers and classes would like to have satisfied in their timetables. While a wide range of non-compulsory constraints may emerge in real life, this work only addresses the most important and common ones.

(s1) The occurrence of gaps between teaching periods should be low for both classes and teachers. This condition is desirable, as it frees larger intervals of available time for study or other professional activities, such as research, besides resulting in more compact timetables.
(s2) Class and teacher timetabling preferences should be satisfied. In almost every school, classes and, to a greater extent, teachers express their preferences, which should be met whenever possible.
(s3) The number of teacher and class shifts between different teaching locations should be minimized. In addition, for institutions with multiple, geographically distant teaching locations, a minimum shift time should be respected to enable teachers and classes to move between different teaching places.

In this context, the CTTP can be defined as the combinatorial optimization problem encountered in the determination of a timetable that satisfies all hard constraints (a non-conflicting timetable or a feasible CTTP solution), while attempting to verify the soft constraints. For this reason, the degree of satisfaction of soft constraints is closely related to the final quality of the timetabling solution. Hence, the highest quality feasible solution must not only assure (h1)–(h8) but also violate the minimum of the timetabling optional requirements expressed by (s1)–(s3).

Note that the above conditions define the most important features regarding our specific CTTP. However, other constraint specifications can emerge from real timetabling (for example, the need to spread the lessons of each subject over the week) and be implemented by similar processes to those proposed in the following sections.

In terms of complexity, the CTTP shows a difficult structure which was proved to be NP-hard by Even et al. [14], even for some simplified versions. Furthermore, due to the large dimension often assumed by real-world instances, as well as the need to include additional specific-problem constraints, the CTTP has triggered a wide range of mentioned above heuristic solution techniques.

3. CTTP neural network encoding

The application of artificial neural networks to solve combinatorial optimization problems was first introduced by the seminal paper of Hopfield and Tank [18]. This work presented a neural network model for optimization purposes, characterized by a fully interconnected network with N binary neurons, whose basic features will now be briefly described. Neuron i has a net input value U_i and an output value or state V_i. The input value U_i results from the weighted sum of output values arising from the other neurons, added to a negative bias current I_i:

$$U_i = \sum_{j \neq i}^{N} w_{ji} V_j + I_i, \tag{1}$$

where, w_{ji} is the weight (synapse connection) from neuron j to i. The output V_i, which is bounded by zero and one, is calculated from U_i by an activation function. Basically there are two types of model versions of the Hopfield and Tank neural network (HTNN): discrete versions, using a hard limiter activation function, and continuous versions, which traditionally use a sigmoid-type activation function and a gain parameter.

Hopfield and Tank proved that, for a symmetrical weight matrix ($w_{ij} = w_{ji}$), the neural network converges to stable states, that is, to local minimum values of the neural energy function:

$$E(\overline{V}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} V_i V_j - \sum_{i=1}^{N} I_i V_i. \tag{2}$$

This minimization ability plays a central role in tackling combinatorial optimization problems through neural networks. For this purpose, one must formulate an appropriate energy function, whose minimum value corresponds to the optimum of the optimization problem. Usually this energy function is derived from the violations of the problem constraints in the form of a penalized sum, added to the objective expression. Using such an approach requires a careful selection of the penalizing parameters, to obtain a balance between constraints and objective terms. A generic procedure for translating combinatorial optimization problems into a neural network model has been described by [24].

For the CTTP case and assuming that L lessons are to be scheduled (involving prior assignments of T teachers to C classes), in P identical time periods and R rooms, a direct formalization [5] uses $L \times P \times R$ binary variables as follows:

$$x_{L \times P \times R} = \begin{cases} 1 & \text{if lesson } l \text{ occurs during time period } p \text{ in room } r, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, within a direct Hopfield and Tank neural encoding, each of these variables is associated with a generic two-state neuron, thus requiring $L \times P \times R$ neurons to completely encode a CTTP solution. However, using a particular factorization scheme proposed in [17], the neural network for this problem only requires $L \times P$ lesson–period neurons, and $L \times R$ lesson–room neurons. The variables V_{lp}^{x} and V_{lr}^{y} represent the states of the lesson–period neuron (l, p) and the lesson–room neuron (l, r), respectively. This encoding is far more efficient than the previous one as it significantly reduces the number of neurons involved. Furthermore, it also allows the direct implementation of some of the hard constraints mentioned in

Section 2 (more precisely constraints (h5), (h6) and (h8)), through an initial freezing of the respective neurons in appropriate states.

The remaining hard and soft constraints are stated in order to define the energy function, with the exception of constraint (h1) which will be addressed in the next section. Constraints (h2) and (h3) are implemented simultaneously through penalty terms E_1 and E_2. First, to prevent room occupation conflicts the following penalty expression is included in the energy function, to be minimized:

$$E_1 = \frac{1}{2}\sum_{p=1}^{P}\sum_{r=1}^{R}\sum_{\substack{l,l'=1\\ l\neq l'}}^{L} V_{lr}^{y} V_{l'r}^{y} S_{lp} S_{l'p}, \tag{3}$$

where $S_{lp} = \sum_{k=0}^{\mathrm{Dur}(l)-1} V_{l,p-k}^{x}$ and $\mathrm{Dur}(l)$ gives the duration of lesson l, an integer number of time periods.

In order to prevent class and teacher time conflicts, the energy component E_2 is defined as

$$E_2 = \frac{1}{2}\sum_{p=1}^{P}\sum_{\substack{l,l'=1\\ l\neq l'}}^{L} Q_{ll'}^{\mathrm{teca}} S_{lp} S_{l'p}, \tag{4}$$

where

$$Q_{ll'}^{\mathrm{teca}} = \begin{cases} 2 & \text{if lessons } l \text{ and } l' \text{ share the same teacher and class,}\\ 1 & \text{if lessons } l \text{ and } l' \text{ share the same teacher or class,}\\ 0 & \text{otherwise.}\end{cases}$$

In addition, the energy term E_3 attempts to avoid scheduling more than one lesson of the same subject on the same day (constraint (h4)):

$$E_3 = \frac{1}{2}\sum_{c=1}^{C}\sum_{\substack{l,l'\in L\mathrm{Class}(c)\\ l\neq l'}} Q_{ll'}^{sub}\sum_{d=1}^{D}\left(\sum_{p\in T\mathrm{Day}(d)} S_{lp}\right)\left(\sum_{p\in T\mathrm{Day}(d)} S_{l'p}\right), \tag{5}$$

where $L\mathrm{Class}(c)$ is the set of lessons attended by class c, $T\mathrm{Day}(d)$ represents the set of time periods of day d, D is the number of teaching days in the week, and

$$Q_{ll'}^{\mathrm{sub}} = \begin{cases} 1 & \text{if lessons } l \text{ and } l' \text{ belong to the same subject,}\\ 0 & \text{otherwise.}\end{cases}$$

To oblige satisfaction of (h7), related to the existence of lunch breaks, two auxiliary sets of variables expressing lesson participation of each class or teacher along the week's time periods, are defined as

$$A_{cp}^{\mathrm{cla}} = \sum_{l\in L\mathrm{Class}(c)} S_{lp}, \quad A_{tp}^{\mathrm{tea}} = \sum_{l\in L\mathrm{Teacher}(t)} S_{lp}, \tag{6}$$

where $L\mathrm{Teacher}(t)$ is the set of lessons taught by teacher t.

By using these variables, the energy component E_4 ensures that a lunch break of at least one time period is created each day d within the set of time periods defined by $T\mathrm{LunchF(d)}$ and $T\mathrm{Lunch}L(d)$, as follows:

$$E_4 = \frac{1}{2}\sum_{d=1}^{D}\left[\sum_{c=1}^{C}\left(\prod_{p=T\mathrm{Lunch}F(d)}^{T\mathrm{Lunch}L(d)} A_{cp}^{\mathrm{cla}}\right) + \sum_{t=1}^{T}\left(\prod_{p=T\mathrm{Lunch}F(d)}^{T\mathrm{Lunch}L(d)} A_{tp}^{\mathrm{tea}}\right)\right]. \tag{7}$$

The next energy terms provide the optimization feature of the CTTP, guaranteeing satisfaction of the soft constraints. Once again, each of these terms measures each soft constraint cost within a solution.

The first soft constraint (s1) whose goal is time contiguity of the lessons—both for classes and teachers—is enforced by the addition of E_5:

$$E_5 = \frac{1}{2}\sum_{c=1}^{C}\left(\mathrm{Sum}L_c^{\mathrm{cla}} - \left(\sum_{d=1}^{D}\sum_{p=T\mathrm{Day}F(d)}^{T\mathrm{Day}L(d)-1} A_{cp}^{\mathrm{cla}}A_{c,p+1}^{\mathrm{cla}}\right)\right) + \frac{1}{2}\sum_{t=1}^{T}\left(\mathrm{Sum}L_t^{\mathrm{tea}} - \left(\sum_{d=1}^{D}\sum_{p=T\mathrm{Day}F(d)}^{T\mathrm{Day}L(d)-1} A_{tp}^{\mathrm{tea}}A_{t,p+1}^{\mathrm{tea}}\right)\right), \tag{8}$$

where $\mathrm{Sum}L_c^{\mathrm{cla}}$ and $\mathrm{Sum}L_t^{\mathrm{tea}}$ are the weekly total number of lessons assigned to class c and teacher t, respectively. The integers $T\mathrm{Day}F(d)$ and $T\mathrm{Day}L(d)$ give the first and last time periods of day d. Note that expression E_5 effectively measures daily lesson dispersion by counting the total weekly number of clusters of lessons.

As for the desire to satisfy the classes' and teachers' time preferences (constraint (s2)) two matrices, with elements represented by $\mathrm{Pre}_{cp}^{\mathrm{cla}}$ and $\mathrm{Pre}_{tp}^{\mathrm{tea}}$, are used within E_6. Each element of these matrices is an integer (from 0-best to 5-worst) expressing the preferences of class c or teacher t to be involved in lessons during time period p, thus giving

$$E_6 = \frac{1}{2}\left[\sum_{c=1}^{C}\sum_{p=1}^{P}\mathrm{Pre}_{cp}^{\mathrm{cla}}A_{cp}^{\mathrm{cla}} + \sum_{t=1}^{T}\sum_{p=1}^{P}\mathrm{Pre}_{tp}^{\mathrm{tea}}A_{tp}^{\mathrm{tea}}\right]. \tag{9}$$

The last penalization term E_7, which implements teachers' and classes' shifts between teaching locations, is defined as follows:

$$E_7 = \frac{1}{2}\sum_{c=1}^{C}\sum_{\substack{l,l'\in L\mathrm{Class}(c)\\ l\neq l'}} Q_{ll'}^{\mathrm{loc}}\sum_{d=1}^{D}\left(\sum_{p\in T\mathrm{Day}(d)} S_{lp}\right)\left(\sum_{p\in T\mathrm{Day}(d)} S_{l'p}\right) + \frac{1}{2}\sum_{t=1}^{T}\sum_{\substack{l,l'\in L\mathrm{Teacher}(t)\\ l\neq l'}} Q_{ll'}^{\mathrm{loc}}\sum_{d=1}^{D}\left(\sum_{p\in T\mathrm{Day}(d)} S_{lp}\right)\left(\sum_{p\in T\mathrm{Day}(d)} S_{l'p}\right), \tag{10}$$

where

$$Q_{ll'}^{\mathrm{loc}} = \begin{cases} 0 & \text{if lessons } l \text{ and } l' \text{ have the same location,}\\ 1 & \text{otherwise.}\end{cases}$$

Finally, the complete neural network energy function is expressed by

$$E(\overline{V}^x, \overline{V}^y) = \alpha(E_1 + E_2 + E_3 + E_4) + \beta(E_5 + E_6 + E_7), \tag{11}$$

where α, β are real positive balance parameters. As the total penalization associated with the activation of any pair of neurons is symmetric, the corresponding weights matrix is symmetric. Hence, minimization of this customized energy function leads to final neural network states corresponding to low conflict configurations (local minima). These point to feasible or near feasible CTTP solutions satisfying most of the soft constraints.

It should be mentioned that, in this CTTP, one was only required to differentiate between two levels of constraints, hard and soft constraints (linked with α and β parameters, respectively). However, if intermediate levels of constraints were required, they could be implemented by means of an appropriate weight balance.

4. Neural network simulation approaches

To solve a combinatorial optimization problem such as the CTTP, by using a type of HTNN, different improvements can be exploited. First, this section describes a continuous Potts mean-field annealing method (CPMFA), which has produced good results when applied to complex optimization problems [23], including the current problem [6,17]. Then a discrete winner-takes-all neuron approach (DWTAN) is proposed as an alternative to tackle the same issue.

As will be seen later, in both models, hard constraint (h1) is automatically satisfied, either by the Potts or the winner-takes-all neuron solution. As a result, the equations required to ensure (h1), by forcing activation of only one neuron from a row of P or R neurons, are always satisfied, for all lessons:

$$\sum_{p=1}^{P} V_{lp}^{x} = 1, \quad \sum_{r=1}^{R} V_{lr}^{y} = 1, \tag{12}$$

where, again, V_{lp}^{x} is the state of the lesson–period neuron (l,p) and V_{lr}^{y} represents the state of the lesson–room neuron (l,r).

4.1. Continuous Potts mean-field annealing approach

As suggested by Peterson et al. [17], to improve the basic HTNN for the CTTP, the two-state Hopfield neurons were replaced by Potts multi-state neurons. To find minimal states of this neural system according to a mean-field annealing method controlled by a temperature parameter T' as suggested in [30], simulation of the dynamics of the neural network is performed by iterating the following equations:

$$V_{lp}^{x} = \frac{e^{U_{lp}^{x}}}{\sum_{p=1}^{P} e^{U_{lp}^{x}}}, \quad U_{lp}^{x} = -\frac{1}{T'}\frac{\partial E}{\partial V_{lp}^{x}}, \tag{13}$$

$$V_{lr}^{y} = \frac{e^{U_{lr}^{y}}}{\sum_{r=1}^{R} e^{U_{lr}^{y}}}, \quad U_{lr}^{y} = -\frac{1}{T'}\frac{\partial E}{\partial V_{lr}^{y}}. \tag{14}$$

Here, each of the neural sub-networks, with neural states given by state matrices $\overline{V}^{y}$ and $\overline{V}^{x}$ respectively, is updated according to the CPMFA generic algorithmic procedure outlined in Fig. 1.

The starting temperature T'_0 was determined by trial and error, thus giving $T'_0 = 1$. In addition, a maximum number of iterations must be proposed to act as a stopping criterion, here $MaxIter = 100$. Both neural sub-networks are initiated with small random perturbations around the equilibrium state values. The

START
  $T'_0 = 1; MaxIter = 100; Iter = 0$
  Initialize all neuron states
  WHILE $(Iter < MaxIter)$ **DO**
    IF $(NSum^{x} = \frac{1}{N^{x}}\sum_{i=1}^{N^{x}}(V_i^{x})^2 < 0.9)$ **THEN**
      Randomly perform one update per neuron of lesson-period sub-network by using equations (13) with T'_{iter}
    IF $(NSum^{y} = \frac{1}{N^{y}}\sum_{i=1}^{N^{y}}(V_i^{y})^2 < 0.9)$ **THEN**
      Randomly perform one update per neuron of lesson-room sub-network by using equations (14) with T'_{Iter}
    $Iter = Iter + 1$
    $T'_{Iter+1} = T'_{Iter} \times 0.95$
  Discretize all neuron states
END

Fig. 1. CPMFA algorithm.

equilibrium values are calculated by inverting the number of active neurons (not previously frozen) in each row of respective state matrices, whose elements are represented by V_{lp}^{x} or V_{lr}^{y}.

Next, the main iterative step of the algorithm is executed while the maximum iteration parameter is not exceeded. This loop consists of updating, for a specific temperature level, all neuron states V_{lp}^{x} and V_{lr}^{y}, using Eqs. (13) and (14), according to a random, asynchronous process. On the assumption that N^{x} and N^{y} represent the total number of neurons within the lesson–room and lesson–period neural sub-networks, updating is performed as long as the respective global neural saturation, defined respectively as $NSum^{x}$ and $NSum^{y}$, is below 0.9. This situation occurs while the neurons assume overall non-discrete values, thus requiring more iterations to define final discrete output values (1 or 0). At the end of this loop, the temperature level and the iteration counter are changed.

The final step forces the discretization, according to Eq. (12), of all neurons not in state 0 or 1, thus allowing direct extraction of the binary solution for the CTTP.

4.2. Discrete winner-takes-all neuron approach

An analogy exists between discrete and continuous versions of HTNN. In fact, in continuous versions, as the temperature parameter is reduced to extremely low levels, the neurons change from initial continuous real values in $[0, 1]$ to 0, 1 values. The main disadvantage of using discrete HTNN versions for optimization purposes concerns the difficulty of avoiding low quality local solutions, which often appear as a result of premature convergence. On the other hand, discrete versions are much more simple and efficient to implement computationally than continuous ones.

Bearing in mind the above considerations, the proposed neural heuristic, which is an alternative to the continuous one previously described in Section 4.1, uses discrete competition winner-takes-all neurons, whose dynamics for both sub-networks are defined by the following equations:

$$V_{lp}^{x} = \begin{cases} 1 & \text{if } U_{lp}^{x} = \max_{i \in P} \left\{ U_{li}^{x} = -\frac{\partial E}{\partial V_{li}^{x}} \right\}, \\ 0 & \text{otherwise;} \end{cases} \tag{15}$$

$$V_{lr}^{y} = \begin{cases} 1 & \text{if } U_{lr}^{y} = \max_{i \in R} \left\{ U_{li}^{y} = -\frac{\partial E}{\partial V_{li}^{y}} \right\}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

For a specific lesson l, the definition of these winner-takes-all neurons forces the activation of only one from the respective row of neurons, which represents the P or R assignment alternatives associated with that lesson l. For each updating procedure using Eqs. (15) or (16), the triggered winner neuron (which is turned on with a state equal to 1) corresponds to the alternative with the minimal additional penalty for the energy function. All the remaining neurons of row l are updated to a state equal to 0. In terms of the basic HTNN model, this process can be compared to a context in which from a row of neurons connected by extreme mutual lateral inhibition weights, the neuron with maximal excitation is activated, while the remainder is, as a result, forced to be turned off.

This discrete approach also includes an improvement mechanism to escape from premature, low quality local optima. This is implemented through a condition-activated sub-procedure.

The complete algorithmic procedure is outlined in Fig. 2 and starts by proposing the maximum number of iterations, $MaxIter = 1000$, and the maximum number of iterations without energy function improvement, $MaxEconst = 20$. The amount of random rows associated with lessons to be arbitrarily selected and changed in order to avoid a potentially low quality local optimum is defined as $RandRows = 10$. After randomly constructing an initial discrete solution, the algorithm enters the main loop. This consists in random, row by row, updating of all neuron states, according to Eqs. (15) or (16). Two flags, *StableX* and

```
START
  MaxIter = 1000; Iter = 0; MaxEconst = 20; EConst = 0
  RandRows = 10; EMin = +∞
  Generate initial discrete states for both neural sub-networks
  WHILE (Iter < MaxIter) DO
    StableX = True; StableY = True
    FOR l = 1 TO L DO
      WHILE (there exists a row in the lesson-period sub-network to be up-
       dated) DO
        Randomly choose a non-updated row k (lesson k)
        Update neuron states of row k, V^x_kp, by using (15)
        IF (any active neuron in row k is changed) THEN StableX = False
      WHILE (there exists a row in the lesson-room sub-network to be updated)
       DO
        Randomly choose a non-updated row k
        Update neuron states of row k, V^y_kr by using (16)
        IF (any active neuron in row k is changed) THEN StableY = False
    Calculate energy E(V̄^x, V̄^y)
    IF E(V̄^x, V̄^y) < EMin THEN DO
      Save best solution; EMin = E(V̄^x, V̄^y); EConst = 0
       ELSE
      EConst = EConst + 1
    IF (StableX AND StableY) OR (EConst > MaxEconst) THEN DO
      Reset neurons with non-zero energy costs
      Randomly choose RandRows rows and active one neuron per row
      EConst = 0
    Iter = Iter + 1
END
```

Fig. 2. DWTAN algorithm.

StableY, are used to detect stable configurations of the sub-networks. Next, the energy function value is calculated and an eventual new best solution is saved, otherwise the *Econst* counter is increased. Then, if both neural sub-networks are stable or the maximum number of iterations without energy improvements is exceeded, two actions occur. Firstly, the neurons with an associated positive weight on the energy function are turned off and, secondly, a number of *RandRows* rows are randomly selected and respective neuron states modified so as to allow the algorithm to search for a new network configuration. The final solution is the one with the lower energy value, thus corresponding to a higher quality CTTP solution and hopefully a feasible one.

5. Computational results

To compare the computational behavior of the aforesaid neural network heuristics, two sets of class/teacher timetabling instances were used. The first set consists in five hard hypothetical instances, with distinct dimensions. The second is a collection of three real instances taken from a university institution in Portugal.

5.1. Description of instances

The hypothetical cases were extracted from the OR-Library [4] (available at http://mscmga.ms.ic.ac.uk/info.html). The main characteristics and number of neurons required to map them are presented in Table 1. These instances presume that a single room has been previously assigned for each lesson. Hence, the neural simulation is restricted to the lesson–period sub-network only.

Table 1
Hard hypothetical CTTP instances—characteristics and number of neurons

Problem instance	Lessons	Time periods	Rooms	Teachers	Classes	Total number of neurons
HDTTP4	120	30	4	4	4	3600
HDTTP5	150	30	5	5	5	4500
HDTTP6	180	30	6	6	6	5400
HDTTP7	210	30	7	7	7	6300
HDTTP8	240	30	8	8	8	7200

Table 2
Hard hypothetical CTTP instances—difficulty issues

Problem instance	Time-slot occupation index	Lesson–room rigidity index	Active constraints
HDTTP4	1	1	h1, h2, h5
HDTTP5	1	1	h1, h2, h5
HDTTP6	1	1	h1, h2, h5
HDTTP7	1	1	h1, h2, h5
HDTTP8	1	1	h1, h2, h5

For the purpose of evaluating the difficulty level associated with each situation, some generic timetabling indexes are defined and calculated in Table 2. One should refer that, these instances were taken from basic constraint satisfaction problems with active hard constraints alone (h1), (h2) and (h5), whose solution corresponds to a full lesson, conflict-free timetable without soft requirements.

The time-slot occupation index measures the level of occupancy by dividing the number of lessons to be scheduled by the number of time-slots (rooms $\times$ time periods). Another issue capable of affecting the degree of difficulty is the number of alternative rooms that can be assigned to each lesson. To evaluate this aspect, the lesson–room rigidity index is calculated by dividing the number of lessons by the total number of alternative rooms. As described in Table 2, indexes show both difficult, fully constrained timetabling contexts.

Three examples of real cases are presented in Table 3, derived from first semester data of the Escola Superior de Gestão, Hotelaria e Turismo of Algarve University in Portugal, on the institution's two campuses (Faro and Portimão).

PREAL is a small dimension instance based on a fraction of the ESGHT2 courses. However, the number of available rooms was reduced in order to increase difficulty level. For this situation, hard constraints (h1), (h2), (h3), (h5) and soft constraint (s1) were considered, as shown in Table 4. ESGHT1 represents the restricted instance occurring on Portimão campus. Its structure is characterized by all hard and soft constraints except (s3). Finally, ESGHT2 represents a complete CTTP instance in which all constraints mentioned in Section 2 are applied. The last columns of Table 3 show the number of lesson–room and lesson–period neurons required to map these instances, which all include lessons of multiple lengths.

Table 3
Real CTTP instances—characteristics and number of neurons

Problem instance	Lessons	Time periods	Rooms	Teachers	Classes	Number of neurons		
						Room	Period	Total
PREAL	37/48*	12	4	6	6	148	444	592
ESGHT1	174/209*	50	9	51	32	1566	8700	10 266
ESGHT2	568/626*	50	27	107	92	15 363	28 450	43 813

*Number of equivalent single time period lessons.

Table 4
Real CTTP instances—difficulty issues

Problem instance	Time-slot occupation index	Lesson–room rigidity index	Active constraints
PREAL	1	0.27	h1, h2, h3, h5, s1
ESGHT1	0.46	0.20	h1,...,h8, s1, s2
ESGHT2	0.46	0.14	h1,...,h8, s1,...,s3

Table 4 indicates the difficulty indexes registered, revealing that all instances have a significant level of time-slot occupation, especially PREAL, which is totally saturated. Furthermore, levels of lesson–room rigidity tend to increase instance intricacy by imposing significant compulsory restrictions through hard constraint (h5). For all real instances the standard time period duration was one and an half hour.

5.2. Comparison of results from the neural approaches

For the purpose of computational implementation, the two neural heuristics described above were programmed in Delphi 5.0, and executed on a Pentium III 800 Mhz with 128 MRam.

The tests were conducted by performing 20 complete runs for each instance and neural heuristic. The energy function balance parameters for all runs were defined experimentally as $\alpha = 100$ and $\beta = 1$. It should be mentioned that different balance schemes were tested between hard and soft constraints and also within the soft constraint group, using distinct parameters for each constraint. However, this did not significantly affect the results.

Table 5 shows the computational results for the first set of instances. The DWTAN approach clearly achieves a higher performance when compared to the CPMFA, both in terms of solution quality and execution time. In fact, the discrete neural heuristic always attained the optimal solution, in particular for HDTTP4, where all runs obtained a zero conflict timetable, that is a feasible solution. Due to full saturation of these instances, the DWTAN parameters *MaxIter* and *RandRows* were changed to 10 000 and 2, respectively.

Table 6 presents the comparative behavior of the heuristics applied to the real instances restricted to the set of hard constraints, that is, with the energy function given by

$$E(\overline{V}^x, \overline{V}^y) = \alpha(E_1 + E_2 + E_3 + E_4). \tag{17}$$

The above findings illustrate the capability of the neural methods to produce feasible solutions for the proposed situations. Once again, the figures testify to the DWTAN method's superior performance. It should be noted that for the larger cases (ESGHT1 and ESGHT2) this approach always found a feasible solution in less than 2 minutes.

Table 5
Hard hypothetical CTTP instances—computation results after 20 runs

Problem instance	CPMFA			DWTAN		
	Energy value		Run time average (seconds)	Energy value		Run time average (seconds)
	Min.	Average		Min.	Average	
HDTTP4	5	10.7	50	0	0	18
HDTTP5	8	13.2	154	0	0.4	102
HDTTP6	11	18.7	308	0	1.65	213
HDTTP7	18	25.6	426	0	2.1	305
HDTTP8	15	28.6	1454	0	3.25	1237

Table 6
Real CTTP instances restricted to hard constraints—computation results after 20 runs

Problem instance	CPMFA			DWTAN		
	Energy value		Run time average (seconds)	Energy value		Run time average (seconds)
	Min.	Average		Min.	Average	
PREAL	1	1.7	136	0	0.2	5
ESGHT1	0	0.25	1251	0	0	13
ESGHT2	0	0.65	6583	0	0	116

Table 7
Real CTTP instances—computation results after 20 runs

Problem instance	CPMFA			DWTAN			Manual approach energy value
	Energy value		Run time average (seconds)	Energy value		Run time average (seconds)	
	Min.	Average		Min.	Average		
PREAL	131	162	251	27	50.4	27	–
ESGHT1	302	358.2	3310	272	289	143	328
ESGHT2	770	853.5	16 214	641	667.2	517	864

Finally, the set of real instances was tackled by both neural heuristics, bearing in mind all constraints (hard and soft). The results are summarized in Table 7.

Once again, the computational figures indicate that DWTAN is more effective in producing high quality solutions than CPMFA. For instance, for ESGHT2, the discrete version was, on average, able to find high quality solutions in less than 10 minutes, while the continuous one took over four hours to obtain an inferior quality solution. For purposes of comparison, the last column of Table 7 shows the energy value corresponding to the institution's actual manual solution produced over a week by a three-person team. As may be seen, the quality of the solutions found by both neural heuristics is globally superior to the quality obtained by manual procedure. This is particulary evident in the case of the DWTAN, which significantly reduced soft constraint cost.

5.3. Extending the comparison to a genetic heuristic

The question of comparing alternative methods to solve timetabling problems is complicated by the fact that these problems appear in many forms with distinct particularities. In practice, each problem becomes a unique issue. Methodologies are often customized for specific cases or instances of a timetabling problem. Should they not tackle the same instances, no comparison between them is possible. For this reason the best results coming from the neural network approaches were compared with the results obtained for the same instances, by a multiobjective genetic heuristic described in detail in [7]. The multiobjective algorithm MOGA was coded in Delphi 5 and is based on the concept of Pareto non-dominance to find efficient solutions for the CTTP regarding minimization of both teacher and class constraint violation, as separate objectives. This approach uses a matrix chromosome representation for the CTTP solution. Additionally, during a maximum of 10 000 iterations a population of 50 chromosomes evolves, while maintaining an elitist secondary population. The population is, in each iteration, submitted to specific-purpose genetic operators of selection, crossover and mutation.

For comparison purposes the values of both objective functions were added and the hard and soft constraints were considered along with the same balance scheme used in the neural approach. Results from neural and genetic algorithms are presented in Table 8.

Table 8
Comparative results between DWTAN and MOGA

Problem instance	DWTAN				MOGA			
	Energy value		% Feasible	Run time average (seconds)	Energy value		% Feasible	Run time average (seconds)
	Min.	Average			Min.	Average		
HDTTP4	0	0	100	18	0	0.6	25	76
HDTTP5	0	0.4	65	102	0	12.4	12	131
HDTTP6	0	1.65	30	213	2	11.1	0	276
HDTTP7	0	2.1	10	305	9	23.85	0	472
HDTTP8	0	3.25	5	1237	12	31.25	0	1508
PREAL	27	50.4	75	34	27	68.4	43	63
ESGHT1	272	289	100	143	267	395.7	27	358
ESGHT2	641	667	100	517	638	751.2	21	561

As regards to the minimum energy value, one may see that DWTAN obtained better results than that of the MOGA for the hypothetical instances, in particular for HDTTP6, HDTTP7 and HDTTP8, where the genetic algorithm couldn't even find a feasible solution. For real cases, the MOGA produced minor improved timetables, for ESGHT1 and ESGHT2. For PREAL the two methods attained equal quality solutions. As for the percentage of feasible solutions obtained, a clear advantage was found in the neural approach which always gave non-conflicting timetables, all of them comparable in quality with the better one from MOGA, but taking a shorter average execution time. Should one consider the time required by each run of the algorithm to attain the stoping criteria, this time difference is even more conspicuous and significant.

6. Conclusions

This article has presented an experimental comparison of two neural network-based heuristics to tackle the class/teacher timetabling problem. First, the problem characteristics were described and a specific energy function was advocated for neural optimization purposes. Then the continuous Potts mean-field annealing and the discrete winner-takes-all neuron approaches were described and compared using sets of hard hypothetical and real CTTP instances. Attention is drawn to the fact that although far more tests were performed, within reality, only three typical instances are reported here. The paper also presents results obtained from a multibojective genetic heuristic previously developed for the CTTP.

Clearly, in terms of solution quality the computational experiments indicate that the DWTAN is preferable as being the most effective neural heuristic, both for hard hypothetical and real test sets. In addition the authors found that DWTAN is significantly faster than CPMFA, particularly for the larger instances. These experiments suggest that the discrete neural algorithm also constitutes an effective method of generating non-conflicting timetables. In fact, a more profound study of the performance of the neural methods for real instances was undertaken in two contexts involving hard constraints alone, and hard plus soft constraints. When considering only hard constraints, both neural heuristics performed satisfactorily. Particular attention is drawn to the DWTAN, which always found feasible solutions for two instances, within a few seconds.

Compared to the results obtained from the multiobjective genetic heuristic, the discrete neural results were also very satisfactory. Moreover, the solutions obtained from this neural approach were effectively implemented in school for instance ESGHT1 in the first semester of the last academic year within a timetabling software application (GAHOR), which continues to be developed for the solution of more

generic CTTPs. On extending the comparison to include the manual approach, DWTAN heuristic's output may be regarded as highly satisfactory. It produced highly improved solutions within a fraction of the time taken by the manual approach.

In future, further research will be developed on the application of the proposed discrete winner-takes-all neural heuristic to other large CTTP instances, which are already being tackled. Moreover, the natural parallelization capability of neural systems may be explored to improve algorithm performance.

References

[1] D. Abramson, Constructing school timetables using simulated annealing: Sequential and parallel algorithms, Management Science 37 (1982) 83–113.
[2] T. Birbas, S. Daskalaki, E. Housos, Timetabling for Greek high schools, Journal of the Operational Research Society 48 (1997) 1191–1200.
[3] E. Burke, D. Elliman, R. Weare, Examination timetabling in British universities—A survey, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling, LNCS vol. 1153, Springer-Verlag, Edinburgh, 1996, pp. 76–90.
[4] J. Beasley, OR-Library: Distributing test problems by electronic mail, Journal of the Operational Research Society 41 (11) (1990) 1069–1072.
[5] M.P. Carrasco, Redes Neuronais na Elaboração de Horários Escolares, Master thesis, Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa, 1995.
[6] M.P. Carrasco, M.V. Pato, A potts neural network heuristic for the class/teacher timetabling problem, in: Proceedings of the 4th Metaheuristics International Conference, 2001, pp. 139–142.
[7] M.P. Carrasco, M.V. Pato, A multiobjective genetic algorithm for the class/teacher timetabling problem, in: E. Burke, E. Erben (Eds.), The Practice and Theory of Automated Timetabling III, LNCS vol. 2079, Springer-Verlag, Konstanz, 2001, pp. 3–17.
[8] M. Carter, G. Laporte, Recent developments in practical examination timetabling, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling, LNCS vol. 1153, Springer-Verlag, Edinburgh, 1996, pp. 3–21.
[9] M. Carter, G. Laporte, Recent developments in pratical examination timetabling, in: E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling II, LNCS vol. 1408, Springer-Verlag, Berlin, 1998, pp. 3–21.
[10] A. Colorni, M. Dorigo, V. Maniezzo, Genetic algorithms and highly constrained problems: the timetabling case, in: Proceedings of the First International Conference on Parallel Problem Solving from Nature, 1982, pp. 55–59.
[11] D. Colorni, M. Dorigo, V. Maniezzo, Metaheuristics for high-school timetabling, Computational Optimization and Applications 9 (3) (1998) 277–298.
[12] D. Costa, A tabu search algorithm for computing an operational timetable, European Journal of Operational Research 76 (1994) 98–110.
[13] K. Dowsland, A timetabling problem in which clashes are inevitable, Journal of the Operational Research Society 41 (10) (1990) 907–908.
[14] S. Even, A. Itai, A. Shamir, On the complexity of timetabling and multicommodity flow problems, SIAM Journal of Computation 5 (4) (1976) 691–703.
[15] P. Gianoglio, Application of neural networks to timetable construction, in: Proceedings of the Third International Workshop on Neural Networks and Their Applications, Nîmes, France, 1990, pp. 53–77.
[16] L. Gislén, C. Peterson, B. Söderberg, Teachers and classes with neural networks, International Journal of Neural Systems 1 (1989) 167–176.
[17] L. Gislén, C. Peterson, B. Söderberg, Complex scheduling with potts neural networks, Neural Computation 4 (1992) 805–831.
[18] J. Hopfield, D. Tank, Neural computation of decisions in optimization problems, Biological Cybernetics 52 (1985) 141–152.
[19] C. Looi, Neural network methods in combinatorial optimization, Computer and Operations Research 19 (3/4) (1992) 191–208.
[20] L. Kiar, J. Yellen, Weighted graphs and university course timetabling, Computer and Operations Research 19 (1) (1992) 59–67.
[21] M. Kovacic, Timetable construction with markovian neural networks, European Journal of Operational Research 69 (1993) 92–96.
[22] L. Paquete, C. Fonseca, A study of examination timetabling with multiobjective evolutionary algorithms, in: Proceedings of the 4th Metaheuristics International Conference, 2001, pp. 149–153.
[23] C. Peterson, B. Söderberg, A new method for mapping optimization problems onto neural networks, International Journal of Neural Systems 1 (3) (1989) 3–22.
[24] L. Ramanujam, P. Sadayappan, Optimization by neural networks, in: Proceedings of the IEEE Second International Neural Network Conference, vol. 2, 1988, pp. 325–332.
[25] A. Schaerf, A survey of automated timetabling, Artificial Intelligence Review 13 (1999) 87–127.

[26] A. Schaerf, Tabu search techniques for large high-school timetabling problems, in: Proceeding of the Fourteenth National Conference on Artificial Intelegence, Portland, 1996, pp. 363–368.
[27] K. Smith, Neural networks for combinatorial optimization: A review of more than a decade of research, INFORMS Journal on Computing 11 (1999) 15–34.
[28] K. Tsuchiya, Y. Takefuji, A neural network parallel algorithm for meeting schedule problems, International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies 7 (3) (1997) 205–213.
[29] A. Tripathy, School timetabling—a case in large binary integer linear programming, Management Science 30 (12) (1984) 1473–1498.
[30] D. Van Den Bout, T. Miller, Improving the performance of the hopfield-tank neural network through normalization and annealing, Biological Cybernetics 62 (1989) 123–139.
[31] M. Yoshikawa, E. Kaneko, Y. Nomura, M. Watanabe, A constraint-based approach to high-school timetabling problems: A case study, in: Proceedings of the 12th Conference on Artificial Intelligence, vol. 94, 1994, pp. 1111–1116.
[32] D. Werra, An introduction to timetabling, European Journal of Operational Research 19 (1985) 151–162.
[33] D. Werra, Some combinatorial models for course scheduling, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling, LNCS vol. 1153, Springer-Verlag, Edinburgh, 1996, pp. 296–308.
[34] D. Werra, The combinatorics of timetabling, European Journal of Operational Research 96 (1997) 504–513.