SCCS report: 782

Mean Field Annealing and non-Euclidean Combinatorial Optimization Problems: The case of Academic Course scheduling, (Part I: Algorithms, Part II: Implementations)

Saleh Elmohamed
Geoffrey Fox
Paul Coddington
Dept. of Computer Science, and
Northeast Parallel Architectures Center,
Syracuse University
Syracuse, NY 13244
{saleh, gcf, paulc}@npac.syr.edu

SCCS report: 782

Mean Field Annealing and non-Euclidean Combinatorial Optimization Problems: The case of Academic Course Scheduling, (Part I: Algorithms, Part II: Implementations)

Saleh Elmohamed, Geoffrey Fox, and Paul Coddington

Northeast Parallel Architectures Center
Syracuse University, Syracuse, NY 13244

{saleh, gcf, paulc}@npac.syr.edu

ABSTRACT

In this study we have tackled the problem of academic course scheduling (or timetabling) at the university level using the method of mean field annealing. We also extended the Potts normalization conditions and equality constraints to accommodate our study case. Experimental analysis as well as summary of the results were included.

Key words : feed-back neural nets, neurons, mean field annealing, mean field theory, tanh functions, Potts systems, constraints, events, cost functions, phase transition.

1 Introduction

The prime objective of this study is to derive an approximate solution to a non-geometric combinatorial optimization (CO) problem, which is academic scheduling at the university level. It can be summarized as follows: given data sets of classes and their days, enrollments, and instructors; rooms and their capacities, types, and locations; distances between buildings; priorities of each building for different departments; and students and their class preferences; the problem is to construct a feasible class schedule satisfying all the hard constraints and minimizing the medium and soft constraints. Hard constraints are space and time constraints that must be satisfied, such as scheduling only one class at a time for any teacher, student, or classroom. Medium and soft constraints are student and teacher preferences that should be satisfied if possible.

The timetabling problem (TTP) is a high-dimensional, non-Euclidean, multi-constraint combinatorial optimization problem, and is consequently very difficult to solve. This problem has been tackled by many researchers, mostly in the field of operations research. A number of different heuristics have been tried on different instances of the problem, from high school to university course scheduling (see the reviews by de Werra [7] and Shaerf [25]) using method such as genetic algorithms [5], simulated annealing [1, 6, 3], tabu search [16], neural networks [18], and various combinations of two or more of these techniques. For small to medium size problems, some of these methods work well, however no particular method has yet been shown to produce good results for large-scale, real-world problems. Very few of these studies have used real-world data, and none of these methods has been used on as large and complex a problem as we have addressed in this work. Also, we are not aware of any large scale study that takes into account constraints due to student preferences.

We have used data for classes at Syracuse University. Currently this problem is handled by the university scheduling department in a semi-automated fashion. A scheduling program is used to find a partial solution, and substantial manual effort is required to iterate towards a final solution. Also, when scheduling a certain semester (e.g. fall 1996), a template of a previous semester (e.g. fall 1995) is used as part of the input data.

We have applied a method consisting of a feed-back artificial neural system (ANS) composed of multi-state neurons using mean field theory (MFT) equations to control the dynamics. This method is usually known as mean field annealing (MFA). The MFT equations are used in place of the computationally demanding stochastic updating procedures often used to avoid getting stuck in local minima.

Much of the work in the neural networks field dealing with optimization problems has centered on understanding and improving on the Hopfield-Tank TSP algorithm [18] (HT, for short). HT proposed that CO problems could be solved using such analog networks. In practice however, considerable difficulties have been encountered carrying out this program for the TSP, leave alone problems such as scheduling.

2 The Timetabling Problem

Timetabling is the assignment of time slots to a set of events, subject to constraints on these assignments. The NP-complete *professors and classes* timetabling problem [1, 6, 13, 14] is a constraint satisfaction problem that can be briefly stated as follows:

For a certain school with N_p professors, N_q classes, N_x classrooms and lecture halls, and N_s students, it is required to schedule N_l professor-class pairs within a time limit of N_t time slots producing a legal schedule. A legal schedule needs to be found such that no professor, class, or student is in more than one place at a time, and no room is expected to accommodate more than one lesson at a time or more students than its capacity. Figure 1 sketches the approximate relationship between the various entities of the timetabling problem.

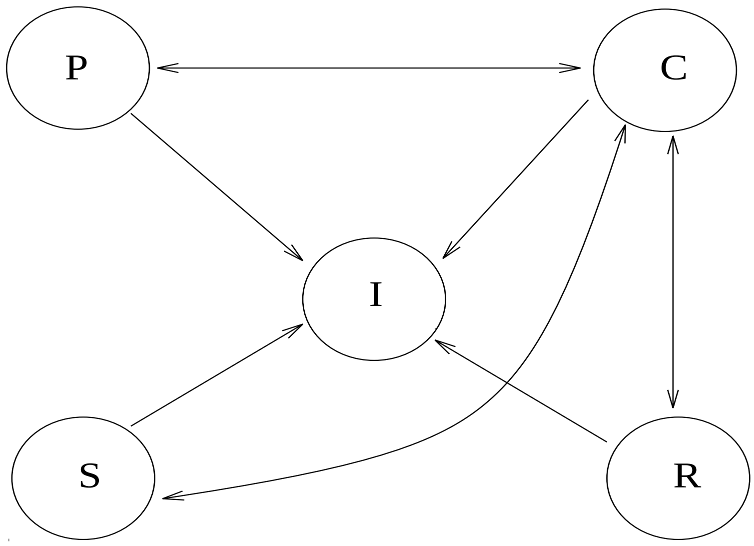

Figure 1: The relationship between various entities of TTP where P: professors, C: classes, S: students, R: rooms, and I: time periods. The direction of the arrow is the direction of the assignment.

The constraints for this problem can be hard, medium or soft. The medium and soft constraints have an associated cost (or penalty), and if they are not satisfied, the goal is to minimize this cost. Soft constraints have a lower priority (and thus lower cost) than medium constraints. The hard constraints must be satisfied, so their associated cost must be reduced to zero. A feasible schedule is one that satisfies all the hard constraints.

Hard constraints are usually constraints that physically cannot be violated. This includes events that must not overlap in time, such as:

- classes taught by the same professor,
- classes held in the same room,

- a class and a recitation or a lab of the same class.

Another examples are space or room constraints:

- A class cannot be assigned to a particular room unless the capacity of the room is greater than or equal to the class enrollment.

- Some classes, such as laboratories, require a certain type of room.

Medium constraints are usually considered to those constraints that fall into the gray area between the hard and soft constraints [8]. In our implementation, we define medium constraints to be constraints such as time and space conflicts which, like hard constraints, cannot physically be violated (for example, it is not possible for one person to be in two different classes at the same time). However we consider these constraints to be medium rather than hard if they can be avoided by making adjustments to the specification of the problem. The primary example is student preferences. We cannot expect to be able to satisfy all student class preferences, in some cases, certain students will have to adjust their preferences since certain classes will clash, or will be oversubscribed.

Medium constraints have a high penalty attached to them, although not as high as that associated with the hard constraints. In the final schedule the penalty of these constraints should be minimized and preferably reduced to zero. Some examples of medium constraints are:

- Avoid time conflicts for classes with students in common.

- Eligibility criteria for the class must be met.

- Do not enroll athletes in classes that conflict with their sport practice time (of course, depending on the sport).

Soft constraints are preferences that do not deal with time conflicts, and have a lower penalty (or cost) associated with them. We aim to minimize the cost, but do not expect to be able to reduce it to zero. Some examples are:

- For each student, balance the three-day (*Mon, Wed, Fri*) as well as the two-day (*Tue, Thu*) schedules.

- Balance or spread out the lectures over the week.

- Classes may request contiguous time slots.

- Balance enrollment in multi-section classes.

- Lunch and other break times may be specified.

- Professors may request periods in which their classes are not taught.

- Professors may have preferences for specific rooms or types of rooms.

- Minimize the distance between the room where the class is assigned and the building housing its home department.

Some soft constraints may have higher priority (and thus higher cost) than others. For example, preferences involving teachers will have higher priority than the preferences of students.

The **cost function** measures the quality of the current schedule and generally involves the weighted sum of penalties associated with different types of constraint violations. The aim of the optimization technique is to minimize the cost function.

3 Brief Notes on the Complexity of Timetabling

Timetabling belongs to the class of constraint satisfaction problems and it is essentially an example of a resource-constrained scheduling problem, which is an NP-hard [11, 4]. Here, resources are physical entities, for example: students, rooms, overhead projectors, and so on. Also time periods can be thought of as resources.

In general, timetabling problems are always NP-complete [10], which means there exists no known polynomially bound algorithm for solving them optimally. As a result, these problems are often solved by means of *heuristics* – solution procedures that focus on finding a feasible schedule of "good" (as opposed to optimal) quality within an acceptable amount of time.

Traditionally, timetabling has been approached by means of linear programming (LP) with binary variables plus some heuristics. For example, if i identifies a teacher, j identifies a time interval and k identifies a class then the binary variable $X_{ijk} = 1$ if teacher i has class k at time-interval j; $X_{ijk} = 0$, otherwise. Suppose we have a set of data as small as 20 teachers, 30 time intervals, and 10 classes. If we use the LP approach to obtain a feasible schedule out of this data, then we need to deal with over 6000 variables to represent the problem, and 2^{6000} possible states. The problem quickly becomes intractable as the number of variables increases.

For the much simpler problem of *class allocation*, for which there is no timetabling involved (i.e. no allocation of classes to time periods), the computational load (n_c) from tackling this problem using the LP method scales as follows [15, 14]:

$$n_c \propto N_c N_x N_t \quad , \tag{1}$$

where N_c is the number of room categories available (i.e. classroom, auditorium, etc.).

4 Mean-Field Annealing (MFA)

One of the potential drawbacks of using methods such as simulated annealing for hard optimization problems is that finding a good solution can often take an unacceptably long time. Mean-field annealing attempts to avoid this problem by using a deterministic approximation to simulated annealing, by attempting to average over the statistics of the annealing process. The result is improved execution speed at the expense of solution quality. Although not strictly a continuous descent technique, MFA is closely related to the Hopfield neural network [18].

MFA has been successfully applied to high-school class scheduling [14]. For scheduling, it is advantageous to use a Potts neural encoding to specify discrete neural variables (or neurons) for the problem. This is defined in its simplest form as a mapping of events onto space-time slots, for example events defined by a professor-class pair (p, q) are mapped onto space-time slots (x, t). Potts neurons $s_{pq,xt}$ are defined to be 1 if event (p, q) takes place in the space-time slot (x, t), and 0 otherwise. In this way, the constraints involved can be embedded in the neural net in terms of a Potts normalization condition such as $\sum_{x,t} S_{pq,xt} = 1$.

For a full derivation of the mean-field annealing algorithm from its roots in statistical physics, see Peterson *et al.* [23]. Here, we will summarize the algorithm as well as Hopfield's model.

4.1 Some Background on the Hopfield Model

The network is built by connecting a large number of neurons to each other to form a dense, parallel array and, in general, any i^{th} neuron in the net is described by two variables: its current state (input) u_i, and its output v_i. The output is related to the input by a monotonically increasing transfer function: $v_i = g_\lambda(u_i)$. For large λ, $g_\lambda(u_i) = \frac{1}{1+e^{-2\lambda u_i}}$ approximates a step function, forcing v_i to $[0, 1] \implies$ (the output of the net lies on the unit hypercube).

4.1.1 Operation of the Network

The network (see Figure 2) operates as follows: Its dynamics are governed by a set of nonlinear coupled differential equations of the form:

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} W_{ij} v_j + I_i^b \ . \tag{2}$$

or the general form of it:

$$\frac{du}{dt} = -\eta u + Wv + I^b \ . \tag{3}$$

where W_{ij} is the connection strength between neuron i and neuron j, I_i^b is an input bias to neuron i, and the η term introduces an element of decay into the network's dynamics, which can be used to ensure that $\mathbf{u}$ variables remain bounded in magnitude.

4.1.2 Updating of the States of the Neurons

To make sure that stability of the of the net is guaranteed, u_i is updated continuously by evaluating $\frac{d}{dt}u_i$ as a function of the total input to the i^{th} neuron.

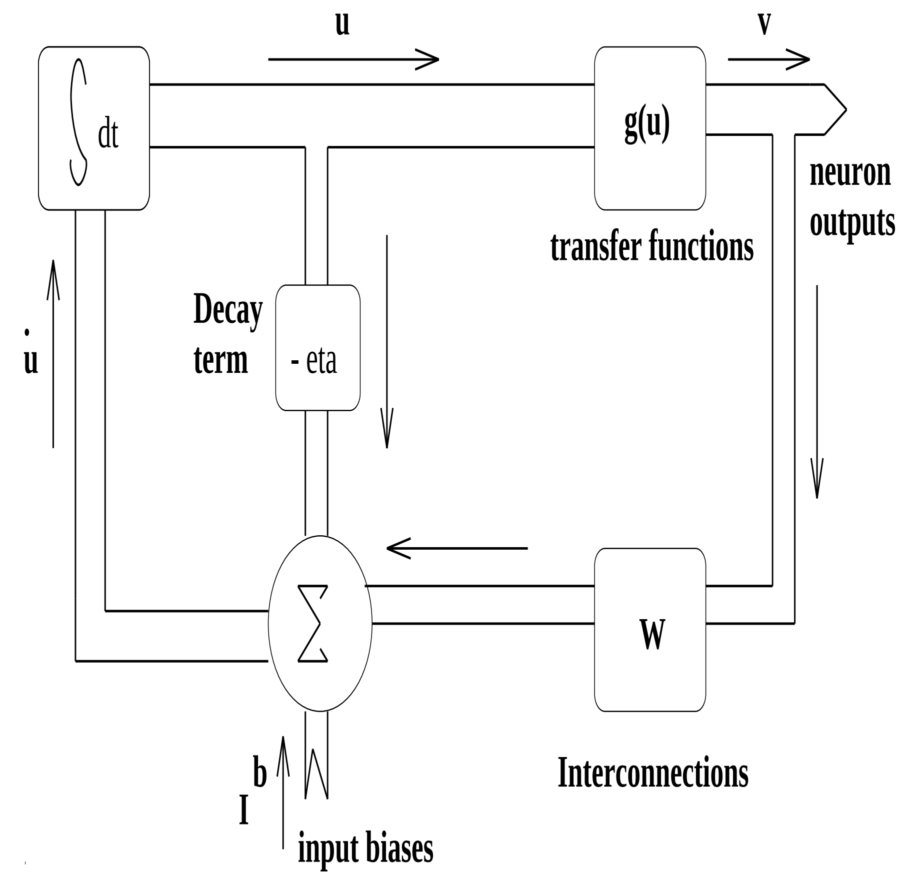

Figure 2: A schematic view of the Hopfield network.

$$\dot{u} = -\eta u + Wv + I^b \ , \tag{4}$$

where

$$v = g(u) \ . \tag{5}$$

Equations 4 and 5 govern the dynamics of the network.

4.1.3 Stability of the Net

Given that W is symmetric, Hopfield proved stability for the network by showing that with the above dynamic equations (4 and 5) the network output v evolves so as to minimize the *Liapunov function*:

$$E^{liap}(v) = \left(-\frac{1}{2} \sum_{ij} W_{ij} v_i v_j - \sum_i I_i^b v_i \right) + \eta \sum_i \int_0^{v_i} g^{-1}(V) \, dV \ . \tag{6}$$

$v = (v_1 \dots v_n)$ is denoting the net state, therefore we have an energy function defined throughout $v \in [0,1]^n$ and dynamics which guarantee convergence to minima. For a suitable W, the minima of the first term lie at the hypercube vertices. The second term is minimized at the hypercube center but has negligible impact at low η.

4.1.4 Updating the States of the Net

Updating of the states (or the output of the neurons) is carried out either asynchronously or synchronously. From a simulation point of view, we found that synchronous updating a bit more efficient. From a theoretical point of view, it makes not much of a difference on the operation of the network as to which updating mode is used.

4.1.5 Mapping the Problem onto the Network

First, a representation of the problem is needed in which the feasible solutions lie at hypercube vertices. Each hypercube vertex is regarded as a configuration of the problem, with an associated energy (or cost) function. The partition function is then formed by summing the Boltzmann factors of the admissible configurations. The main part to derive the solutions is a quadratic energy function (to be mapped onto the first term of 6) for which the minima correspond to solutions and the depth of each minimum reflects the solution quality.

4.1.6 Shortcomings of the Hopfield Model

The network's partition function sums over a vast number of configurations (e.g. for the TSP, all of the 2^{N^2} configurations are admissible). These configurations, again, for the TSP, are nothing like valid tours, and even though these offenders have small Boltzmann weights their large number inevitably affects the thermal average quantities.

5 The Potts Approach to MFA for Timetabling

Clearly the original HT algorithm had difficulty constraining the network into valid solutions. One way forward is therefore to lessen the computational burden being placed on the network. This can be done by constraining each city (in the TSP case) to be "on" only once, by enforcing $\sum_a V_{ia} = 1 \; \forall i$, ($V_{ia} = 1/0$ means that the city i is (is not) the a^{th} city visited in the tour), rather than relying on an energy penalty term to try and do this (as in HT). This idea was first utilized and analyzed for the TSP case by, among others, Peterson *et al.* [23]. It greatly enhances the degree of convergence to valid solutions. Later, Peterson and colleagues somewhat extended the technique and applied it to a case of high school scheduling [13, 14].

In [23] the normalization occurs in the context of a *mean field annealing* approach. This generally yields the same final network equations as the "neuronal circuit" approach of [19], but the exposition is a little bit clearer, as it is laid out in a statistical mechanics framework*. The idea is to regard each hypercube vertex (see [19]) as a configuration, with an associated energy (or cost) E. The partition function is then formed by summing the Boltzmann factors of the admissible configurations. For example, in the HT algorithm, for the TSP, as previously noted all of the 2^{N^2} configurations are admissible, whereas if each city is restricted to being visited only once, then only N^N vertices are admissible. After taking a mean field approximation, saddle point equations are derived (see next sections), the solutions of which to pick out the dominant states of the network at the current temperature T.

This statistical mechanics characterization gives a somewhat clearer picture of why the normalized model (which corresponds to a type of Potts model [32] in physics, in that each city represented by a spin which can be in only one of N states) ought to perform better than HT model. As Peterson *et al.* [23] indicated, the HT algorithm's partition function sums over a vast number of configurations which are nothing like valid tours in the TSP, and even though these offenders have small Boltzmann weights their large number inevitably affects the thermal average quantities. The other advantage that draws us to this derivation is that, by introducing a temperature, the concept of annealing, i.e. reducing T during a run, can be justifiably employed.

The above two paragraphs outline more or less the reasons behind our choice to use MFA.

5.1 Constraint Formulations in the Potts Model

In this section we define some key terms and concepts used in this paper and following the notation of Peterson and colleagues on this subject.

Let (N_p) to be a number of professors that give (N_q) classes in (N_x) class-rooms at (N_t) time slots. Event-wise, (p,q) event – denoting a professor p giving a class q – takes place in the space-time slot (x,t) – denoting classroom x at time t. Let $S_{pq;xt}$ denote the Potts (or multi-state) neuron, where

$$S_{pq;xt} = \begin{cases} 1 & \text{if event } (p,q) \text{ takes place in } (x,t) \ , \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Now, we define a number of **hard constraints** as follows: It is known that an event (p,q) should occupy precisely one space-time slot (x,t) so its associated constraint can be embedded in the neural net in terms of the Potts normalization condition:

$$\sum_{x,t} S_{pq,xt} = 1 \tag{8}$$

Different events (p_1,q_1) and (p_2,q_2) should not occupy the same space-time slot (x,t).

$$E_{XT} = \frac{1}{2} \sum_{x,t} \sum_{p_1,q_1} \sum_{p_2,q_2} S_{p_1q_1;xt} S_{p_2q_2;xt} = \frac{1}{2} \sum_{x,t} \left(\sum_{pq} S_{pq;xt} \right)^2 \tag{9}$$

*This equivalence was also noted in [18]. P. Simic and Martin Simmen brought this point to our attention.

A teacher p should have at most one class at a time.

$$E_{PT} = \frac{1}{2} \sum_{p,t} \sum_{q_1,x_1} \sum_{q_2,x_2} S_{pq_1;x_1t} S_{pq_2;x_2t} = \frac{1}{2} \sum_{p,t} \left(\sum_{qx} S_{pq;xt} \right)^2 \tag{10}$$

A class q should have at most one teacher at a time.

$$E_{QT} = \frac{1}{2} \sum_{q,t} \sum_{p_1,x_1} \sum_{p_2,x_2} S_{p1q;x_1t} S_{p2q;x_2t} = \frac{1}{2} \sum_{q,t} \left(\sum_{px} S_{pq;xt} \right)^2 \tag{11}$$

In [18, 19], Hopfield introduced a model with neurons having a graded response. So given I_i a fixed input bias to neuron x_i, W_{ij} the weight of the connection between neurons (x_i, x_j), and parameter T to denote the temperature, the behavior of x_i can be stated as:

$$x_i = \frac{1}{2} + \frac{1}{2} \cdot tanh \left((\sum_{i \neq j} W_{ij} x_j - I_i) \times \frac{1}{T} \right) \tag{12}$$

When $T \longrightarrow 0$, function (12)'s response takes on a step-wise form. These models of [19] are known to be a bit sensitive to parameter tuning for real world applications, especially those of the type that involves "finding" or "searching for" 1 out-of N type problems.

At a finite T, x_i (of eq. 12) can be replaced by an MFT variable υ_{ia}, then differentiating to obtain the Potts MFT equations:

$$\upsilon_{ia} = -\frac{1}{T} \cdot \frac{\partial E[\vec{\nu}]}{\partial \nu_{ia}} \tag{13}$$

$$\nu_{ia} = \frac{e^{\upsilon_{ia}}}{\sum_b e^{\upsilon_{ib}}} \tag{14}$$

Which enforces

$$\sum_a \nu_{ia} = 1 \quad \forall i \,. \tag{15}$$

Where a denotes the vector components or all the elements of the group. Also, we could state that $\nu_{ia} \approx$ Probability(neuron(i) $\in$ state(a)). Equations (13, 14) are computed in an iterative manner.

In [23, 31], it was stated that the performance of MFA on certain problems can be improved using *neuron normalization*. This idea was originally inspired by an analogy with Potts glasses in statistical physics. *Neuron normalization* modifies the MFA algorithm so that constraints of the form $\sum_{a \in \text{ States}} \nu_{ia} = 1$ are explicitly enforced, without the need for a penalty term. It was found by Peterson and other to be particularly useful for mapping the TSP and graph partitioning (GP) type problems.

Now, to carry out the minimization of the energy functions stated above, we need to convert the neural variables $\{0, 1\}$ of S_i to their corresponding *mean field variables* at T. These correspondences are derived (see [14]) to be:

$$V_{pq;xt} \approx < S_{pq;xt} >_T \ ; \tag{16}$$

which is the thermal average of the original binary spines, with the following multi-state MFT equations (see (13, 14) for a general format):

$$U_{pq;xt} = -\frac{1}{T} \cdot \frac{\partial E}{\partial V_{pq;xt}} \tag{17}$$

$$V_{pq;xt} = \frac{e^{U_{pq;xt}}}{\sum_{x't'} e^{U_{pq;x't'}}} \tag{18}$$

The MFT equations of υ and ν are solved *iteratively* either synchronously or asynchronously, under T. Also, these equations give rise to 2 phase transitions; one in x (space) and the other in t (time). This leads to factorization of the system by replacing $S_{pq;xt}$ by X**-neurons** $S^{(X)}_{pq;x}$ and T**-neurons** $S^{(T)}_{pq;t}$:

$$S_{pq;xt} = S^{(X)}_{pq;x} \cdot S^{(T)}_{pq;t} \tag{19}$$

With separate Potts conditions:

$$\sum_x S^{(X)}_{pq;x} = 1 \tag{20}$$

$$\sum_t S^{(T)}_{pq;t} = 1 \tag{21}$$

For X**-neurons** and T**-neurons**, respectively. Rather than dealing with p and q directly, an independent variable (i) -acting as an event index- to which p and q are attributes $p(i)$ and $q(i)$. The value of these exits in a look-up table which contains all the relevant information to process each event (i). Now, the above (20, 21) conditions can be re-stated as follows:

$$\sum_x S^{(X)}_{i;x} = 1 \tag{22}$$

$$\sum_t S^{(T)}_{i;t} = 1 \tag{23}$$

During annealing, as we go from a higher temperature to a lower one, a *phase transition* is passed at $(T = T_c)$, where T_c† is the *critical temperature*. The value of T_c set the scale or the range of the value of T, and it is estimated by expanding 18 around some trivial fixed-point such as

$$V^{(0)}_{pq;xt} = \frac{1}{N_x N_t} \quad \text{, at } T = 0 \ . \tag{24}$$

Also T_c depends on the self-coupling coefficient β, and on the method of the update. This parameter β acts as a Lagrangian multiplier for the given energy constraint(s). See section on the generic Potts algorithm and [23].

†See pp 29-31, [17] for an approximate value in the case of Ising ferromagnets.

During the revision process of an existing schedule, sometimes it is desirable to have a certain cluster of events to stick together in time (or within some specified period of time). This is generally referred to as *relative clamping* and amounts to making the following notational replacement:

$$i \longrightarrow (j,k) \quad ; \tag{25}$$

where (j) denotes the cluster and (k) an event within (j). In this situation, one has a common t-neuron ($S^{(T)}_{j;t}$) for cluster j but distinct x-neurons ($S^{(X)}_{jk;x}$) for the individual events.

To deal with events that take place over two consecutive hours, what is called effective *time*-neurons $\tilde{S}^{(T)}_{i;t}$ are introduced:

$$\tilde{S}^{(T)}_{i;t} = \sum_{k=0}^{g_i-1} S^{(T)}_{i;t-k} \quad ; \tag{26}$$

where $g_i = 1$ for single hours and $= 2$ for double hours. For those events that occur between one and two hours, g_i can be adjusted accordingly to handle these cases. Using these effective *time*-neurons, constraints (9, 10, 11) can be put into the following forms:

$$E_{XT} = \frac{1}{2} \sum_{x,t} \sum_{i \neq i'} S^{(X)}_{i;x} S^{(X)}_{i';x} \tilde{S}^{(T)}_{i;t} \tilde{S}^{(T)}_{i';t} \tag{27}$$

$$E_{PT} = \frac{1}{2} \sum_{p,t} \sum_{i \neq i'} \delta_{p(i),p(i')} \tilde{S}^{(T)}_{i;t} \tilde{S}^{(T)}_{i';t} \tag{28}$$

$$E_{QT} = \frac{1}{2} \sum_{q,t} \sum_{i \neq i'} \delta_{q(i),q(i')} \tilde{S}^{(T)}_{i;t} \tilde{S}^{(T)}_{i';t} \tag{29}$$

The Kronecker $\delta_{q(i),q(i')}$ ensures that only events i and i' with identical "primordial" classes (or point of origination) are, for example, summed over. Here, $q(i)$ and $q(i')$ refer to classes of the same type or 2 sections derived from the same original class.

$$\delta_{r,s} = \begin{cases} 1 & \text{if } r = s \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

One important soft constraint that need to be handled is *group formation*. Briefly, it can be stated as follows: for some optional subjects the classes are broken up into option groups temporarily forming new classes. In order to account for this, we need a formalism that allows for the breaking up of these primordial classes and subsequent recombinations into quasi-classes. To carry out such an extension one might have contributing pairs of events where the primordial classes are different. Hence one should replace δ with a more structured overlap matrix Γ which is given by a look-up table.

$$\delta_{q(i),q(i')} \longrightarrow \Gamma_{q(i),q(i')} \ . \tag{31}$$

Using relative clamping and form 31, we need to make sure that $(\forall k)$ events within a given (j) cluster $p(j,k) \neq 0$. This says that there can be no "uncovered" class period or a class event (that is, periods when no professor has been assigned to an already scheduled class). For this purpose we introduce $\tilde{\delta}_{j,k}$ according to:

$$\tilde{\delta}_{j,k} = \delta_{j,k}(1 - \delta_{j,0}\delta_{k,0}) \ . \tag{32}$$

If $j = 0$ then $\forall k$, $k = 0$.

Using the abovementioned concepts, the collision terms of equations (27, 28, and 29) are extended to ensure that all the hard constraints are satisfied (or the solutions obtained are legal). The extension takes the following forms:

$$E_{XT} \text{ (form 1) } = \frac{1}{2} \sum_{jk \neq j'k'} \sum_{x} S^{(X)}_{jk;x} S^{(X)}_{j'k';x} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} \tag{33}$$

$$E_{XT} \text{ (form 2) } = \frac{1}{2} \sum_{j \neq j'} \sum_{kk'} \sum_{x} S^{(X)}_{jk;x} S^{(X)}_{j'k';x} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} + \frac{1}{2} \sum_{j} g_j \sum_{k \neq k'} \sum_{x} S^{(X)}_{jk;x} S^{(X)}_{jk';x} \tag{34}$$

$$E_{PT} \text{ (form 1) } = \frac{1}{2} \sum_{p,t} \left[\sum_{jk} \tilde{\delta}_{p,p(j,k)} \tilde{S}^{(T)}_{j;t} \right] \left[\sum_{j' \neq j} \sum_{k'} \tilde{\delta}_{p,p(j',k')} \tilde{S}^{(T)}_{j';t} \right] \tag{35}$$

$$E_{PT} \text{ (form 2) } = \frac{1}{2} \sum_{j \neq j'} \sum_{kk'} \tilde{\delta}_{p(j,k),p(j',k')} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} \tag{36}$$

$$E_{QT} \text{ (form 1) } = \frac{1}{2} \sum_{qt} \left[\sum_{jk} \delta_{q,q(j,k)} \tilde{S}^{(T)}_{j;t} \right] \left[\sum_{j' \neq j} \sum_{k'} \Gamma_{q,q(j',k')} \tilde{S}^{(T)}_{j';t} \right] \tag{37}$$

$$E_{QT} \text{ (form 2) } = \frac{1}{2} \sum_{j \neq j'} \sum_{kk'} \Gamma_{q(j,k),q(j',k')} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} \tag{38}$$

Note: Each (form 2) above is a derivation from its corresponding (form 1).

The factorization idea shown in equations (33, 34, 35, 36, 37, and 38) can be extended to *mean field* variables $V^{(X)}_{jk;x}$ and $V^{(T)}_{j;t}$, corresponding to neurons $(S^{(X)}_{jk;x})$ and $(S^{(T)}_{j;t})$, respectively. $V^{(X)}_{jk;x}$ and $V^{(T)}_{j;t}$ can be interpreted as being the probabilities that the corresponding events occur at given x-value and t-value with normalization given by the equations (22, 23) .

Substituting V for S in equations (22, 23), the *mean field* equations at a finite T, are given in terms of the *local fields* $U^{(X)}_{jk;x}$ and $U^{(T)}_{j;t}$. Now, the *mean field* dynamics proceed over the following local and mean field variables:

$$U^{(X)}_{jk;x} = -\frac{1}{T} \cdot \frac{\partial E}{\partial V^{(X)}_{jk;x}} \tag{39}$$

$$U_{j;x}^{(T)} = -\frac{1}{T} \cdot \frac{\partial E}{\partial V_{j;t}^{(T)}} \tag{40}$$

$$V_{jk;x}^{(X)} = \frac{e^{U_{jk;x}^{(X)}}}{\sum_{x'} e^{U_{jk;x'}^{(X)}}} \tag{41}$$

$$V_{j;t}^{(T)} = \frac{e^{U_{j;t}^{(T)}}}{\sum_{t'} e^{U_{j;t'}^{(T)}}} \tag{42}$$

The iterations of the *MF* dynamics over equations (39 to 42) can either be **synchronous** or **serial** updating. Synchronous updating involves a single sweep consists of first computing all the local fields, and then updating all the neurons. That means first perform equations (39,40) then perform equations (41,42). While serial updating is done as follows: for each neuron $S_{jk;x}^{(X)}$ and/or $S_{j;t}^{(T)}$, the *local field* is computed immediately before the corresponding *neuron state* is updated. That means either do equation (39) followed by (41) and/or do equation (40) followed by (42).

Next, we need to formalize two important **soft constraints** encountered when scheduling classes.

Spreading: different classes in a particular area (i.e. computer science) should be spread over the week in a (MWF) and (TuTh) type of spread. First, assume we have access to a subject attribute $s(j,k)$. Second, we define an effective "day-neuron" as follows ($\tilde{S}_{j;dh}^{(T)}$ is reduced to a D-type neuron by summing over all hours of the day):

$$S_{i;d}^{(D)} = \sum_h \tilde{S}_{j;dh}^{(T)} = g_j \cdot S_{j;dh}^{(T)} \tag{43}$$

So the penalty term E_{QSD} that spreads classes Q in a particular subject S over different week-days D :

$$E_{QSD} = \frac{1}{2} \sum_{qsd} \left[\sum_{jk} \sum_h \delta_{q,q(j,k)} \delta_{s,s(j,k)} \tilde{S}_{jk;dh}^{(T)} \right]^2 \tag{44}$$

Refining the above a bit more by considering 2 different clusters (j) and $(\hat{j})$ of classes over D : (q and s are not used in the summation, since no change occurs to them)

$$E_{QSD} = \frac{1}{2} \sum_d \sum_{j \neq \hat{j}} \sum_{kh} \sum_{\hat{k}\hat{h}} \delta_{q(j,k),q(\hat{j},\hat{k})} \delta_{s(j,k),s(\hat{j},\hat{k})} \hat{S}_{j;dh}^{(T)} \hat{S}_{j;d\hat{h}}^{(T)} \tag{45}$$

Now for some day (d) of the week:

$$E_{QSD} = \frac{1}{2} \sum_{j \neq \hat{j}} \sum_{k\hat{k}} \delta_{q(j,k),q(\hat{j},\hat{k})} \delta_{s(j,k),s(\hat{j},\hat{k})} \sum_d S_{j,d}^{(D)} S_{\hat{j},d}^{(D)} \tag{46}$$

Gluing: The schedule should have as few "holes" as possible; that is lessons should be glued together. We would like to point out that this constraint operates directly opposite to *spreading*, as a result when *spreading* is present we have not dealt with *gluing* and vice versa.

The penalty term that rewards situations where (d,h)-events are glued to $(d,h-1)$-events can be defined as follows:

$$E_{QDH} = -\alpha \sum_{q} \sum_{d,h} \left[\sum_{j,k} \delta_{q,q(j,k)} \tilde{S}^{(T)}_{j;dh} \right] \left[\sum_{\hat{j} \neq j} \Gamma_{q,q(\hat{j},\hat{k})} \tilde{S}^{(T)}_{\hat{j};d(h-1)} \right] \tag{47}$$

The energy term of (47) can be summarized to say that: *if cluster (j) with events each symbolized by (k) occurs on hour (h) of day (d) then it would be quite desirable to have another event $(\hat{k})$ from a different cluster $(\hat{j})$ to occur on hour $(h-1)$ of that day.*

Next, since (q) is fixed we could re-write the above equation as follows:

$$E_{QDH} = -\alpha \sum_{\hat{j} \neq j} \sum_{k,\hat{k}} \Gamma_{q(j,k),q(\hat{j},\hat{k})} \sum_{d,h>1} \tilde{S}^{(T)}_{j;dh} \tilde{S}^{(T)}_{\hat{j},d(h-1)} \tag{48}$$

The presence of the $(-)$ sign in front of (47 and 48) indicates that such energy functions are to be minimized. Also, the (α) in front of the the above forms governs the strength of this reward relative to the energies of equations (9,10,11). This (α) takes on a value between zero and 1.

Finding *legal* solutions with no collisions corresponds to minimizing the *hard* energy:

$$E_{hard} = combine(E_{XT}, E_{PT}, E_{QT}) \tag{49}$$

to zero. The *soft energy* is given by:

$$E_{soft} = combine(E_{QSD}, E_{QDH}) \ . \tag{50}$$

The total energy E to be minimized is then:

$$E = combine(E_{hard}, E_{medium}, E_{soft}) \ . \tag{51}$$

5.2 The generic Potts Algorithm

The generic MFA algorithm appears in Figure 3. Let (N) denotes the total number of the utilized neurons. At high T, the mean field solutions will tend to be states near to the (symmetrical) maximum entropy state: $\forall(i,a) v_{ia} = 1/N$. Conversely at low T, finding a mean field solution will be equivalent to using a local optimization method on the internal energy — a procedure highly sensitive to the initial conditions and known to be ineffective (see p. 149 of [18]).

These characteristics are similar to those of simulated annealing, which is no surprise since both it and the mean field method compute thermal averages over Gibbs distributions of discrete states, the former stochastically and the later through a deterministic approximation. It is therefore natural to couple the mean field method with the concept of annealing from high to low temperatures. The resulting MFA method can be a powerful and general technique which has found many applications in combinatorial optimization.

1. Choose a problem $\Longrightarrow$ (mapped into) $\{w_{ij}\}$.
2. Find the approximate phase transition temperature by linearizing equation (14).
3. Add a self-coupling β-term if necessary. Neural-wise, this corresponds to a feedback connection from a neuron to itself.
4. Initialize the neurons ν_{ia} with $\frac{1}{N}\pm$ *random values*; and set $T(0) = T_c$.
5. Until $(\Sigma \approx \frac{1}{N}\sum_{ia}\nu_{ia}^2 \geq 0.99)$ do:
   - At each $T(n)$, update all ν_{ia} via υ_{ia}:
     - $\upsilon_{ia} = -\frac{1}{T(n)} \cdot \frac{\partial E[\nu]}{\partial \nu_{ia}}$.
     - $\nu_{ia} = \frac{e^{\upsilon_{ia}}}{\sum_b e^{\upsilon_{ia}}}$. $;(\equiv f_{ia})$
   - $T(n+1) = 0.9 \times T(n)$.
6. After $\Sigma \geq 0.99$ is reached perform a *greedy heuristics* if needed to account for possible imbalances or rule violations.

Figure 3: The Generic Mean-Field Annealing Algorithm

In addition to the structure of the energy function, there are three major interdependent issues which arise in completely specifying a mean field annealing algorithm for a timetabling problem:

- The values of the coefficients of terms in the energy function.
- The types of dynamics used to find solutions of the mean field equations at each T.
- The annealing schedule details, i.e. the initial temperature $T(0)$, the rules for deciding when to reduce T and by how much, and the termination criteria.

A quantity introduced by [23] is called *saturation*, Σ, defined as

$$\Sigma = \frac{1}{N}\sum_{ia}\nu_{ia}^2 \quad , \tag{52}$$

to characterize the degree of clustering‡ of events in time and/or space; clearly $\Sigma_{min} = \frac{1}{N}$ and $\Sigma_{max} = 1$.

‡Other measures could also do this, e.g. an entropy measure $-\sum_{ia}\nu_{ia}ln(\nu_{ia})$, but Σ will be used here for consistency with the original work.

5.3 Some implementation details

The general idea is to initialize the system close to the symmetric state at a temperature near T_c, then, allowing the system to converge to a fixed point at each temperature, anneal (i.e. reduce T) until either the system reaches a stable high saturation state or some other termination criterion is satisfied. Specifically, the implementation used here has the following ingredients[§]: (Martin Simmen [30] investigated in depth these and most of the other issues pertaining to the Potts model, using the TSP as a test case.)

Here are 4 of the main issues involved:

Initial State: Another possible initial value of $\nu_{ia}^{n=0}$ is $(1+\xi_{ia})/N$, with the ξ_{ia} being random variables drawn uniformly from $[-\xi, \xi]$. Clearly ξ should be $\ll 1$.

Convergence: At each T the ν_{ia} variables are updated as above or by the following rule[¶]:

$$\nu_{ia}^{(n+1)} = \nu_{ia}^{(n)} + \gamma(f_{ia} - \nu_{ia}^{(n)}), \quad 0 < \gamma \leq 1 \ . \tag{53}$$

Setting $\gamma = 1$ recovers the dynamics of [23], and the $\gamma \to 0$ limit gives the first order Euler integration method for the differential equation:

$$d\nu_{ia}/dt = f_{ia} - \nu_{ia} \ . \tag{54}$$

The fixed points (also referred to as the free energy minima) of this equation constitute the mean field solutions. The degree of convergence is monitored through some sort of a "tolerance" quantity of the form $\sum_{ia} (\nu_{ia} - f_{ia})^2$.

Annealing Schedule: A simple exponential scheme is used, i.e. after reaching a fixed point at T, $T \mapsto T \cdot T_r$, where $T_r \in (0, 1)$ controls the cooling rate. Roughly speaking, T_r should be set to 0.9.

Termination: One criterion is as outlined above in the algorithm. The other two involve terminating runs which either fail to reach a fixed point within a certain number of sweeps at some temperature, or which allow T to fall below some T_{min}, ($T_{min} \ll T_c$), without ever satisfying the $\Sigma > \Sigma^{thresh}$ criterion. T_{min} was set to 10^{-3}.

5.4 More Constraints for the Potts Model

Let $q(i)$ represents a section of class q, $O_{q(i)}$ indicates that section $q(i)$ is open, and $C_{q(i)}$ is closed. Also, let $b(x, t)$ be a balanced interval for some time period (t), R is the overall set of students, and an arbitrary given subset is $r \subseteq R$.

Here, we need to introduce the R-neuron: $S^{(R)}_{r(i);(d(t)d(x)}$ where $k = 1, 2, \ldots$. Each student $r(i) \in R$ should have a balanced schedule over the two-group of days (MWF) and $(TuTh)$. We will also use r_s to denote a single student.

For X and T neurons, let $x(j)$ and $t(j)$ where $j = 1, 2, \ldots, n$ stand for places and time slots (or events), respectively. Using such notation, an example of the Potts normalization will be: $S^{(X)}_{p(i)q(i);x(j)}$ and $S^{(T)}_{p(i)q(i);t(j)}$ respectively, where $i = 1, 2, \ldots, n$ is a section index.

[§] These follow the method used in [23] but with some additions/improvements done by the authors -see algorithm above.

[¶] Thanks to M. Simmen for pointing this out. Also see [23]

The athletes case: Let $\vec{r} \subseteq R$ denotes the group of athletes that are dealt with in this study. Let (r^*,b) be playing/training events scheduled during the weekdays at (or during) times when academic (attending lectures) events (r^+,a) are also scheduled, or may possibly occur. Let ($d(x)$, $d(t)$) a place/time during the week at which any of these events can occur.

In addition to the above stated Potts neurons, we also will be using :

1. $S^{(\vec{r},X)}_{r^*b;d(x)}$, $S^{(\vec{r},T)}_{r^*b;d(t)}$: It takes a value of (1) if event (r^*, b) takes place in the place and time slot, respectively, of a weekday d, and zero otherwise.
2. $S^{(\vec{r},X)}_{r^+a;d(x)}$, $S^{(\vec{r},T)}_{r^+a;d(t)}$: It takes a value of (1) if event (r^+, a) takes place in the place and time slot, respectively, of a weekday d, and zero otherwise.
3. $S^{(\vec{r},X)}_{jk;d(x)}$: It takes a value of (1) if event (k) of cluster (j) that is associated with the x-neuron of $\vec{r}$ occurs in place $d(x)$, and zero otherwise.
4. $S^{(\vec{r},X)}_{\hat{j}\hat{k};d(x)}$: It takes a value of (1) if event ($\hat{k}$) of cluster ($\hat{j}$) that is associated with the x-neuron of $\vec{r}$ occurs in place $d(x)$, and zero otherwise.
5. $\tilde{S}^{(\vec{r},T)}_{j;d(t)}$: It takes a value of (1) if cluster (j) that is associated with the t-neuron of $\vec{r}$ occurs in time $d(t)$, and zero otherwise.
6. $\tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)}$: It takes a value of (1) if cluster ($\hat{j}$) that is associated with the t-neuron of $\vec{r}$ occurs in time $d(t)$, and zero otherwise.

In the above list, the last 2 neurons act as what Peterson calls *effective time*-neurons. Not only will they handle periods of single hours but also variants of longer and shorter intervals. These neurons can be defined as follows:

Set (ϵ) and (c) to some desired values depending on the time period of the set of classes, now, an *effective* t-neuron is:

$$\tilde{S}^{(\vec{r},T)}_{j;d(t)} = \sum_{k=0}^{g_j-c} S^{(\vec{r},T)}_{j;d(t-k+\epsilon)} \tag{55}$$

$\tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)}$ can be defined in a similar manner. If ($c = 1$) and ($g_i = 2$), then we are dealing with double-hour intervals so we set ($\epsilon = 0$). If ($c = 1$) and ($g_i = 1$) then ($\epsilon = 0$) and

$$\tilde{S}^{(\vec{r},T)}_{j;d(t)} = S^{(\vec{r},T)}_{j;d(t)}. \tag{56}$$

Also, let $S^{(R)}_{q(i)r(i);b(x,t)}$ for balancing the enrollment in a multi-section courses. Section $q(i)$ of students $r(i)$ are balanced over space-time $b(x, t)$.

To enroll student $r(i)$ in a particular section $q(i)$ given the tag EC to indicate the status of the student, we have $S^{(X[T])}_{q(i)r(i);(d(x)[d(t)])}|EC \in \{0, 1\}$ is set to 0 if EC is 0, otherwise, it is set to 1 and the action $q(i)r(i)$ occurs during place $d(x)$ and/or time $d(t)$.

Let $lin(c_i, c_j)$ representing the concept that course c_i is linked to some other course c_j ; in this case, c_i is a pre-requisite to c_j. The degree of "linkage" varies between 0 and 1. It takes a value of 1 if under all circumstances, course c_i must be taken before c_j. To deal with this link concept, we assumed above that we have an attribute $s(j,k)$ to represent a subject. For a strong linkage, we will assume that we're dealing with subjects that belong to the same cluster (j) (e.g. physics courses), so the link between various events of (j) is: $lin(s(j,k), s(j,\hat{k}))$. Other degrees of linkage can involve subjects or courses from different groups or clusters (e.g. physics and math courses, or engineering and math courses), and this connection is in the form: $lin(s(j,k), s(\hat{j},\hat{k}))$.

Let $\hat{x}(i)$ be a single location (or a seat) in class $x(i)$ such that $i = 1, 2, \ldots, num$; where (num) is the number of class-rooms.

5.4.1 Day-Class Balancing

Each student of any particular section (i.e. $r(k)$) should have a balanced schedule between (*MWF*) and (*TuTh*). The balancing operation is accomplished by spreading as much as possible the courses of the student over each of (*MWF*) and (*TuTh*).

Let D be the set $\{\{MWF\}, \{TuTh\}\} = \{D_1, D_2\}$. Assume the existence of the subject attribute $s(j,k)$. We introduce the energy function E_{QSD} that spreads the classes (Q) in a particular field or subject (S) over the set $\{D_1, D_2\}$.

The Potts condition of balancing $S^{(\mathbf{R})}_{q(m)r(m);b(x,t)}$ (take a value of 1 for balanced and 0, otherwise) is accomplished through one (or all) of these energies:

$$E^r_{QSD} = \frac{1}{2} \sum_{j \neq \hat{j}} \sum_{k\hat{k}} \delta_{q(j,k),q(\hat{j},\hat{k})} \delta_{s(j,k),s(\hat{j},\hat{k})} \sum_{d_1,d_2} S^{(D)}_{j,d_1} S^{(D)}_{\hat{j},d_2} \tag{57}$$

$$E^r_{QSD_1} = \frac{1}{2} \sum_{j_1 \neq \hat{j}_1} \sum_{k_1\hat{k}_1} \delta_{q(j_1,k_1),q(\hat{j}_1,\hat{k}_1)} \delta_{s(j_1,k_1),s(\hat{j}_1,\hat{k}_1)} \sum_{d_1} \sum_{i \neq \hat{i}} S^{(D)}_{i,d_1} S^{(D)}_{\hat{i},d_1} \tag{58}$$

$$E^r_{QSD_2} = \frac{1}{2} \sum_{j_2 \neq \hat{j}_2} \sum_{k_2\hat{k}_2} \delta_{q(j_2,k_2),q(\hat{j}_2,\hat{k}_2)} \delta_{s(j_2,k_2),s(\hat{j}_2,\hat{k}_2)} \sum_{d_2} \sum_{i \neq \hat{i}} S^{(D)}_{i,d_2} S^{(D)}_{\hat{i},d_2} \tag{59}$$

5.4.2 The Case of Student Athletes

The main constraint of this case very much analogous to the **hard** constraint stated somewhere above : *different events should not occupy the same space-time slot.*

$$E^{\vec{r}}_{d(XT)} = \frac{1}{2} \sum_{d} \sum_{xt} \sum_{\dot{r}b} \sum_{\ddot{r}a} S^{(\vec{r},X[T])}_{\dot{r}b;d(x)[d(t)]} S^{(\vec{r},X[T])}_{\ddot{r}a;d(x)[d(t)]} \tag{60}$$

Where $(\dot{r}b)$ and $(\ddot{r}a)$ are some two different events. After some further refinement, we obtain:

$$E^{\vec{r}}_{d(XT)} = \frac{1}{2} \sum_{j \neq \hat{j}} \sum_{k\hat{k}} \sum_{d(x)} S^{(\vec{r},X)}_{jk;d(x)} S^{(\vec{r},X)}_{\hat{j}\hat{k};d(x)} \sum_{d(t)} \tilde{S}^{(\vec{r},T)}_{j;d(t)} \tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)} + \frac{1}{2} \sum_{j} g_j \sum_{k \neq \hat{k}} \sum_{d(t)} S^{(\vec{r},X)}_{jk;x} S^{(\vec{r},X)}_{j\hat{k};x} \tag{61}$$

Since it is always the case that athletic training facilities are kept separate from the academic class-rooms and lecture halls, a more refined energy function is obtained:

$$E^{\vec{r}}_{d(T)} = \frac{1}{2} \sum_{j \neq \hat{j}} \sum_{d(t)} \tilde{S}^{(\vec{r},T)}_{j;d(t)} \tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)} \tag{62}$$

5.4.3 Eligibility for Enrollment

Before going through the process of balancing, each student ($r(i) = r_s$) eligibility for enrolling in a given class need to be determined. Let the student's eligibility criteria be denoted by (EC), using the notation stated above, we could derive the following energy function —note: if EC is set to "No" then it takes a 0 value in the energy function, otherwise, it is a 1 :

$$E_{QR} = \frac{1}{2} \sum_{qr} \sum_{jk} \sum_{i} \delta_{q,q(j,k)} \delta_{r,r(i)} S^{X}_{ij;x(i)} | (EC \in \{0, 1\}) \tag{63}$$

$$\sum_{i} O_{q(i)} = 1 \Rightarrow \text{ Test: } \{EC(R) + \text{ satisfy-constraints } (Hard, Soft)\} \tag{64}$$

5.4.4 Course Pre-requisite and Subsequent Assignment

Utilizing the link concept ($\mathbf{lin}(\mathbf{s}, \mathbf{s}')$) between two different subjects, a general energy assignment can be put together as follow:

Let (i) stands for the situation at hand (also can be viewed as a space-time event).

$$E_{link(1)} = -\gamma \sum_{is} \left[\sum_{jk} \delta_{s,s(j,k)} \tilde{S}^{(X[T])}_{ln(s,s');i} \right]^2 \tag{65}$$

$$E_{link(2)} = -\gamma \sum_{s} \sum_{j \neq j'} \sum_{k \neq k'} \delta_{s(j,k),s(j',k')} \sum_{i} \tilde{S}^{(X[T])}_{ln(s(j,k),s(j',k'));i} \tilde{S}^{(X[T])}_{ln(s(j,k),s(j,k'));i} \tag{66}$$

$$E_{link(3)} = -\gamma \sum_{j \neq j'} \sum_{k \neq k'} \delta_{s(j,k),s(j',k')} \tilde{S}^{(X[T])}_{ln(s(j,k),s(j',k'));i} \tilde{S}^{(X[T])}_{ln(s(j,k),s(j,k'));i} \tag{67}$$

If ($\gamma > 0$) then function minimization is the target here, otherwise maximization will take place.

5.4.5 Enroll in all parts of a course

There are a number course of different types that not only have lecture periods but also associated sections that consist of either labs or recitations or both. Signing up for such a course entails enrolling in all associated sections of it.

Let us assume the existence of some subject attribute $s(j,k)$ consisting of 2 components: $\dot{s}$ denoting lectures, and $\ddot{s}$ stands for labs. Now the energy equation depicting this assignment can be outlined as follow:

$$E_{enroll(1)} = \frac{1}{2}\sum_s \sum_{jk} \sum_i \delta_{s,s(j,k)}\delta_{r,r(i)} \sum_t S^{(T)}_{jk;t} \tag{68}$$

$$E_{enroll(2)} = \frac{1}{2}\sum_{si} \sum_{t\neq t'} \left[\sum_{jk} \delta_{s,\dot{s}(j,k)}\delta_{r,r(i)} S^{(T)}_{jk;t}\right] \left[\sum_{jk'} \delta_{s,\ddot{s}(j,k')}\delta_{r,r(i)} S^{(T)}_{jk';t'}\right] \tag{69}$$

$$E_{enroll(3)} = \frac{1}{2}\sum_{k\neq k'} \delta_{s,\dot{s}(j,k)}\delta_{s,\ddot{s}(j,k')} \sum_{t\neq t'} S^{(T)}_{jk;t} S^{(T)}_{jk';t'} \tag{70}$$

5.4.6 Lecture hall capacity

In order to satisfy this constraint, the following inequality need to hold.

$$\forall x \sum_i ||x(i)| - \sum_j \sum_k S^{(X)}_{jk;x(i)}| \geq 0\ . \tag{71}$$

5.4.7 Student status and priority mapping

$$E_{assign} = \frac{1}{2}\sum_{m,\hat{x},x} \sum_{i\neq i'} S^{(X)}_{\hat{x};x} \left(S^{(R)}_{r(i);(\hat{x}(m)\in x(m))}|(M = Req) + S^{(R)}_{r(i');(\hat{x}(m)\in x(m))}|(M = \chi)\right) \tag{72}$$

5.5 On the Energy Equation of the Potts Model

The energy function of the Potts spin model is of the form:

$$E = E_A + E_B + E_C + \ldots \tag{73}$$

In [23], in which the TSP is used as study case, parameters α and β are used to control the strength of E's restraints. The same parameters can also be used when dealing with scheduling.

Observation 5.1 For timetabling, β is split into two parts ($\beta^{(T)}$ and $\beta^{(X)}$), one with respect to the time-based neurons and the other with respect to the space-based neurons. E_β becomes as follows:

$$-\frac{\beta^{(T)}}{2}\sum_j \sum_t [S^{(T)}_{j;t}]^2 - \frac{\beta^{(X)}}{2}\sum_{jk} \sum_x [S^{(X)}_{jk;x}]^2 \tag{74}$$

Here, the appearance of $\beta^{(T)}$ and $\beta^{(X)}$ is only in the diagonal terms. They play a crucial role in monitoring the dynamics (phase transitions, etc.). These diagonal terms correspond to self coupling interactions for the neurons of the network. In our simulations, each is set to -0.1. □

6 Some Performance Analysis of MFA

Given M_x = average number of rooms for each category of classes, N_x = the overall total number of rooms, and N_t = total number of slots , putting the figures of the students aside, with a computational complexity of $O(M_x N_x N_t)$ per sweep, software implementation of the MFA are unlikely to be competitive with conventional serial algorithms such as 3-opt and Lin-Kernighan's for problems such as the TSP, which empirically have an overall $O(N^2)$ running time [20]. Ideally, one would like to study the likely performance of the method through numerical simulations using a small step-size γ to approximate continuous time dynamics. However, this is computationally very expensive: not only do the simulation times grow as $\frac{1}{\gamma}$, but also small values of (γ) accentuate the difference between the expected value T_c^{exp} of the critical temperature T_c and the theoretically computed value T_c^{theo}. That is, low γ runs tend to depress T_c^{exp} artificially and thereby degrade solution quality, and as we observed, result in a number of constraint violations. Many of those violated constraints are hard, but the majority are classified to be soft to medium. This later point can be compensated for by more stringent annealing parameters, but at the cost of longer simulation times. No attempts in our part have been made to carry this out. Also, we have observed that the overall quality of the solution degraded as the problem size increased. This degradation spans both soft/medium as well as hard constraints, and hence, illegal timetables.

One thing that remains unclear is to what extent very slow annealing can bring MFA close to finding the optimal (not only legal) schedule. Whereas simulated annealing, given certain conditions, guarantees (theoretically) convergence to the optimal solutions for asymptotically slow annealing, such results is not true for MFA algorithms, because MFA is an approximation.

7 Experimental Analysis

Our computations were done with a number of goals in mind. The main objective was to provide a schedule which satisfied all hard constraints and minimized the cost of medium and soft constraints, using real-life data sets for a large university. Our study case involved real scheduling data covering three semesters at Syracuse University. The size and type of the three-semester data is shown in Table 1. Nine types of rooms were used: auditoriums, classrooms, computer clusters, conference rooms, seminar rooms, studios, laboratories, theaters, and unspecified types. Staff and teaching assistants are considered part of the set of professors. Third semester (summer) data was much smaller than other semesters, however, there were additional space and time constraints and fewer available rooms. Our data was quite large in comparison to data used by other researchers. For example, high school data used by Peterson and colleagues [13, 14] consists of approximately 1000 students, 20 different possible majors, and an overall periodic school schedule (over weeks). In the case of Abramson *et al.* [2], their data set was created randomly and was relatively small, and they stated that problems involving more than 300 tuples were very difficult to solve.

Table 1 lists all major components of the data and Tables 2 and 3 show the rates of sparseness of the three-semester data which greatly depends on number of time slots and rooms used. The level of sparseness increases as the overall size of the data, particularly, number of rooms, time slots, and number of classes. Also, after scheduling all the classes, there will be about $N_{sp} = (N_x N_t - N_l)$ spare space-time slots, hence, the sparseness of the problem is defined as as the ratio $N_{sp}/(N_x N_t)$. As we can see that Table 3 degree of sparseness is less than that of Table 2 and the reason is the presence of the students' preferences. Including student preferences makes the problem much harder, but these are viewed as medium constraints and thus are not necessarily satisfied in a valid solution.

Table 4 shows the percentage of classes that could be scheduled in accordance with the hard constraints. Other constraints such as medium and soft were also taken into account. Since the annealing process in MFA is "deterministic" the results shown were averaged over 10 runs each having different initial conditions. As expected, due to a higher sparseness ratio of the third (summer) semester, the method does quite well. Also, we have observed that the overall results are generally below those of SA and ES [9]. In addition, we have found in the implementation of this method that the results were quite sensitive to the size of the data as well the type of constraints involved. If we confine ourselves to the set of hard constraints, the results are as good as or even better than the other methods. However if we take into account the medium and soft constraints, that is, the overall cost function, this method does not perform as well.

8 Conclusion

We have applied the method of Mean Field Annealing to the difficult problem of academic scheduling for a large university. We found that MFA works well for small scheduling problems, but does not appear to scale well to large problems with many complex constraints. It is more difficult to tune the parameters, and use a preprocessor to take advantage of a good initial solution, than methods such as simulated annealing. Because of the complexity and size of the Potts neural encoding in MFA, there seems to be no clear way of preserving the state of an initial configuration provided by a preprocessor when carrying out the Potts encoding.

	First Semester	Second Semester	Third Semester
Rooms	509	509	120
Classes	3839	3590	687
Professors	1190	1200	334
Students	13653	13653	2600
Buildings	43	43	11
Schools/Colleges	20	21	17
Departments/ Course Prefix	143	141	108
Areas of Study (majors)	200	200	200

Table 1: Size of a three-semester data.

Academic Time Period	sparseness ratio
First Semester	0.50
Second Semester	0.53
Third Semester	0.62

Table 2: A three-semester sparseness ratio each (students were not included) computed using $N_{sp}/(N_x N_t)$ is shown. The smaller the ratio, the less sparse the problem is and for this problem the ratio is based on $N_t = 15$ time slots each with a duration of 55 minutes. The first slot starts at 8:00 am and the last slot ends at 9:45 pm.

Academic Time Period	sparseness ratio
First Semester	0.35
Second Semester	0.38
Third Semester	0.50

Table 3: A three-semester sparseness ratio (including students) computed by $\mathcal{R}(N_{sp}/(N_x N_t)$, ((majors * max classes taken per students) / number of students)). The smaller the ratio, the less sparse the problem is and for this problem the ratio is based on $N_t = 15$ time slots each with a duration of 55 minutes. The first slot starts at 8:00 am and the last slot ends at 9:45 pm.

Academic Time Period	Scheduled (average) %	highest Scheduled %	lowest Scheduled %
First Semester	65.60	71.00	61.00
Second Semester	67.20	75.00	65.00
Third Semester	88.00	95.00	82.00

Table 4: A three-semester MFA results: percentage of scheduled classes, averaged over 10 runs of different initial conditions.

References

[1] Abramson, D, "Constructing school timetables using simulated annealing: sequential and parallel algorithms," *Management Science*, 37(1), 98-113, 1991.

[2] Abramson, Dang, and Krishnamoorthy, "An Empirical Study of Simulated Annealing Cooling Schedules", Griffith Univ. report, Nathan, Qld, Aus. (1994).

[3] *Applied Simulated Annealing*, edited by Rene Vidal, Lecture Notes in Economics and Mathematical Systems no. 396, Springer-Verlag, 1993.

[4] Carter, M, and C. Tovey, "When is the classroom assignment problem hard?," Working Paper 89-03, Dept. of Industrial Engineering, Univ. of Toronto, 1989. (was submitted to *Operations Research*).

[5] Colorni, A, M. Dorigo, and V. Maniezzo, "On the use of genetic algorithms to solve the timetable problem", report 90-060, Dipartimento Di Elettronica E Informazione, Politecnico Di Milano, Milano, Italia.

[6] Dowsland, K, Using Simulated Annealing for Efficient Allocation of Students to Practical Classes. Stat and OR Group, EBMS, Univ College of Swansea, UK, 1993.

[7] de Werra, D, "An introduction to timetabling," *European Journal of Operational Research*, 19, 151-162, 1985.

[8] Eiselt H. A, and G. Laporte, "Combinatorial Optimization Problems with Soft and Hard Requirements," *J. Operational Research Society*, vol. 38, No. 9, pp. 785-795, 1987.

[9] Elmohamed, S, P. Coddington, and G. Fox, "A comparison of Annealing Techniques for Academic Course Scheduling", submitted to the 2nd international conference on the practice and theory of automated timetabling.

[10] Even, S., H. Itai, and A. Shamir, "On the complexity of the timetable and multicommodity flow problems", SIAM Jr. Computing 5 (76), 691-703.

[11] Gary, M. and D. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness.* Freeman, San Francisco (1979).

[12] Gee, A, private communication.

[13] Gislén, L, B. Söderberg, and C. Peterson, "Teachers and Classes with Neural Nets", *Inter. Jr. of Neural Systems* 1, 167 (1989).

[14] Gislén, L, B. Söderberg, and C. Peterson, "Scheduling High Schools with Neural Nets", Lund University Preprint LU-TP-91-9, Lund, Sweden (1991). (appeared in *Neural Computation* 1992)

[15] Gosselin, K, and M. Truchon, "Allocation of Classrooms by Linear Programming", *Journal of Operational Research Society* 37, 561 (1986).

[16] Hertz, A. "Finding a feasible course schedule using Tabu search", *Discrete Applied Mathematics* 35, 255-270, 1992.

[17] Hertz,J., A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley, Redwood City, 1991.

[18] Hopfield, J, and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics* 52, 141 (1985).

[19] Hopfield, J. "Neurons with graded response have collective computational properties like those of two-state neurons". *Proceedings of the National Academy of Science* USA 81, 3088-3092 (1984).

[20] Johnson, D. "Local Optimization and the TSP," Springer *Lecture Notes in CS* 443, 446-461, Springer-Verlag, 1990.

[21] Mouritsen, O. G, Computer Studies of Phase Transitions and Critical Phenomena. Springer-Verlag, Berlin, 1984.

[22] Peterson, C, "Parallel Distributed Approaches to Combinatorial Optimization Problems - Benchmark Studies on TSP", *Neural Computation* 2, 261 (1990).

[23] Peterson, C, B. Söderberg, "A New Method for Mapping Optimization Problems onto Neural Nets", *International Jr. of Neural Systems* 1, 3 (1989).

[24] Peterson, C, B. Söderberg, "Artificial Neural Nets and Combinatorial Optimization Problems", To appear in *Local Search in Combinatorial Optimization*, edited by Aarts and Lenstra , Wily and Sons 1997.

[25] Schaerf, A, "A survey of automated timetabling," Department of Software Technology, Report CS-R9567, CWI, Amsterdam, The Netherlands.

[26] Shrijver, A, *Theory of Linear and Integer Programming*, John Wiley and Sons, Chichester, UK, 1986.

[27] Simic, P., "Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimizations", *Network* 1, 89-103 (1990).

[28] Simic, P., "Constrained Nets for Graph Matching and Other Quadratic Assignment Problems", *Neural Computation*, v.3, n.2, 268-281 (1991).

[29] P. Simic, private communication.

[30] Simmen, M, private communication.

[31] Van den Bout, D. E, and T. K. Miller III. Graph partationing using annealed neural networks. *IEEE Transactions on Neural Networks*, 1(2):192-203, June 1990.

[32] Wu, F, Y. "The Potts model", *Reviews of Modern Physics* 54, 235-268 (1982).