

EFFICIENT TIMETABLING FORMULATIONS FOR HOPFIELD NEURAL NETWORKS

KATE A. SMITH

School of Business Systems, Monash University, Clayton, Victoria, 3168, Australia.

DAVID ABRAMSON and DAVID DUKE

School of Computer Science and Software Engineering, Monash University, Clayton, Victoria, 3168, Australia.

ABSTRACT:

Timetabling problems involve allocating classes, teachers, and rooms to periods to minimise clashes. When mapping such timetabling problems onto a Hopfield neural network, the dimensionality of the network can become quite large. This paper proposes an alternative formulation for timetabling which reduces the dimensionality of the network, and enables larger problems to be solved more efficiently. Benchmark data sets are used to demonstrate the improvement in performance that can be obtained by using a more efficient formulation, and results are compared to those obtained using simulated annealing.

INTRODUCTION

Timetabling is an area of research that finds application in many areas of business and engineering. Whether the entities being timetabled are people (as in staff rostering, interview scheduling, or school timetabling) or jobs for machines, the problem is similar: one of constraint satisfaction. In this paper we focus on the school timetabling problem. In its most general form the problem involves the assignment of a teacher, class and venue to a particular time-slot or period in such a way that no person is assigned to more than one venue at any time, and no venue has more than one class and teacher in it at any time. Such problems are NP-complete [1] and so heuristic approaches are used in order to obtain a useful solution within a reasonable amount of computation time. Various metaheuristic techniques have been used successfully over the last decade or so, including simulated annealing [2].

The approach taken in this paper is to use the same benchmark data sets used by Abramson and Dang [2] to compare the performance of different neural network formulations for timetabling problems with venue allocations. We also compare the performance with the simulated annealing results of Abramson and Dang [2] for benchmarking purposes.

TIMETABLING FORMULATIONS

Let $\mathcal{V}$ be a set of V venues (rooms) and $\mathcal{C}$ and $\mathcal{T}$ be the set of C classes and T teachers. The requirements matrix is of dimension $C \times T \times V$ where R_{ijk} is the number of periods teacher j is to meet class i in venue k for the duration of the timetable (a set $\mathcal{P}$ of P periods). Suppose we define a boolean variable

$$X_{ijkl} = \begin{cases} 1 & \text{if class } i \text{ and teacher } j \text{ are assigned to venue } k \text{ in period } l \\ 0 & \text{otherwise} \end{cases}$$

Then a solution to the timetabling problem involves satisfying the constraints (TT1):

$$\sum_{l=1}^P X_{ijkl} = R_{ijk} \quad \forall i \in \mathcal{C}, j \in \mathcal{T}, k \in \mathcal{V} \quad (1)$$

$$\sum_{j=1}^T \sum_{k=1}^V X_{ijkl} \leq 1 \quad \forall i \in \mathcal{C}, l \in \mathcal{P} \quad (2)$$

$$\sum_{i=1}^C \sum_{k=1}^V X_{ijkl} \leq 1 \quad \forall j \in \mathcal{T}, l \in \mathcal{P} \quad (3)$$

$$\sum_{i=1}^C \sum_{j=1}^T X_{ijkl} \leq 1 \quad \forall k \in \mathcal{V}, l \in \mathcal{P} \quad (4)$$

$$X_{ijkl} = 0 \text{ or } 1 \quad \forall i \in \mathcal{C}, j \in \mathcal{T}, k \in \mathcal{V}, l \in \mathcal{P} \quad (5)$$

Equation (1) ensures the requirements are satisfied, while equations (2), (3), and (4) prevent class, teacher, and venue clashes respectively. Equation (5) is required to ensure the integrality of the solutions. Unfortunately, the dimensions of this formulation are rather large for practical sized problems. For this reason, techniques that use this formulation are unlikely to be competitive with heuristics that treat the problem as one of mapping a triplet (class, teacher, venue) to a period. This formulation is nonetheless the one arrived at when extending the work of Hopfield and Tank [3] to solve the class-teacher-venue problem.

In order to generate neural network results that are competitive with simulated annealing and other approaches that have used the idea of swapping the position of triplets in a timetable, we now propose an alternative timetabling formulation that is similar in concept, and better suited to mapping onto a Hopfield neural network. Suppose there are τ triplets that need to be timetabled across the set of P periods. Each triplet takes the form “class i meets teacher j in venue k ”. Each unique triplet will appear R_{ijk} times in the set of all triplets Υ , so that the total number of triplets is determined by $\tau = \sum_{i=1}^C \sum_{j=1}^T \sum_{k=1}^V R_{ijk}$.

The task of assigning each triplet in the set Υ to a unique period can then be formulated using the boolean variable:

$$X_{tp} = \begin{cases} 1 & \text{if triplet } t \text{ is assigned to period } p \\ 0 & \text{otherwise} \end{cases}$$

In order to identify possible clashes, we need to be able to determine if a given class, teacher, or venue appear in a given triplet. Thus we define the constant matrices $\hat{\mathbf{C}}$, $\hat{\mathbf{T}}$, $\hat{\mathbf{V}}$, which function as look-up tables with $\hat{C}_{it} = 1$ if class i is in triplet t , $\hat{T}_{jt} = 1$ if teacher j is in triplet t , $\hat{V}_{kt} = 1$ if venue k is in triplet t . The solution matrix $\mathbf{X}$ then defines a valid solution to the timetabling problem if the following constraints are satisfied (TT2):

$$\sum_{p=1}^P X_{tp} = 1 \quad \forall t \in \Upsilon, \quad (6)$$

$$\sum_{t=1}^{\tau} X_{tp} \hat{C}_{it} \leq 1 \quad \forall i \in \mathcal{C}, p \in \mathcal{P} \quad (7)$$

$$\sum_{t=1}^{\tau} X_{tp} \hat{T}_{jt} \leq 1 \quad \forall j \in \mathcal{T}, p \in \mathcal{P} \quad (8)$$

$$\sum_{t=1}^{\tau} X_{tp} \hat{V}_{kt} \leq 1 \quad \forall k \in \mathcal{V}, p \in \mathcal{P} \quad (9)$$

$$X_{tp} = 0 \text{ or } 1 \quad \forall t \in \Upsilon, p \in \mathcal{P} \quad (10)$$

Equation (6) ensures each triplet appears exactly once in the timetable, while equations (7), (8), and (9) prevent class, teacher, and venue clashes, and equation (10) ensures the integrality of the solutions. Clearly this formulation is more compact than TT1, due to the reduction of dimensionality.

HOPFIELD NEURAL NETWORKS

Hopfield neural networks [4,5] are a biologically inspired mathematical tool that can be used to solve difficult optimisation problems. Biological modelling of the human brain is attempted by utilising a fully inter-connected system of N neurons. In this paper, we use discrete Hopfield networks, which are networks of binary states. Neuron i has internal state u_i and output level v_i of 0 or 1. The internal state u_i incorporates a bias current denoted by I_i , and the weighted sums of outputs from all other neurons. The weights, which determine the strength of the connections from neuron i to j , are given by W_{ij} . The relationship between the internal state of a neuron and its output level is determined by a discrete activation function $g(u_i) = 1$ if $u_i > 0$, and 0 otherwise. The time evolution of the Hopfield network is determined by the following equations, which update one neuron at a time:

$$u_i(t+1) = u_i(t) + \sum_j W_{ij} v_j + I_i \quad \text{where } v_i = g(u_i). \quad (11)$$

Hopfield networks can be extremely useful in solving difficult combinatorial optimisation problems since provided the weights are symmetric ($W_{ij} = W_{ji}$), the network converges to a minimum of the energy function

$$\mathcal{E}(\mathbf{v}) = -\frac{1}{2} \sum_i \sum_j W_{ij} v_i v_j - \sum_i I_i v_i. \quad (12)$$

Hopfield and Tank [3] showed that if a 0-1 optimisation problem can be expressed in terms of an energy function of the form given by (12), a Hopfield network can be used to find locally optimal solutions of the energy function, which may translate to the local minimum solutions of the optimisation problem. Typically, the network energy function is made equivalent to the objective function of the optimisation problem which is to be minimised, while the constraints of the problem are included in the energy function as penalty terms. The network parameters can then be inferred by comparison with the standard energy function given by (12). We refer the interested reader to [6] for further details of Hopfield networks for optimisation.

Weight Derivations

Focusing first on TT1, the cost and constraints of the problem can be expressed as penalty terms in a single function:

$$\begin{aligned} & \frac{\alpha}{2} \sum_{i=1}^C \sum_{j=1}^T \sum_{k=1}^V \left(\sum_{l=1}^P X_{ijkl} - R_{ijk} \right)^2 + \frac{\beta}{2} \sum_{i=1}^C \sum_{j=1}^T \sum_{k=1}^V \sum_{l=1}^P \sum_{\substack{j'=1 \\ j' \neq j}}^T \sum_{\substack{k'=1 \\ k' \neq k}}^V X_{ijkl} X_{ij'k'l} \\ & + \frac{\chi}{2} \sum_{i=1}^C \sum_{j=1}^T \sum_{k=1}^V \sum_{l=1}^P \sum_{\substack{i'=1 \\ i' \neq i}}^C \sum_{\substack{k'=1 \\ k' \neq k}}^V X_{ijkl} X_{i'jk'l} + \frac{\gamma}{2} \sum_{i=1}^C \sum_{j=1}^T \sum_{k=1}^V \sum_{l=1}^P \sum_{\substack{i'=1 \\ i' \neq i}}^C \sum_{\substack{j'=1 \\ j' \neq j}}^T X_{ijkl} X_{i'j'kl} \end{aligned} \quad (13)$$

where α, β, χ , and γ are the penalty parameters chosen to equally balance the terms of the function. The first term of equation (13) is zero only if the requirements constraint (1) is satisfied, and is positive otherwise. The remaining three terms each encourage satisfaction of the constraints (2), (3), and (4) by counting pairwise products and returning a non-zero value if two 1's are found in the same dimension of the solution matrix $\mathbf{X}$, representing a clash.

After expanding and rearranging the function (13), comparison with the energy function (12) reveals that the two equations are equivalent provided

$$\begin{aligned} W_{ijkl, i'j'k'l'} &= -\alpha \delta_{ii'} \delta_{jj'} \delta_{kk'} - \beta \delta_{ii'} (1 - \delta_{jj'}) (1 - \delta_{kk'}) \delta_{ll'} \\ &\quad - \chi (1 - \delta_{ii'}) \delta_{jj'} (1 - \delta_{kk'}) \delta_{ll'} - \gamma (1 - \delta_{ii'}) (1 - \delta_{jj'}) \delta_{kk'} \delta_{ll'} \end{aligned} \quad (14)$$

$$I_{ijkl} = \alpha R_{ijk} \quad (15)$$

where $\delta_{yy'}$ is the Kronecker-Delta function, $\delta_{yy'} = 1$ if $y = y'$ and 0 otherwise.

The second class-teacher-venue formulation, TT2, can similarly be mapped onto a Hopfield neural network. For TT2, a penalty function can be constructed which will be minimised when all constraints are satisfied. One such function is

$$\begin{aligned} & \frac{\alpha}{2} \sum_{t=1}^{\tau} \left(\sum_{p=1}^P X_{tp} - 1 \right)^2 + \frac{\beta}{2} \sum_{i=1}^C \sum_{p=1}^P \sum_{t=1}^{\tau} \sum_{\substack{t'=1 \\ t' \neq t}}^{\tau} X_{tp} \hat{C}_{it} X_{t'p'} \hat{C}_{it'} \\ & + \frac{\chi}{2} \sum_{j=1}^T \sum_{p=1}^P \sum_{t=1}^{\tau} \sum_{\substack{t'=1 \\ t' \neq t}}^{\tau} X_{tp} \hat{T}_{jt} X_{t'p'} \hat{T}_{jt'} + \frac{\gamma}{2} \sum_{k=1}^V \sum_{p=1}^P \sum_{t=1}^{\tau} \sum_{\substack{t'=1 \\ t' \neq t}}^{\tau} X_{tp} \hat{V}_{kt} X_{t'p'} \hat{V}_{kt'} \end{aligned} \quad (16)$$

The first term of (16) will be zero if each triplet is represented exactly once in the timetable. The second term attempts to minimise class clashes by inspecting all triplets involving a given class in a given period, and summing any non-zero pairwise products found. This term will be zero if the timetable is clash-free for all classes in all periods. Similarly, the third and fourth terms attempt to minimise teacher and venue clashes.

Rearranging this function in a similar manner to that described for TT1 gives the following parameters:

$$\begin{aligned} W_{tp, t'p'} &= -\alpha \delta_{tt'} - \beta (1 - \delta_{tt'}) \delta_{pp'} \left(\sum_{i=1}^C \hat{C}_{it} \hat{C}_{it'} \right) \\ &\quad - \chi (1 - \delta_{tt'}) \delta_{pp'} \left(\sum_{j=1}^T \hat{T}_{jt} \hat{T}_{jt'} \right) - \gamma (1 - \delta_{tt'}) \delta_{pp'} \left(\sum_{k=1}^V \hat{V}_{kt} \hat{V}_{kt'} \right) \end{aligned} \quad (17)$$

$$I_{ijkl} = \alpha \quad (18)$$

Modifications

In order to improve the quality and speed of the solutions, we have made some modifications to the steepest descent dynamics of the Hopfield network. These are based on the observation that the network works hardest and most productively (as measured by a rapidly decreasing energy function) during the first few iterations. An iteration is defined as one complete update of all N neurons. We therefore terminate the updating after a small number of iterations, introduce stochasticity into the neuron states to perturb the solution, renormalise the u values to 0 or 1 based on the current v values, and restart the descent. This procedure is repeated for a certain number of descents. The stochasticity at the end of each iteration is created by randomly choosing an “off” neuron and switching it to “on”. A threshold is used to control the stochasticity, so that if the threshold is 0.5, then there is an equal chance of a neuron turning “on”, but if the threshold was increased to say 0.9, then there is only a 10% chance of a neuron switching “on”. Neurons are easily switched “off” through energy minimisation.

RESULTS

The data sets used in this paper are the same ones benchmarked by Abramson and Dang [2], and have been randomly constructed so that the timetable is totally constrained, ie. each period contains every class, teacher, and venue. Since the solution is generated, we know there is a perfect solution to each problem, but the lack of flexibility in the final solution makes it extremely difficult to find. Table 1 shows the characteristics of the four data sets.

Problem Name	Number of Classes (C)	Number of Teachers (T)	Number of Venues (V)	Number of Periods (P)	Number of Triplets (τ)
hdtt5	5	5	5	30	150
hdtt6	6	6	6	30	180
hdtt7	7	7	7	30	210
hdtt8	8	8	8	30	240

Table 1: Description of data sets

Table 2 presents the comparative results where NN-TT1 and NN-TT2 are our Hopfield neural networks solving formulations TT1 and TT2 respectively. SA is the simulated annealing approach of Abramson and Dang [2] implemented using their hardware accelerator. With this method, triplets are mapped to periods and, as triplets are removed or added to the timetable, the change in cost is used to determine the next transition. The results reported in Table 2 are based on a set of 20 runs, and record the cost of the best solution found, the average cost of the 20 runs, and the average CPU time in seconds over the 20 runs. For the neural networks, 500 descents of 10 iterations were used with a threshold of 0.85, and $\alpha = 3, \beta = \chi = \gamma = 1$. The neural networks results were run on a 333MHz Pentium II with 128MB RAM, while SA was run using the hardware accelerator of Abramson and Dang [2].

method		hdt5	hdt6	hdt7	hdt8
NN-TT1	best cost	0	0	0	0
	average cost	0.5	0.8	1.1	1.4
	av CPU (secs.)	115.9	290.9	735.7	1465.1
NN-TT2	best cost	0	0	0	0
	average cost	0.5	0.7	1.0	1.2
	av CPU (secs.)	146.3	226.9	323.7	431.3
SA	best cost	0	0	2	2
	average cost	0.67	2.5	2.5	3.83
	av CPU (secs.)	72.45	80.22	6635.00	7859.00

Table 2: Results for two neural network formulations and simulated annealing on timetabling data sets

CONCLUSIONS

The results presented in Table 2 demonstrate clearly that both neural network methods outperform simulated annealing in terms of solution quality and speed. Furthermore, NN-TT2 scales better than NN-TT1. By following the approach of Hopfield and Tank, a less effective formulation is reached. Thus, the message of this paper is that researchers should be sure to consider alternative formulations of problems that can improve the chance of the Hopfield network competing with other techniques.

References

- [1] S. Even, A. Itai and A. Shamir, "On the complexity of timetable and multicommodity flow problem", *SIAM Journal of Computing*, vol. 5, no. 4, pp. 691-703, 1976.
- [2] D. Abramson and H. Dang, "School timetables: a case study in simulated annealing", in *Applied Simulated Annealing, Lecture Notes in Economics and Mathematics Systems*, edited by V. Vidal, Springer-Verlag: Berlin, Chapter 5, pp. 103-124, 1993.
- [3] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems", *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.
- [4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings National Academy of Sciences*, vol. 79, pp. 2554-2558, 1982.
- [5] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neuron", *Proceedings National Academy of Sciences*, vol. 81, pp. 3088-3092, 1984.
- [6] K. A. Smith, "Neural networks for combinatorial optimisation: a review of more than a decade of research", *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15-34, 1999.