DRAFT-Report.1

Mean Field Annealing, Simulated 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

DRAFT-Report

Mean Field Annealing, Simulated 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

ABSTRACT

In this study we tackle a complex problem of academic course scheduling (or timetabling) at the university level. First, we utilize feed-back artificial neural system models along with the mean field theory approximation. We also extend the Potts normalization conditions and equality constraints to accommodate our study case. The second approach used is that of simulated annealing using two different annealing schedules. A rule-based system is used as a pre-processor in both approaches.

Key words : feed-back neural nets, MFT, tanh functions, Potts systems, constraints, events, stochastic updating, annealing, simulated annealing, cost functions, temperature, annealing schedule, phase transition, rule-based system.

June 18, 2000

Contents

1 Introduction

The prime objective of this study is to derive an approximate solution to a high-dimensional and non-geometric combinatorial optimization (CO) problem, which is academic scheduling at the university level.

From the various categories of artificial neural systems (ANS) for tackling CO problems, we choose one most suitable for the problem at hand; which is the family of feedback ANS based on multi-state neurons with mean field theory (MFT) equations controlling 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.

The second approach is that of simulated annealing. Two different schedules are used: the standard geometric cooling schedule, and the geometric re-heating as a function of cost. Our focus of analysis in this approach will center around the latter schedule.

2 Background and Motivation

The problem of academic scheduling or timetabling has been tackled by many researchers [Carter-93], mostly in the field of operations research. This is a very difficult problem due to its nature as a multi-objective non-Euclidean combinatorial optimization problem for which no good solution has previously been found.

Lately, there have been a number of *non-conventional* (or novel) methods utilized to tackle this problem from different angles. Chief among them are neural networks [HT-85], simulated annealing [Vidal-93] and [Abramson-94], genetic algorithms [CDM-92], tabu search [Hertz-92], and various combinations of two or more of these techniques. In this paper, we shall confine ourselves to simulated annealing using the schedule of re-heating as a function of cost, and mean field annealing. A rule-based system is used as a preprocessor for each of these method.

Much of the work in the neural nets field has centered on understanding and improving on the Hopfield-Tank TSP algorithm [HT-85] (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 non-Euclidean CO problems such as scheduling.

3 The Timetabling Problem - description and solution methods

One of the problems that does not have a well-defined objective function to optimize and of the class of constraint satisfaction problems is the NP-complete *teachers and classes* timetabling problem studied in [Abramson-94], [Dowsland-93], and [GSP-89.1].

Briefly, the problem is defined as follows:

Definition 3.1 For a certain school with N_p teachers, N_q classes, N_x classrooms or lecture halls, and N_s students, it is required to schedule N_l lessons (defined as teacher-class pairs) within a time limit of N_t lesson periods (or time slots).

A legal schedule needs to be found such that no teacher, class, or student is in more than one place at a time, and no lecture hall is expected to accommodate more than one lesson at a time or more students than its capacity. □

From this point on, we will refer to academic scheduling or timetabling as TTS (short for a timetabling system).

3.1 The Constraint Satisfaction Problem

We now briefly describe the general constraint satisfaction problem which has been widely studied in the Artificial Intelligence community [Montanari][Ladkin-Maddux][Mackworth]

Definition 3.2 *An instance* of a constraint satisfaction problem *consists of*:

- *A finite set of variables,* N_i*, where* $i = 1, 2, \ldots, n$.
- *A domain of* D *of possible values of* N_i .
- *A set of constraints* $\{C(S_1), C(S_2), \ldots, C(S_c)\}$.

  Each S_i *is an ordered subset of the variables, and each constraint* $C(S_i)$ *is a set of tuples indicating the mutually consistent values for the variables in* S_i .

□

The length of the tuples in a given constraint will be called the *arity* of that constraint*. In particular, unary constraints specify the allowed values for a single variable, and binary constraints specify the allowed combinations of values for a pair of variables.

A *solution* to a constraint satisfaction problem is an assignment of values to the variables which is consistent with all of the constraints. Deciding whether or not a given problem instance has a solution is NP-complete in general [Mackworth], even when the constraints are restricted to binary constraints.

3.2 Constraints of TTS

In a nutshell, timetabling is the assignment of time slots to a set of events, **subject to** constraints on these assignments. The medium and soft constraints have an associated cost and if they are not satisfied, the goal is to minimize this cost. In regard to the hard constraints, one must derive their associated cost (or penalty) to zero.

These constraints are divided into three types:

*Strictly speaking we also need to associate an arity with an empty set of tuples, but we will neglect this special case, in order to simplify the presentation.

1. Hard - These constraints that must be satisfied.
2. Medium - These constraints should be satisfied, if possible, but are not required for a valid solution.
3. Soft - These constraints have a lower priority but always the preference is to minimize them.

3.2.1 Hard Constraints

The most common and important hard constraint is what is sometimes referred to as an *edge* constraint between two events. This reference is due to the similarity of any **simple** timetabling problem with *vertex coloring* in graph theory; which simply states that any given pair of events $\mathbf{e_1}$ and $\mathbf{e_2}$ must not overlap in time. ********[more on this, shortly]************

Those events that must not overlap in time include:

- two classes taught by the same professor,
- a class and a recitation or a lab of the same class,
- classes with students in common.

Other general hard constraints include:

- A class can not be assigned to a particular room unless *size of the room* $\geq$ *the class enrollment*. The smaller the difference between these two sizes, the better the assignment.
- A limit on how many classes taught by each instructor is enforced.

Other constraints are those directly related to students such as:

- To sign up for a class or section, it must be open.
- Eligibility criteria for the section must be met, etc.
- Do not enroll athletes in courses that conflict with their sport practice time, etc.

These constraints have to be respected at all costs.

3.2.2 Medium Constraints

These kinds of constraints fall into, so to speak, the shaded region between the *hard* and *soft* constraints in the sense that they do not deal with time conflicts but do have a high penalty attached to them. Also, this penalty is not as high as that associated with the hard constraints. In the final (legal) schedule the penalty of these constraints is minimized and it is quite preferable to reduce it to 0.

In short, these kind of constraints should be satisfied but are nevertheless relaxable.

Examples of medium constraints

- For each student, balance *MWF* and *TuTh* sections.
- Balance or spread out the lectures over the week.
- Classes may request contiguous time slots.
- Classes may be grouped with other classes.
- The overall schedule should have as few "holes" as possible; that is lessons should be glued together, etc.

3.2.3 Soft Constraints

Like medium constraints, they also do not deal with time conflicts but unlike the medium constraints, have a low penalty (or cost) associated with them. Furthermore, this cost is lower than that of the medium constraints and much lower than that of the hard ones. In addition, in the final schedule, this cost is minimized but does not often approach zero.

Examples of soft constraints

- Balance enrollment in multi-section courses.
- Lunch and other break times may be specified.
- Teachers may request periods in which their classes are not taught.

3.3 The overall cost function

The *cost function* measures the infeasibility of the current solution and may involve the weighted sum of penalties associated with different types of feasibility violation.

We consider the following as part of the function:

- Distance cost: proportional to the distance between where the class is assigned and the building which is housing its home department or school.
- Time period cost consisting of :
  1. class cost,

2. professor cost,
3. cost of class spreading over the week and within each of the weekdays,
4. cost of class glueing and filling in any holes of the timetable,
5. room size cost, and
6. students cost which is the sum of various costs associated with each student, such as eligibility criteria and number of classes taken by the student.

- The cost of professor is computed from:

  1. cost incurred from assigning more classes than the maximum number taught by that professor, plus
  2. cost from not having a long enough time interval between courses taught by the same professor, and
  3. cost of assigning to the same professor two or more classes that overlap in time.

- The cost of room size is computed in a non-linear fashion depending on the magnitude of the difference between the class enrollment and the room size.

Note: *Using summation is not strictly necessary, provided that the components can be combined in some fashion to reflect the quality of the solution.*

3.4 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-complete problem, see Gary and Johnson [GJ-79].

The problem 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.

Using the LP representation, for a set as small as 20 teachers, 30 time-intervals, and 10 classes, the problem would be represented by 6000 variables. This makes the problem intractable.

The overall complexity of timetabling is in the order of $2^{(N_c N_x N_t)^a}$, for some $a \geq 1$.

For a much simpler problem, which is the *class allocation* (no timetabling part included), using linear programming [GT-86], the computational load (n_c) scales as follows:

$$n_c \propto N_c^3 N_x^2 N_t^3 \quad , \tag{1}$$

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

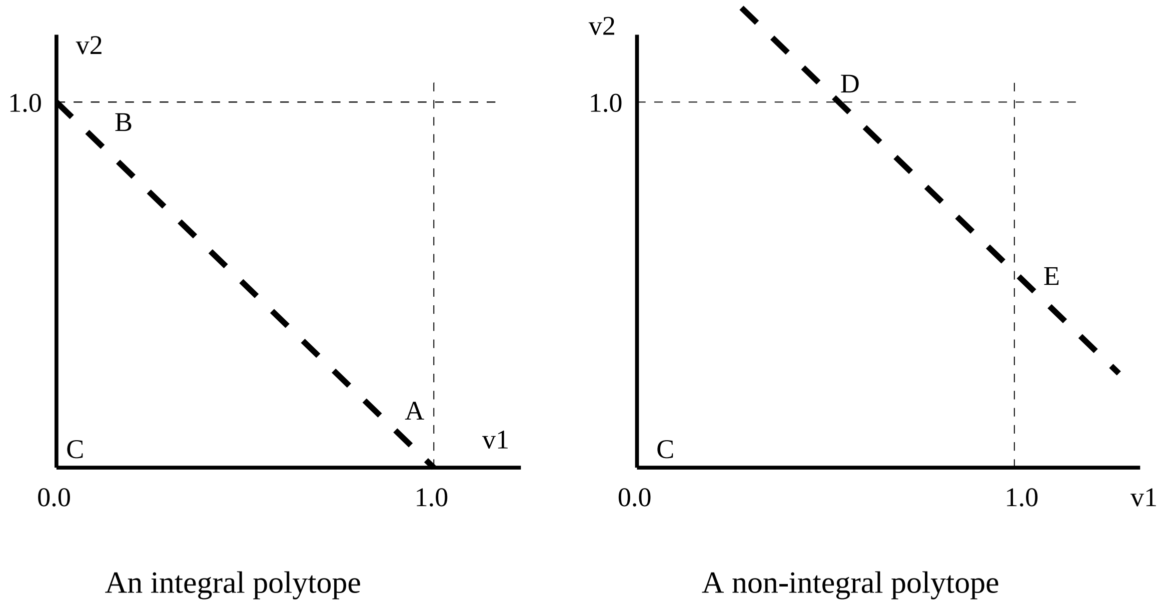

Figure 1: An integral polytope has all its vertices at integral points (points whose coordinates are all integers). It follows that an integral polytope within the unit hypercube has all its vertices at $[0 - 1]$ points.

4 Some Methods of Tackling Timetabling

The timetabling problem has no cost function, only constraints (hard, medium and soft) each with their associated penalty (or cost); all we have to do is find any integral vertex within the constraint polytope P (see Figure 1). To elaborate a bit on this, using the work of Schrijver [Shrijver-86] and [Shrijver-95]: we think of each constraint as a hyperplane in the solution space. The set of constraints define a bounded polyhedron (or polytope) within which the state vector must remain if it is not to violate any constraints. The 0 or 1 solution points will lie at some of the vertices of this polytope, and whatever annealing method is used, it must force convergence to one of these points. The problem with annealing (of course, depending on the schedule used) is that the best one can expect is convergence to a vertex, not necessarily a 0 or 1 vertex. In the case of MFA, the ability of the neural net finding a valid solution point reliably therefore depends on the polytope having vertices of 0 or/and 1 values (in short, 0-1 vertices), in which case it is called an integral polytope. Now, polytopes like the one for the permutation matrix mapping for the TSP are integral, so there is no problem finding valid TSP solutions (though the tour length is another question altogether). Class scheduling polytopes are a totally different story: Gee [Gee-93.1] and some of his colleagues at Cambridge University have done some experiments on them and found that for timetabling problems as small as five teachers, five classrooms, five classes, and five time-slots, the proportion of 0-1 vertices is only about 1%. This mean that on average one has to run a descent procedure about 100 times to find a valid solution! Some of our experiments with sets of teachers, classes, classrooms, students and time-slots of various sizes confirm some of Gee's results and conclusions. More on this in section (?????).

Timetabling problems can be characterized by their *sparseness*. After all the required lessons have been scheduled, there will be $N_s = (N_x N_t - N_l)$ spare space-time slots, hence, the sparseness of the problem is defined as the ratio $N_s/(N_x N_t)$. The sparseness decreases as $N_s/(N_x N_t)$ goes to zero or as N_s becomes smaller and smaller. Furthermore, [Gee-93.1] notes that for sparse problems, the probability of an arbitrary descent procedure converging to an integral vertex is near one, though this falls off rapidly for less sparse problems. For dense problems, there is an additional correlation involving the problem size: as the problem gets larger (more classes, teachers, rooms, students, etc.), the proportion of integral vertices gets smaller. From this we could deduce that a one-shot descent procedure, operating on an arbitrary "objective function", is not likely to be very successful with difficult constraint satisfaction problems, most probably converging to a non-integral vertex of P, for example. On the other hand, any "partial" or "imperfect" solution to a timetabling problem found by methods such as simulated annealing (using some good adaptive schedules) is likely to be of some use. Since simulated annealing does not search over the interior of the unit hypercube, but only over its vertices, the solution it finds will be a unique timetable, possibly -as it is the case in our experiments- violating as few as one or two constraints. As part of the post-processing operation, we could at least remove some lessons to obtain a valid (or legal) timetable which fulfills most of the teaching requirements. Furthermore, we could do few more swaps (or moves) of classes around to arrive at a timetable meeting all the original requirements.

5 Mapping of Timetabling to a Graph Coloring Problem

5.1 Graph Coloring

Given a graph $G = (V, E)$, the graph coloring problem is to find a coloring of its vertices with as few colors as possible. The main constraint is that two vertices linked by an edge should not receive the same color (such coloring are called feasible coloring). One could define the set S of feasible solutions as the set of feasible coloring which do not use more than a given number of colors; for example, this upper bound may be the number of colors used in the initial solution.

5.2 The Mapping

1. The courses are the vertices of a graph.
2. The time periods correspond to the colors.
3. Two vertices are linked by an edge if the corresponding courses cannot be given at the same time.
4. Additional constraints have to be taken into account and mapped onto the graph.

5.3 Interpretation of the Solution Space

After setting up or mapping the problem we need to determine the following:

- A feasible solution is defined as a schedule that satisfies a subset (C') of constraints (C) : $C' \subseteq C$; provided that all hard constraints are part of C' .
- A neighboring solution is obtained by modifying the current solution space (i.e. swapping two or more classes in time).
- Here are two guidelines in determining whether a given constraint to be included in C' or not:
    1. A class should never be assigned to a time period if such an assignment induces only class schedules that do not satisfy all the constraints (specifically the hard constraints).
    2. A conflict between two or more classes should not be a sufficient condition for hindering an assignment.
    3. If all courses last one time period, this is equivalent to changing the color of a vertex, for example.

6 The Preprocessor Stage

6.1 The Problem

Given data sets of courses and their days, hours, enrollments, and instructors; rooms and their capacities, specification, and locations; distances between buildings; priorities of each building; students and their courses: the problem is to construct legal school timetable(s) satisfying all the hard constraints and minimizing the medium and soft constraints.

Courses that are not assigned to rooms due to conflict in time or size are left out of the final timetable for the user to handle manually.

6.2 The Method

The technique used is a rule-based system with lots of heuristics and conventional recursion to assist in carrying out course assignments. Here are few examples of some of the implemented rules dealing with **space and hours**, **type of room**, and **priority of room**:

6.2.1 Space and Hours

IF [room(capacity) > course(space-requested)] and [no time conflict in this room] THEN assign the room to the course:

[If more than a room having an identical availability time are present in the database, then choose the one with the minimal magnitude of the difference between room(capacity) and course(space-requested)].

Input: CPS-615 requests space for 75 students and meeting hours of [MWF 10:40 - 11:35 am].

In the database we have, for example, the following rooms:

- room: 1-218 CST , capacity: 45 students, and pre-occupied hours: none.
- room: 207 HL, capacity 120 students, and pre-occupied hours: [MW 09:45 - 11:10 am].
- room: Kitt-Aud HBC, capacity: 127 students, and pre-occupied hours: [TTH 10:40 - 12:00 am].

Output: Room Kitt-Aud is assigned to CPS-615.

6.2.2 Type of Room

IF [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]

THEN assign the room to the course: [If identical rooms in type are present in the database, then choose the one with the minimal magnitude of the difference between room(capacity) and course(space-requested)].

Input: BIO-615 requests space (not biology lab) for 35 students and meeting hours in [MW 10:40 - 11:35].

In the database we have, for example, the following rooms:

- room: 1-218 CST, capacity: 45 students, type: classroom, and pre-occupied hours: none.
- room: 124-SIMS, capacity: 40 students, type: laboratory, and pre-occupied hours: none.

Output: Room 124-SIMS is assigned to BIO-615.

6.2.3 Room Priority

Rooms are classified as:

1. General Purpose: used by all users.
2. Priority Rooms: are used based on certain priority of the class or the department.
3. Special Purpose: are only assigned to specific users.

- *IF [room(priority) == general] and [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]*

  THEN assign any user to this room.

- *IF [room has a priority user] and [the user is already assigned] and [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]*

  THEN assign other users.

- *IF [room(priority) == special purpose] and [user belongs to the special users of this room] and [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]*

  THEN assign the room to the user.

- *IF [room has a priority user] and [the user is already assigned] and [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]*

  THEN assign other users.

- *IF [room(priority) == special purpose] and [user belongs to the special users of this room] and [room(capacity) > course(space-requested)] and [no time conflict in assigning this room] and [type(room) matches type(requested)]*

  THEN assign the room to the user.

6.3 Components of the Rule-Based System

This system consists of a number of heuristics implemented to take care of various cases that arise when carrying out class scheduling. We have developed this system specifically for the problem of academic scheduling. The basic data structures or components of the system are:

- **Distance Matrix** of integer unit values between each academic department and every other building under use for scheduling.

- **Course Structure** of each course scheduled anywhere in campus. These structures are capable of linking with each other.

- **Room Structure** of each room (regardless of type) involved in the scheduling process. Like courses, rooms are capable of linking.

- **Time Periods Structures** to keep track of which hour or time slot is occupied and which is not.

- **Structure of department inclusion** many departments belong to other larger departments or colleges, etc.

6.4 The Overall Function of the System

- It takes in data from external sources:
  - Course data supplied by the university.
  - Building-room data also provided the registrar's office.
  - Department-to-building distance matrix.
  - Inclusion data, which a list of all colleges and schools and their associated departments or sections. Each list is headed by a college or a school prefix followed by its associated departments prefixes (i.e. ECS CS MAE etc.).

  Then builds an *internal database.*
- Using this internal database, the scheduling process takes place.

The scheduling process involves:

- Checking distance.
- Checking room type, building, and hours occupied.
- Check and compare hours for any possible conflict.
- Check rooms for any possible space conflict.
- Keep track of and update the list of hours already scheduled (or occupied).

6.5 The Output of the Rule-Based System

The *first output* of this system consists of classes, with their associated professors and students, assigned to various rooms.

The *second output* is a list of classes which could not be assigned due to time/space or other conflict(s).

These two outputs are fed to the main heuristic which can be either a *simulated annealing* or a *mean field annealing* algorithm to yield a final legal schedule.

6.6 The Rationale of Using a Preprocessor

- It is a good practice when using heuristics such as simulated annealing, to start at a good point in the search space or with a partial solution of the problem; regardless of the difficulty of the problem.
- A good starting point can only have a positive effect on the overall solution, and that is improving its quality.
- A good starting point or a partial solution will also have a positive effect on the convergence of the system; faster convergence to the final solution.

6.7 Mapping the Problem Components onto Time Periods

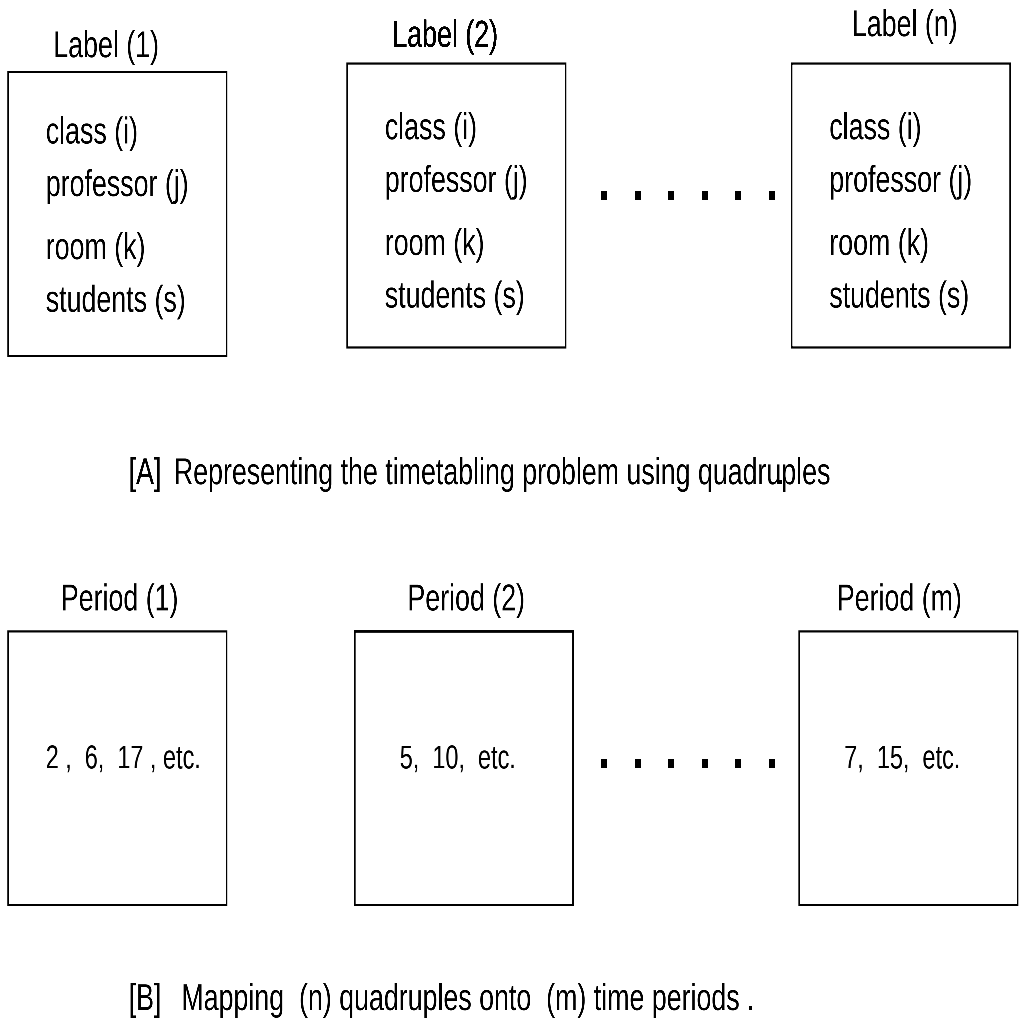

Figure 2: This shows how the main problem components are mapped onto the desired time periods.

7 Mean Field Annealing

Physical System	Optimization Problem
System State	Feasible Solution
Change of State	Neighboring Solution
Energy	Cost
Ground State	Optimal Solution
Rapid Quenching	Local Search
Careful Annealing	Simulated Annealing
Temperature	Control Parameter

Table 1: The analogy between physical systems and the optimization problems.

To start with, a summary of the Hopfield network is given as a prelude to mean field annealing.

7.1 The Hopfield Net - A summary of the continuous model

- The network is built by connecting a large number of neurons to each other to form a dense, parallel array.
- 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: $v_i = g_\lambda(u_i)$.
- The net's dynamics are governed by the first order differential equation:

$$\frac{\mathbf{du}}{\mathbf{dt}} = -\eta\mathbf{u} + \mathbf{W}\mathbf{v} + \mathbf{I^b} \ . \tag{2}$$

- $g_\lambda(u_i) = \frac{1}{1+e^{-2\lambda u_i}}$, for large λ, this function approximates a step function, forcing v_i to $[0,1] \implies$ (the output of the net lies on the unit hypercube).
- W_{ij} is the connection strength between neuron i and neuron j.
- I_i^b is an input bias to neuron i.
- Neurons may be updated individually at random (asynchronously) or all together (synchronously).

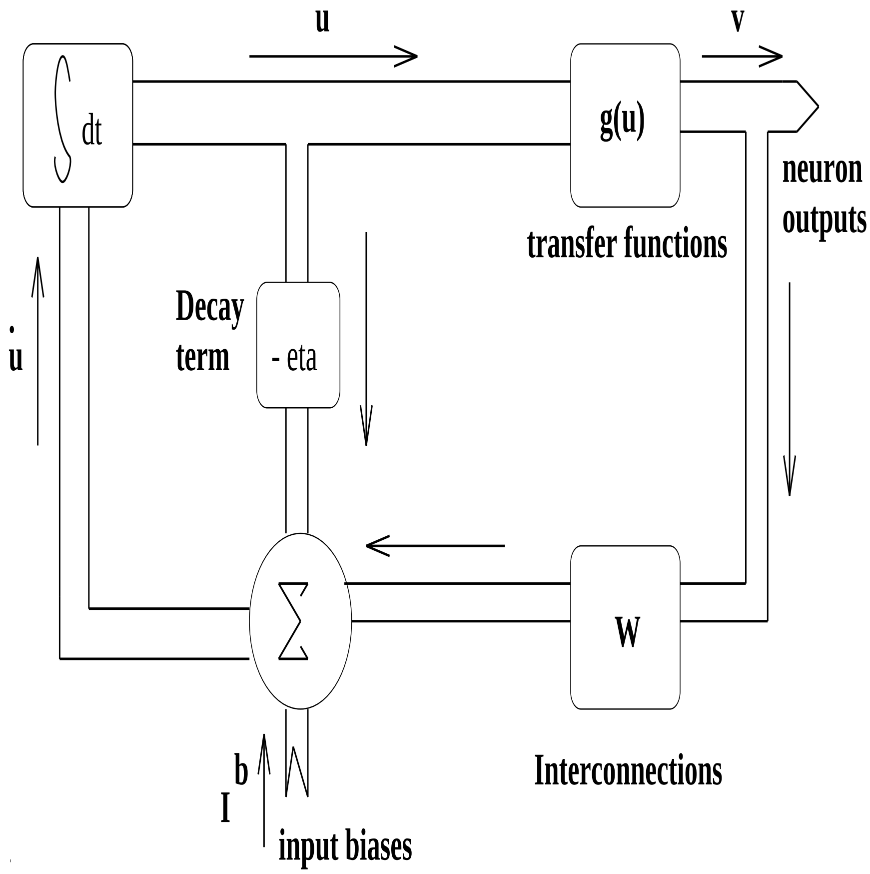

Figure 3: A schematic view of the Hopfield network.

7.1.1 Operation of the Network

The network (see Figure 3) operates as follows:

- The network's dynamics are governed by a set of nonlinear coupled differential equations of the form:

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} W_{ij} v_j + I_i^b \ . \tag{3}$$

  or the general form of it:

$$\frac{\mathbf{du}}{\mathbf{dt}} = -\eta \mathbf{u} + \mathbf{W}\mathbf{v} + \mathbf{I^b} \ . \tag{4}$$

- The I_i^b corresponds to an input bias to neuron i.
- 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.

7.1.2 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.

$$\dot{\mathbf{u}} = -\eta\mathbf{u} + \mathbf{W}\mathbf{v} + \mathbf{I^b} \ , \tag{5}$$

where

$$v = g(u) \ . \tag{6}$$

Equations 5 and 6 govern the dynamics of the network.

7.1.3 Stability of the Net

Given that $\mathbf{W}$ is symmetric, Hopfield proved stability for the network by showing that with the above dynamic equations (5 and 6) the network output $\mathbf{v}$ evolves so as to minimize the *Liapunov function*:

$$E^{liap}(v) = \left(-\frac{1}{2} \sum_{ij} W_{ij} v_i v_j - \sum_i I_i^b v_i \right) + \eta \sum_i \int_0^{v_i} g^{-1}(V)\, dV \ . \tag{7}$$

- $v = (v_1 \ldots v_n)$ 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.

7.1.4 Updating the States of the Net

1. Updating of the states (or the output of the neurons) is carried out either asynchronously or synchronously.
2. From a simulation point of view, we found that synchronous updating a bit more efficient.
3. 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.

7.1.5 Mapping the Problem onto the Network

- 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.
- A quadratic energy function (to be mapped onto the first term of 7) for which the minima correspond to solutions and the depth of each minimum reflects the solution quality.
- Hopfield and Tank example is the TSP, which they have mapped onto the net using a permutation matrix representation of the TSP tours.

7.1.6 Shortcomings of the Hopfield Model

- The net'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.

7.2 Mean Field Annealing: The Potts Approach 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 and Sӧderberg [PS-89]. 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 [GSP-89.1] and [GSP-91].

In [PS-89] the normalization (definition 7.5) occurs in the context of a *mean field annealing* approach. This generally yields the same final network equations as the 'neuronal circuit' approach of [Hopfield-84], 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 [Hopfield-84]) 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, 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, saddlepoint equations

†This equivalence was also noted in [HT-85]. P. Simic and Martin Simmen brought this point to our attention.

are derived (see next sections), the solutions of which pick out the dominant states of the network at the current temperature $\mathcal{T}$.

This statistical mechanics characterization gives a somewhat clearer picture of why the normalized model (which corresponds to a type of Potts model [Wu-82] 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 [PS-89] 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 $\mathcal{T}$ during a run, can be justifiably employed.

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

7.3 Notation and Definitions (The First Part)

In this section we define some key terms and concepts used in this paper and primarily adapted from the work 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.

Definition 7.1 Let $S_{pq;xt}$ denote the Potts (or multi-state) neuron, where

$$S_{pq;xt} = \begin{cases} 1 & \text{if event } (p, q) \text{ takes place in } (x, t) \;, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

□

Definition 7.2 There are two types of constraints to deal with in this paper:

1. Those involving professors, classrooms, subjects, and time periods.
2. Those dealing with students scheduling.

In addition, some of these constraints are usually classified to be **hard** and the rest are **soft**. □

Definition 7.3 Here, we define a number of **hard constraints** of the first type (see definition 7.2) .

- An event (p, q) should occupy precisely one space-time slot (x, t).

This constraint can be embedded in the neural net in terms of the Potts normalization condition:

$$\sum_{x,t} S_{pq,xt} = 1 \tag{9}$$

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

$$E_{XT} = \frac{1}{2} \sum_{x,t} \sum_{p_1,q_1} \sum_{p_2,q_2} S_{p_1 q_1;xt} S_{p_2 q_2;xt} = \frac{1}{2} \sum_{x,t} \left(\sum_{pq} S_{pq;xt} \right)^2 \tag{10}$$

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

$$E_{PT} = \frac{1}{2} \sum_{p,t} \sum_{q_1,x_1} \sum_{q_2,x_2} S_{pq_1;x_1 t} S_{pq_2;x_2 t} = \frac{1}{2} \sum_{p,t} \left(\sum_{qx} S_{pq;xt} \right)^2 \tag{11}$$

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

$$E_{QT} = \frac{1}{2} \sum_{q,t} \sum_{p_1,x_1} \sum_{p_2,x_2} S_{p_1 q;x_1 t} S_{p_2 q;x_2 t} = \frac{1}{2} \sum_{q,t} \left(\sum_{px} S_{pq;xt} \right)^2 \tag{12}$$

□

In [Hopfield-84][HT-85], 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 $\mathcal{T}$ to denote the *Temperature*, the behavior of x_i can be stated as:

$$x_i = \frac{1}{2} + \frac{1}{2} \cdot tanh \left((\sum_{i \neq j} W_{ij} x_j - I_i) \times \frac{1}{\mathcal{T}} \right) \tag{13}$$

When $\mathcal{T} \longrightarrow 0$, function (13)'s response takes on a step-wise form. These models of [HT-85] 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.

Definition 7.4 At a finite $\mathcal{T}$, x_i (of eq. 13) can be replaced by an MFT variable υ_{ia}, then differentiating to obtain the Potts MFT equations:

$$\upsilon_{ia} = -\frac{1}{\mathcal{T}} \cdot \frac{\partial E[\vec{\nu}]}{\partial \nu_{ia}} \tag{14}$$

$$\nu_{ia} = \frac{e^{\upsilon_{ia}}}{\sum_b e^{\upsilon_{ib}}} \tag{15}$$

Which enforces

$$\sum_a \nu_{ia} = 1 \quad \forall i \, . \tag{16}$$

Where a denotes the vector components or all the elements of the group. Also, we could state that $\nu_{ia} \approx \Pr(\text{neuron(i)} \in \text{state(a)})$. Equations (14, 15) are computed in an iterative manner. □

Definition 7.5 In [PS-89] and [van-den-Bout.90], 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. □

Definition 7.6 To carry out the minimization of the energy functions of definition (7.1) we need to convert the neural variables $\{0, 1\}$ of S_i to their corresponding *mean field variables* at $\mathcal{T}$. These correspondences are derived (see [PS-92]) to be:

$$V_{pq;xt} \approx < S_{pq;xt} >_{\mathcal{T}} \; ; \tag{17}$$

which is the thermal average of the original binary spines, with the following multi-state MFT equations (see (14, 15) for a general format):

$$U_{pq;xt} = -\frac{1}{\mathcal{T}} \cdot \frac{\partial E}{\partial V_{pq;xt}} \tag{18}$$

$$V_{pq;xt} = \frac{e^{U_{pq;xt}}}{\sum_{x't'} e^{U_{pq;x't'}}} \tag{19}$$

□

The MFT equations of υ and ν are solved *iteratively* either synchronously or asynchronously, under $\mathcal{T}$. 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** $S^{(T)}_{pq;t}$:

Definition 7.7

$$S_{pq;xt} = S^{(X)}_{pq;x} \cdot S^{(T)}_{pq;t} \tag{20}$$

With separate Potts conditions:

$$\sum_x S^{(X)}_{pq;x} = 1 \tag{21}$$

$$\sum_t S^{(T)}_{pq;t} = 1 \tag{22}$$

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 (21, 22) conditions can be re-stated as follows:

$$\sum_x S^{(X)}_{i;x} = 1 \tag{23}$$

$$\sum_t S^{(T)}_{i;t} = 1 \tag{24}$$

□

During annealing, as we go from a higher temperature to a lower one, a *phase transition* is passed at ($\mathcal{T} = \mathcal{T}_c$), where $\mathcal{T}_c$[‡] is the *critical temperature*. The value of $\mathcal{T}_c$ set the scale or the range of the value of $\mathcal{T}$, and it is estimated by expanding 19 around some trivial fixed-point such as

$$V^{(0)}_{pq;xt} = \frac{1}{N_x N_t} \quad , \text{ at } \mathcal{T} = 0 \ . \tag{25}$$

Also $\mathcal{T}_c$ 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 [PS-89].

Definition 7.8 *Relative Clamping*: 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 amounts to making the following notational replacement:

$$i \longrightarrow (j, k) \quad ; \tag{26}$$

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. □

Definition 7.9 To deal with events that take place over two consecutive hours, what [PS-89] called effective *time*-neurons $\tilde{S}^{(T)}_{i;t}$ are introduced:

$$\tilde{S}^{(T)}_{i;t} = \sum_{k=0}^{g_i - 1} S^{(T)}_{i;t-k} \quad ; \tag{27}$$

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. □

Utilizing definition (7.9), constraints (10, 11, 12) can be put into the following form:

Definition 7.10

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

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

$$E_{QT} = \frac{1}{2} \sum_{q,t} \sum_{i \neq i'} \delta_{q(i),q(i')} \tilde{S}^{(T)}_{i;t} \tilde{S}^{(T)}_{i';t} \tag{30}$$

□

[‡]See pp 29-31, [HKP-91] for an approximate value in the case of Ising ferromagnets.

Definition 7.11 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.

$$\delta_{r,s} = \begin{cases} 1 & \text{if } r = s \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

□

Definition 7.12 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. This is a common practice with those classes where the students have many choices. To carry out such an extension to quasi-classes 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.

$$\delta_{q(i),q(i')} \longrightarrow \Gamma_{q(i),q(i')} \ . \tag{32}$$

□

Definition 7.13 From definitions (7.8, 7.12), 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:

$$\tilde{\delta}_{j,k} = \delta_{j,k}(1 - \delta_{j,0}\delta_{k,0}) \ . \tag{33}$$

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

Definition 7.14 Utilizing the concepts of the above definition (7.12, 7.13), the collision terms of definition (7.10) are extended to ensure that all the hard constraints are satisfied (or the solutions obtained are legal). The extension takes the following forms:

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

$$E_{XT} \text{ (form 2) } = \frac{1}{2} \sum_{j \neq j'} \sum_{kk'} \sum_{x} S^{(X)}_{jk;x} S^{(X)}_{j'k';x} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} + \frac{1}{2} \sum_{j} g_j \sum_{k \neq k'} \sum_{x} S^{(X)}_{jk;x} S^{(X)}_{jk';x} \tag{35}$$

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

$$E_{PT} \text{ (form 2) } = \frac{1}{2} \sum_{j \neq j'} \sum_{kk'} \tilde{\delta}_{p(j,k),p(j',k')} \sum_{t} \tilde{S}^{(T)}_{j;t} \tilde{S}^{(T)}_{j';t} \tag{37}$$

$$E_{QT}\ \text{(form 1)} \ = \frac{1}{2}\sum_{qt}\left[\sum_{jk}\delta_{q,q(j,k)}\tilde{S}^{(T)}_{j;t}\right]\left[\sum_{j'\neq j}\sum_{k'}\Gamma_{q,q(j',k')}\tilde{S}^{(T)}_{j';t}\right] \tag{38}$$

$$E_{QT}\ \text{(form 2)} \ = \frac{1}{2}\sum_{j\neq j'}\sum_{kk'}\Gamma_{q(j,k),q(j',k')}\sum_{t}\tilde{S}^{(T)}_{j;t}\tilde{S}^{(T)}_{j';t} \tag{39}$$

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

The factorization idea of (dfn 7.14) 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 above equations (23, 24) .

Substituting V for S in (23, 24), the *mean field* equations at a finite $\mathcal{T}$, 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:

Definition 7.15

$$U^{(X)}_{jk;x} = -\frac{1}{\mathcal{T}}\cdot\frac{\partial E}{\partial V^{(X)}_{jk;x}} \tag{40}$$

$$U^{(T)}_{j;x} = -\frac{1}{\mathcal{T}}\cdot\frac{\partial E}{\partial V^{(T)}_{j;t}} \tag{41}$$

$$V^{(X)}_{jk;x} = \frac{e^{U^{(X)}_{jk;x}}}{\sum_{x'} e^{U^{(X)}_{jk;x'}}} \tag{42}$$

$$V^{(T)}_{j;t} = \frac{e^{U^{(T)}_{j;t}}}{\sum_{t'} e^{U^{(T)}_{j;t'}}} \tag{43}$$

The iterations of the *MF* dynamics over equations (40 - 43) can either be **synchronous** or **serial** (see [PS-92]). □

Definition 7.16 *Synchronous Updating*: A single sweep consists of first computing all the local fields, and then updating all the neurons. That means do equations (40,41) then perform equations (42,43). □

Definition 7.17 *Serial Updating*: For each neuron $S^{(X)}_{jk;x}$ and/or $S^{(T)}_{j;t}$, the *local field* is computed immediately before the corresponding *neuron state* is updated. That means either do equation (40) followed by (42) and/or do equation (41) followed by (43). □

Next, we need to define 2 important **soft constraints** encountered when scheduling classes.

Definition 7.18

- *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 ($\tilde{S}^{(T)}_{j;dh}$ is reduced to a D-type neuron by summing over all hours of the day):

$$S^{(D)}_{i;d} = \sum_h \tilde{S}^{(T)}_{j;dh} = g_j \cdot S^{(T)}_{j;dh} \tag{44}$$

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

$$E_{QSD} = \frac{1}{2} \sum_{qsd} \left[\sum_{jk} \sum_{h} \delta_{q,q(j,k)} \delta_{s,s(j,k)} \tilde{S}^{(T)}_{jk;dh} \right]^2 \tag{45}$$

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)

$$E_{QSD} = \frac{1}{2} \sum_{d} \sum_{j \neq \hat{j}} \sum_{kh} \sum_{\hat{k}\hat{h}} \delta_{q(j,k),q(\hat{j},\hat{k})} \delta_{s(j,k),s(\hat{j},\hat{k})} \hat{S}^{(T)}_{j;dh} \hat{S}^{(T)}_{\hat{j};d\hat{h}} \tag{46}$$

Now for some day (d) of the week:

$$E_{QSD} = \frac{1}{2} \sum_{j \neq \hat{j}} \sum_{k\hat{k}} \delta_{q(j,k),q(\hat{j},\hat{k})} \delta_{s(j,k),s(\hat{j},\hat{k})} \sum_{d} S^{(D)}_{j,d} S^{(D)}_{\hat{j},d} \tag{47}$$

- *Glueing*: The schedule should have as few "holes" as possible; that is lessons should be glued together.

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

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

The energy term of (48) 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:

$$E_{QDH} = -\alpha \sum_{\hat{j} \neq j} \sum_{k,\hat{k}} \Gamma_{q(j,k),q(\hat{j},\hat{k})} \sum_{d,h>1} \tilde{S}^{(T)}_{j;dh} \tilde{S}^{(T)}_{\hat{j},d(h-1)} \tag{49}$$

The presence of the $(-)$ sign in front of (48, 49) 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 (10,11,12). This (α) takes on a value between zero and 1.

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

$$E_{hard} = combine(E_{XT}, E_{PT}, E_{QT}) \tag{50}$$

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

$$E_{soft} = combine(E_{QSD}, E_{QDH}) \ . \tag{51}$$

The total energy E to be minimized is then:

$$E = combine(E_{hard}, E_{medium}, E_{soft}) \ . \tag{52}$$

□

7.4 The generic Potts Algorithm

7.4.1 Some general comments on the method

Let (N) denotes the total number of the utilized neurons. At high $\mathcal{T}$, 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 $\mathcal{T}$, 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 [HT-85]).

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 *mean field annealing* method can be a powerful and general technique which has found many applications in combinatorial optimization.

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 $\mathcal{T}$.
- The annealing schedule details, i.e. the initial temperature $\mathcal{T}(0)$, the rules for deciding when to reduce $\mathcal{T}$ and by how much, and the termination criteria.

A quantity introduced by [PS-89] is called *saturation*, Σ, defined as

$$\Sigma = \frac{1}{N} \sum_{ia} \nu_{ia}^2 \ , \tag{53}$$

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

7.4.2 The Algorithm

- Choose a problem $\Longrightarrow$ (mapped into) $\{w_{ij}\}$.
- Find the approximate phase transition temperature by linearizing equation (15), see also above section.
- Add a self-coupling β-term if necessary. Neural-wise, this corresponds to a feedback connection from a neuron to itself.
- Initialize the neurons ν_{ia} with $\frac{1}{N}\pm$ *random values*; and set $\mathcal{T}(0) = \mathcal{T}_c$.
- Until ($\Sigma \approx \frac{1}{N} \sum_{ia} \nu_{ia}^2 \geq 0.99$) do:
  - At each $\mathcal{T}(n)$, update all ν_{ia} via υ_{ia}:
    * $\upsilon_{ia} = -\frac{1}{\mathcal{T}(n)} \cdot \frac{\partial E[\nu]}{\partial \nu_{ia}}$.
    * $\nu_{ia} = \frac{e^{\upsilon_{ia}}}{\sum_b e^{\upsilon_{ia}}}$. $(\equiv f_{ia})$
  - $\mathcal{T}(n+1) = 0.9 \times \mathcal{T}(n)$.
- After $\Sigma \geq 0.99$ is reached perform a *greedy heuristics* if needed to account for possible imbalances or rule violations.

7.4.3 Some implementation details

The general idea is to initialize the system close to the symmetric state at a temperature near $\mathcal{T}_c$, then, allowing the system to converge to a fixed point at each temperature, anneal (i.e. reduce $\mathcal{T}$) 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 [Simmen-93.1] investigated in depth these and most of the other issues pertaining to the Potts model, using the TSP as a test case.)

- **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 $\ll 1$.

[§]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.

[¶]These follow the method used in [PS-89] but with some additions/improvements done by the authors -see algorithm above.

- **Convergence** At each $\mathcal{T}$ the ν_{ia} variables are updated as above or by the following rule[‖]:
$$\nu_{ia}^{(n+1)} = \nu_{ia}^{(n)} + \gamma(f_{ia} - \nu_{ia}^{(n)}), \quad 0 < \gamma \leq 1 \ . \tag{54}$$
Setting $\gamma = 1$ recovers the dynamics of [PS-89], and the $\gamma \to 0$ limit gives the first order Euler integration method for the differential equation:
$$d\nu_{ia}/dt = f_{ia} - \nu_{ia} \ . \tag{55}$$
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 $\mathcal{T}$, $\mathcal{T} \mapsto \mathcal{T} \cdot \mathcal{T}_r$, where $\mathcal{T}_r \in (0,1)$ controls the cooling rate. Roughly speaking, $\mathcal{T}_r$ 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 $\mathcal{T}$ to fall below some $\mathcal{T}_{min}$, ($\mathcal{T}_{min} \ll \mathcal{T}_c$), without ever satisfying the $\Sigma > \Sigma^{thresh}$ criterion. $\mathcal{T}_{min}$ was set to 10^{-3}.

8 Our study case using the Potts model

8.1 Notation and Definitions (The Second part)

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.

Definition 8.1 *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 :

[‖]Thanks to M. Simmen for pointing this out. Also see [PS-89]

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. □

Definition 8.2 Set (ϵ) and (c) to some desired values depending on the time period of the set of classes, now, the above *effective* t-neurons can be defined as follows:

$$\tilde{S}^{(\vec{r},T)}_{j;d(t)} = \sum_{k=0}^{g_j - c} S^{(\vec{r},T)}_{j;d(t-k+\epsilon)} \tag{56}$$

$\tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)}$ 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

$$\tilde{S}^{(\vec{r},T)}_{j;d(t)} = S^{(\vec{r},T)}_{j;d(t)}. \tag{57}$$

□

Definition 8.3 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 in definition (7.18) 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.

□

8.2 Constraints

The study case dealt with here is *course scheduling* at the university level for both graduate and undergraduate students. Given the various student and course section information, a host of constraints need to be satisfied for any successful scheduling of this nature. In addition to those previously stated in this report, here is a sample (*hard and soft*):

1. Section must be open. (*hard*)
2. Student must meet eligibility criteria for the section. (*hard*)
3. Section meeting time may not conflict with other "enrolled" sections. (*hard*).
4. An enrollment balance by gender should be obtained in Honors seminar sections. (*soft*)
5. For each student, balance MWF and $TuTh$ sections. (*hard*)
6. Do not enroll athletes in courses that conflict with their sport practice time. (may be *hard* or *soft*, depending on the sport)
7. Balance enrollment in multi-section courses. (*soft*)
8. Enroll students in ALL parts of a course when associated sections (i.e. recitation, labs) are required. (*hard*)
9. Optimize 1st choice course enrollments by attempting all 1st choice courses for all students before attempting 2nd and 3rd choices. (*soft*)
10. A student with mapping status of *Req* should occupy a class seat over a student with a status of priority (χ). (*hard*)
11. Class size/capacity to be observed. (*hard*)
12. Distance/building preference or department-to-building distance (*soft*).
13. Various preferences of students or/and professors have to be considered. (*soft*).

Other constraints that may need to be taken into account during the scheduling process:

- It is not the case, in general, that all classrooms are available for all subjects. In reality, many subjects require a special purpose rooms.
- Some classes are taught for two or more hours in a row.
- Group formation: for some subjects the classes are broken up into option groups temporarily forming new classes.
- The schedule should have as few "holes" as possible; that is classes should be "glued" together.
- Student with a "required" mapping status should occupy a class seat over a student with other status.
- Two courses of the same topic are scheduled on two different working days (may be).
- The predecessor-successor relation between the subjects/topics is satisfied.
- Classes may be designated as clashing with other classes because of common students.
- The timetable may consist of an arbitrary number of periods per day.
- Some rooms may be declared unavailable during any time period.
- Class subject combinations may request a preference for certain periods.
- Individual professor and class preferences may be specified for specific rooms even if they are taken from a group or rooms.
- Some classes may be grouped with others.
- Daily limits must be specified to restrict professors from taking more than a specified number of classes per day.
- In order to schedule a number of classes at the same time, it is possible to link class tuples together. In this way, the cost of a move is the sum of the individual costs of each class tuple, however, they are all moved at the same time.

8.2.1 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 $\{D_1, D_2\}$.

Definition 8.4 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:

$$E^r_{QSD} = \frac{1}{2}\sum_{j\neq\hat{j}}\sum_{k\hat{k}}\delta_{q(j,k),q(\hat{j},\hat{k})}\delta_{s(j,k),s(\hat{j},\hat{k})}\sum_{d_1,d_2}S^{(D)}_{j,d_1}S^{(D)}_{\hat{j},d_2} \tag{58}$$

$$E^r_{QSD_1} = \frac{1}{2}\sum_{j_1\neq\hat{j_1}}\sum_{k_1\hat{k_1}}\delta_{q(j_1,k_1),q(\hat{j_1},\hat{k_1})}\delta_{s(j_1,k_1),s(\hat{j_1},\hat{k_1})}\sum_{d_1}\sum_{i\neq\hat{i}}S^{(D)}_{i,d_1}S^{(D)}_{\hat{i},d_1} \tag{59}$$

$$E^r_{QSD_2} = \frac{1}{2}\sum_{j_2\neq\hat{j_2}}\sum_{k_2\hat{k_2}}\delta_{q(j_2,k_2),q(\hat{j_2},\hat{k_2})}\delta_{s(j_2,k_2),s(\hat{j_2},\hat{k_2})}\sum_{d_2}\sum_{i\neq\hat{i}}S^{(D)}_{i,d_2}S^{(D)}_{\hat{i},d_2} \tag{60}$$

□

8.2.2 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.*

Definition 8.5

$$E^{\vec{r}}_{d(XT)} = \frac{1}{2}\sum_{d}\sum_{xt}\sum_{\dot{r}b}\sum_{\ddot{r}a}S^{(\vec{r},X[T])}_{\dot{r}b;d(x)[d(t)]}S^{(\vec{r},X[T])}_{\ddot{r}a;d(x)[d(t)]} \tag{61}$$

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

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

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:

$$E^{\vec{r}}_{d(T)} = \frac{1}{2}\sum_{j\neq\hat{j}}\sum_{d(t)}\tilde{S}^{(\vec{r},T)}_{j;d(t)}\tilde{S}^{(\vec{r},T)}_{\hat{j};d(t)} \tag{63}$$

□

8.2.3 Eligibility for Enrollment

Definition 8.6 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 :

$$E_{QR} = \frac{1}{2}\sum_{qr}\sum_{jk}\sum_{i}\delta_{q,q(j,k)}\delta_{r,r(i)}S^{X}_{ij;x(i)}|(EC \in \{\mathit{0},\mathit{1}\}) \tag{64}$$

$$\sum_{i} O_{q(i)} = 1 \Rightarrow \text{ Test: } \{EC(R) + \text{ satisfy-constraints } (Hard, Soft)\} \tag{65}$$

□

8.2.4 Course Pre-requisite and Subsequent Assignment

Utilizing the link concept ($\mathbf{lin(s,s')}$) between two different subjects (see definition 8.3), a general energy assignment can be put together as follow:

Definition 8.7 Let (i) stands for the situation at hand (also can be viewed as a space-time event).

$$E_{link(1)} = -\gamma\sum_{is}\left[\sum_{jk}\delta_{s,s(j,k)}\tilde{S}^{(X[T])}_{ln(s,s');i}\right]^2 \tag{66}$$

$$E_{link(2)} = -\gamma\sum_{s}\sum_{j\neq j'}\sum_{k\neq k'}\delta_{s(j,k),s(j',k')}\sum_{i}\tilde{S}^{(X[T])}_{ln(s(j,k),s(j',k'));i}\tilde{S}^{(X[T])}_{ln(s(j,k),s(j,k'));i} \tag{67}$$

$$E_{link(3)} = -\gamma\sum_{j\neq j'}\sum_{k\neq k'}\delta_{s(j,k),s(j',k')}\tilde{S}^{(X[T])}_{ln(s(j,k),s(j',k'));i}\tilde{S}^{(X[T])}_{ln(s(j,k),s(j,k'));i} \tag{68}$$

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

8.2.5 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:

$$E_{enroll(1)} = \frac{1}{2}\sum_{s}\sum_{jk}\sum_{i}\delta_{s,s(j,k)}\delta_{r,r(i)}\sum_{t}S^{(T)}_{jk;t} \tag{69}$$

$$E_{enroll(2)} = \frac{1}{2}\sum_{si}\sum_{t\neq t'}\left[\sum_{jk}\delta_{s,\dot{s}(j,k)}\delta_{r,r(i)}S^{(T)}_{jk;t}\right]\left[\sum_{jk'}\delta_{s,\ddot{s}(j,k')}\delta_{r,r(i)}S^{(T)}_{jk';t'}\right] \tag{70}$$

$$E_{enroll(3)} = \frac{1}{2}\sum_{k\neq k'}\delta_{s,\dot{s}(j,k)}\delta_{s,\ddot{s}(j,k')}\sum_{t\neq t'}S^{(T)}_{jk;t}S^{(T)}_{jk';t'} \tag{71}$$

8.2.6 Lecture hall capacity

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

$$\forall x \sum_i ||x(i)| - \sum_j \sum_k S^{(X)}_{jk;x(i)}| \geq 0 \ . \tag{72}$$

8.2.7 Student status and priority mapping

$$E_{assign} = \frac{1}{2} \sum_{m,\hat{x},x} \sum_{i \neq i'} S^{(X)}_{\hat{x};x} \left(S^{(R)}_{r(i);(\hat{x}(m) \in x(m))} |(M = Req) + S^{(R)}_{r(i');(\hat{x}(m) \in x(m))} |(M = \chi) \right) \tag{73}$$

8.3 On the Energy Equation of the Potts Model

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

$$E = E_A + E_B + E_C + \ldots \tag{74}$$

In [PS-89], 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 non-Euclidean type CO problems such as scheduling.

Observation 8.8 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:

$$-\frac{\beta^{(T)}}{2} \sum_j \sum_t [S^{(T)}_{j;t}]^2 - \frac{\beta^{(X)}}{2} \sum_{jk} \sum_x [S^{(X)}_{jk;x}]^2 \tag{75}$$

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. $\square$

8.4 Some Analysis of the MFA Performance on the TTS

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, which empirically have an overall $O(N^2)$ running time [Johnson-90]. 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 these constraints are hard, but the majority are classified to be soft. This later point can be compensated for by more stringent annealing parameters, but at the cost of longer simulation times. No attempts have been make to carry this out. Also, we observe that the overall quality of the solution degrades as the problem size increases. This degradation moves from soft constraint violation to major violations of hard constraints, and hence, illegal timetables. From various correspondences with Andrew Gee, I'd like postulate the following as an observation:

Observation 8.9 We observe that the NP-completeness of the problem (the timetabling, that is) is manifested in the fact that the constraint polytope is non-integral. Furthermore, for a decent number of constraints, there will be many (probably in the thousands) time more non-integral vertices than 0-1 vertices. [Note here, since there is no objective, we just have to find any 0-1 point within the polytope to get a valid timetable]. So, chances are that as we use some annealing mechanism to derive the state vector towards the boundary of the polytope , it's going to settle into a non-integral vertex, which is no use to anyone. (Actually, Peterson alludes to these states in his paper with Gislen, claiming that they are easily decomposed into two good solutions!) Basically, there is no getting around this. Thinking of it from a polyhedral perspective, we're trying to find the one integral vertex in several thousands non-integral ones, and we're just driving the state vector in a random direction towards the polytope boundary: well, chances that it's not going to work. □

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 convergence to the optimal solutions for asymptotically slow annealing, such results is not true for MFA algorithms, because MFA is an approximation.

9 Highlights of Mean Fields Annealing

9.1 Brief Background

To get around the difficulties arising from Hopfield-Tank (HT) algorithm, one way forward is to reduce the computational burden being placed on the network:

Constrain each neuron representing , for example, a class-professor to be "on" only once (each class-professor to take place only once during a certain time period.) by enforcing:

$$\sum_a V_{ia} = 1 \;\; \forall i, V_{ia} = 1/0 \; ; \tag{76}$$

saying that event i to take place during interval a or i is the a-th event to occur in the schedule, rather than relying on an energy penalty term to try to do this (as is done in HT).

9.2 MFA variables: Ising to the Potts model

9.2.1 The Ising Case

The Ising model describes a magnetic system in terms of binary spins $s_i \in \{-1, 1\}$. The system is governed by the energy function

$$E = -\frac{J}{2}\sum_i s_i s_{i+1} \quad , \tag{77}$$

where nearest neighbors interact pairwise with a constant attractive (if $J > 0$) coupling of strength J. The lowest energy state is reached by iterative updating of

$$s_i = sgn[J(s_{i-1} + s_{i+1})] \ , \tag{78}$$

which leads to a state where all spins point in one of two possible directions.

9.2.2 The Potts Representation

Restricting the state space of the neurons such that exactly *one neuron* at every site is allowed to be on, can be written as:

$$\sum_a s_{ia} = 1 \quad \forall i \ . \tag{79}$$

Thus for every i, s_{ia} should be 1 for only one value of a, and 0 for the rest.

Choose a k-dimensional vector representation of the neuron states: The a^{th} possible state of the neuron vector $\vec{s_i}$ associated with the node i is represented by the unit vector in the direction of a, so we end up with K different states, $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$, $\ldots$, $(0, 0, \ldots, 1)$ are all normalized and mutually orthogonal. The number of states available at every node is thereby reduced from 2^K to K. Technically, we now have a K-state Potts model.

The explicit enforcement of the following equality is referred to as *neuron normalization*:

$$\sum_{a \in A} V_{ia} = 1 \ , \tag{80}$$

where A is a set of states, in the MFA algorithm without any need for a penalty term.

9.3 Mapping Timetabling onto the Potts Neural Nets

9.3.1 The Relationship Between the Key Timetabling Entities

Our problem consist of the following sets: teachers (P), classes (C), students (S), rooms (R), and time periods ($\mathcal{I}$).

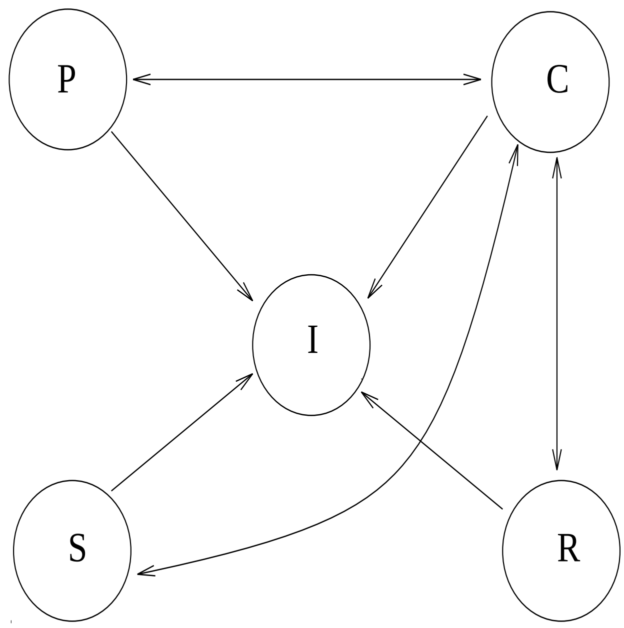

Figure 4: A one-way arrow indicates that the "bottom" entity depends on the "top" one. A two-way arrow indicates a mutual dependency between the two entities.

9.3.2 Neural Mapping

Involve mapping the problem onto the Potts model.

- These multi-state neurons, forming the network, represent different activities of the timetabling problem.
- Construct the constraints of the problem using these neurons.
- Derive those MF equations controlling the dynamics of the net:
  - Introduce the MF variables , for example,

$$V_{pq;xt} =< S_{pq;xt} >_T \quad . \tag{81}$$

  - Start off with the partition function:

$$Z = \sum_{[\vec{s}]} e^{-E[\vec{s}]/T} \, . \tag{82}$$

  - Re-write Z in terms of a multidimensional integral over continuous vectors $(\vec{u}_i, \vec{v}_i)$.
  - The MFT approximation to $< \vec{s}_i >$ is given by the value $\vec{u}_i$ at a saddle-point equations:

$$\frac{\partial E}{\partial v_{ia}} = 0 \tag{83}$$

and

$$\frac{\partial E}{\partial u_{ia}} = 0 \ , \tag{84}$$

generating a set of self-consistency equations in the MF variables:

$$u_{ia} = -\frac{1}{T}\frac{\partial E[\vec{v}]}{\partial v_{ia}} \ , \tag{85}$$

and

$$v_{ia} = \frac{e^{u_{ia}}}{\sum\limits_{b} e^{u_{ib}}} \ . \tag{86}$$

Where a denotes the vector components.
For events (pq, xt), we get the following MF variables:

$$u_{pq,xt} = -\frac{1}{T}\frac{\partial E}{\partial v_{pq,xt}} \ , \tag{87}$$

and

$$v_{pq,xt} = \frac{e^{u_{pq,xt}}}{\sum\limits_{x't'} e^{u_{pq,x't'}}} \ . \tag{88}$$

From the later follows, that $\sum\limits_{a} v_{ia} = 1$. One can think of the mean field v_{ia} as the probability for the Potts neuron i to be in state a.

- Iterate $v_{pq,xt}$ equation with annealing, i.e. starting at a high temperature, and successively lowering it in the course of the process. A phase transition is passed at $T = T_c$.
- Estimate the critical temperature T_c, which sets the scale of T, by expanding the above $v_{pq,xt}$ equation around the trivial fix-point : at $T = 0$

$$v^{(0)}_{pq,xt} = \frac{1}{N_x N_t} \ . \tag{89}$$

10 Timetabling and Simulated Annealing (SA)

Simulated annealing has been widely used for tackling different combinatorial optimization problems, particularly academic scheduling. The results obtained depend heavily on the cooling schedule used. We initially used the most commonly known and used schedule, which is the geometric cooling schedule, but later tried the geometric re-heating schedule based on cost, due to Abramson and colleagues [Abramson-94].

A comprehensive discussion of the theoretical and practical details of simulated annealing is given in [van.Laarhoven-87]. It suffices here to say that the elementary operation in the Metropolis method for a combinatorial problem such as scheduling is the generation of some new candidate configuration, which is then automatically accepted if it lowers the solution cost (C), or accepted with probability $exp(-\Delta C/(kT))$, where k is the Boltzmann constant and T is the temperature, if it would increase the cost by ΔC. Thus the technique is essentially a generalization of the local optimization strategy, where, at non-zero temperatures, thermal excitations can facilitate escape from local minima.

10.1 Some Advantages of SA

1. Relatively easy to implement.
2. Applicable to almost any combinatorial optimization problem.
3. It usually provides reasonable solutions, depending on the cooling schedule(s) used.

10.2 The Algorithm

The following is a sketch of the SA algorithm with a "generic" cooling schedule: Let $\mathcal{N}_s$ be the neighborhood of configuration s. k is the Boltzmann constant = 1.

1. Generate an initial solution (configuration) s .
2. Compute cost of s : $C(s)$.
3. Compute initial temperature T_0.
4. Set the temperature $T = T_0$.
5. While <u>stop criterion</u> is not satisfied do:
   (a) Repeat <u>Markov chain length</u> (M) times:
      i. Select a configuration $s' \subset \mathcal{N}_s$.
      ii. Let $\Delta = C(s') - C(s)$.
      iii. If ($\Delta \leq 0$ {downhill move}), set $s = s'$.
      iv. If ($\Delta > 0$ {uphill move}), set $s = s'$ with a probability $= e^{-\Delta/(kT)}$.
   (b) Reduce (or update) temperature T.
6. Return the solution s.

10.3 Mapping of Timetabling Into Annealing Algorithm

The most obvious mapping of timetabling (TTS) into the SA algorithm involves the following constructs:

Let max(P) = maximum number of classes assigned to any professor.

1. a **state** is a timetable containing the following sets:
   - P: a set of professors.
   - C: a set of classes.
   - S: a set of students.
   - R: a set of classrooms.
   - $\mathcal{I}$: a set of time periods.
2. a **cost** or "energy" E(P, C, S, R, $\mathcal{I}$) such that:
   - E(P): is the cost of assigning more than max(P) classes to the same professor, plus scheduling one or more courses that causes a conflict in the professor's schedule.
   - E(C): is the cost of scheduling certain classes at/within the same time period in violation of the exclusion constraint, for example.
   - E(S): is the cost of having two or more classes conflict in time; plus cost of having in the schedule one or more classes that really don't meet the student's major, class requested, or course requirements; plus the cost of not having the courses evenly spread over the week, etc.
   - E(R): is the cost resulting from assigning room(s) of the wrong size and/or type to a certain class.
   - E($\mathcal{I}$): is the cost of having more or less time periods than required, plus cost of an imbalanced class assignments (a certain period will have more classes assigned to than others, etc.).
3. a **swap** (or a move) is the exchange of one or more of the following:
   - Class swapping: class c_i with class c_j in the set C with respect to time periods $\mathcal{I}_i$ and $\mathcal{I}_j$, and/or with respect to classroom R_i and R_j, respectively.

   A **swap** which would either decrease or increase the cost of ΔE is accepted with probability $Pr(\Delta E)$. This probability defines a Markov chain on TTS, and have the distribution in which $Pr(TTS)$ is proportional to $e^{-E(TTS)/T}$. As $T \rightarrow \infty$, the distribution gives the limiting probability of being at a certain state of (P, C, S, R, $\mathcal{I}$).

10.4 The Annealing Schedules

Two annealing schedules have been used for our experiments: geometric cooling and an adaptive re-heating as a function of cost (RFC, for short). This later schedule is due to Abramson[Abramson-94].

10.4.1 Geometric Annealing Schedule

The new temperature in the above schedule is usually calculated according to:

$$T^{'} = \alpha T , \tag{90}$$

where α is typically in the range $0.95 \ldots 0.99$. This schedule is used as a base line for comparison of other scheduling schemes.

Advantages:

- Have a good theoretical foundation.
- Proven history of being good in practice.
- Well understood by all researchers working on annealing.

Disadvantages:

The state of the system is not taken into account in any way, and thus can not adapt the intensity of the search depending on the complexity of the problem.

10.4.2 Reheating as a Function of Cost (RFC) Schedule

Some background:

- In combinatorial optimization problems, **phase transitions** can be observed as sub-parts of the problem are resolved. Lister [Lister-93] , in tackling the TSP, and Abramson [Abramson-94] observed that at high temperatures the gross structure of the solution is resolved, and at low temperatures the minute details of the solution are resolved.
- Applying a reheating type procedure, depending on the phase, would allow the algorithm to spend more time in the low temperature phases, thus reducing the total amount of time required to solve a given problem.
- To calculate the temperature at which a **phase transition** occurs, it is necessary to compute what is called the **specific heat** of the substance. A **phase transition** occurs at a temperature T_{msh} when the specific heat is maximal, and this triggers the change in the state ordering.

- If the best solution found to date is poor (i.e. has a high energy or cost) then the super-structure may require re-arrangement. This can be done by raising the temperature to a level which is higher than the phase transition temperature (T_{msh}). Generally, the higher the best cost, the higher the temperature which is required to escape the local minimum.

- T_{msh} is the temperature at which the maximum specific heat occurs, or at which the system changes state.

<u>The schedule</u>:

- Let C_i to denote the cost of configuration i, and k the Boltzmann constant.

- At temperature T, the acceptance probability is:

$$Pr_i(T) = \frac{e^{\frac{-C_i}{kT}}}{\sum_j e^{\frac{-C_j}{kT}}} \ . \tag{91}$$

- For a set of configurations $\mathcal{N}$, the expected cost at temperature T:

$$< C(T) >= \sum_{i \in \mathcal{N}} C_i Pr_i(T) \ . \tag{92}$$

- The expected square cost at T:

$$< C^2(T) >= \sum_{i \in \mathcal{N}} C_i^2 Pr_i(T) \ . \tag{93}$$

- The variance of the cost (or energy) at T is:

$$\sigma^2(T) =< C^2(T) > - < C(T) >^2 \ . \tag{94}$$

- Using the above derivations, the specific heat:

$$T_{sh}(i) = \frac{\sigma^2(T)}{T^2} \ , \text{ where } \ i \in \{1, 2, \ldots, n\} \, . \tag{95}$$

- Now the maximum specific heat:

$$T_{msh} = max(T_{sh}(i)) \ , \text{ where } \ i = 1, 2, \ldots, n \, . \tag{96}$$

- The new temperature is set to:

$$T_{new} = P \cdot C_b(s) + T_{msh} \ . \tag{97}$$

  T_{new} is the reset temperature, P is a constant, and $C_b(s)$ is the current best cost of the given configuration s.

It follows from the Boltzmann distribution (see White [White-84] and lemma 4.1.10 of [GS-92]) that

$$\frac{d}{dT} E_T = \frac{d}{dT} C(T) = \frac{\sigma^2(T)}{T^2} \tag{98}$$

11 More on Simulated Annealing

Simulated annealing is a robust technique and has been applied with success to a wide variety of problems. However, to obtain good results the annealing runs often require a great deal of computer time. [Johnson-89] criticized much of the earlier research for only comparing simulated annealing with the simpler local optimization heuristics (i.e. those using r-opt style moves) and not with the more competitive Lin-Kernighan (LK) heuristics. They undertook a systematic comparison of the above three methods' performance on the graph bisection problem, for low degree random and random geometric graphs. When assessed using equivalent amounts of computer time, simulated annealing emerged a clear winner over simple local optimization for both types of graph. Against the LK-style algorithm however, the new technique (SA) was only slightly better for random graphs and worse for the random geometrical class. A similar investigation is currently underway for the TSP [Johnson-92] and [Johnson-94]; preliminary results [Johnson-90] suggest that, matched for run-time, simulated annealing beats 3-opt (see Table.**??**) but is worse than LK.

In summary, simulated annealing appears competitive with many of the best local optimization methods. Its major drawback is the requirement for long run times. Furthermore, the inherently serial nature of the approach in general precludes any obvious parallel implementation, though several attempts have been made to overcome this problem such as the recent work by Frost [Frost-94].

12 Simulation Issues and Results

Our study case is quite large in comparison to the scheduling case used by Peterson and colleagues [GSP-91]. The largest case they have dealt with involve around 1000 students, 20 different possible majors, and an overall periodic schedules (over weeks). In the case of Abramson and colleagues [Abramson-94], the problems they dealt with are quite small and they state that problems involving more than 300 tuples** are quite difficult to solve.

Our study case involve real scheduling data covering three semesters here at Syracuse University. Those of first and second semester are roughly of the same size: around 13653 students, 2500 of professors, staff, and teaching assistants, 581 rooms (classrooms, seminar rooms and auditoriums), and 3624 classes. With the exception of number of rooms, the third semester data is much smaller. Along with this, we also have utilized the distance (in a form of a distance matrix) between various buildings on main campus. Given the various building preferences of the departments, we have constructed a "finite state automaton" in a matrix format $[I, J]$ between all the academic schools and departments $[I]$ and all the buildings involved in scheduling $[J]$. Using this matrix in conjunction with the real -appropriately scaled- distance, a final state matrix is derived and directly used in the scheduling process.

**A tuple is a combination of a class, teacher, and a classroom. The time periods are assumed to be given.

12.1 On Results obtained via SA and MFA

The main objective here is to generate a solution (i.e. a feasible timetable) subject to all binding (or hard) constraints and to minimize the cost incurred form the medium and soft constraints.

	Total Courses	Scheduled	Not Scheduled	% Scheduled
First Term	4400	2992	1408	68.00
Second Term	4082	2881	1201	70.57
Third Term	709	569	140	80.25

Table 2: Performance figures for three terms using a *rule-based system* only. Both class and students constraints are used.

	Total Courses	Scheduled	Not Scheduled	% Scheduled
First Term	4400	3168	1232	72.00
Second Term	4082	3189	893	78.12
Third Term	709	631	78	89.00

Table 3: Performance figures using a *rule-based system (or expert system)*. The only type of constraints used are those for classes.

	Scheduled	Not Scheduled	% Scheduled
SA (geometric)	2627	1773	59.70
SA (cost-based)	2914	1486	66.22
Mean Field	2151	2249	48.88

Table 4: Performance figures for three algorithms with no pre-processor used. Both students and class constraints are used and the data is that of the first term. NOTE: Annealing is carried out on space AND time. Also, all shown figures are averaged over 10 runs.

12.2 On Results obtained via MFA

To summarize, the results obtained from mean field annealing range from not quite satisfactory to totally unacceptable. Not quite satisfactory in the sense that when dealing with either hard or soft constraints separately (but not at the same time), the resulting timetables do satisfy the given constraints, but depends on how many of these constraints we're dealing with. As the number of constraints increases, the violations start appearing till it is no longer viable to continue with the procedure.

	Scheduled	Not Scheduled	% Scheduled
SA (geometric)	4280	120	97.27
SA (cost-based)	4333	67	98.47
Mean Field	3985	415	90.56

Table 5: Performance figures for three algorithms using an expert system as a pre-processor. Both students and class constraints are used and the data is that of the first term. NOTE: Annealing is carried out on space AND time. Also, all shown figures are averaged over 10 runs.

When running the method on a combined hard and soft constraints, violations start occurring at a higher frequency and many involves the hard constraints, clearly, yielding illegal timetables.

13 Summary and Conclusion

This part outlines the basic method and the scheduling constraints. An academic scheduling problem that is based upon equality constraints is considered. Two methods are used to apply our data and produce the target schedules. This target need to be a valid timetable of classes, professors, and students with no violations at all of the hard constraints and preferrabaly satisfying most , if not all, of the soft constraints. Using simulated annealing with an adaptive re-heating schedule based on cost and with no pre-processing, over 60 out of each 100 timetables produced, do satisfy all of the hard constraints and most of the soft constraints used. On the other hand, mean field annealing produces less than 50 legal timetables out of each 100 timetables produced. The rule-based system which is used as pre-processor for each method contributed greatly to the improvement of the simulated annealing results as well as those of mean field annealing. The use of pre-processing with MFA improved the results to around 90 percent; that is 90 out of 100 schedules do satisfy all of the hard constraints. While the use of pre-processing with SA yielded a 98 percent accuracy; that is 98 out of 100 schedules are legal (satisfying all the hard constraints and most of medium and soft ones).

In lights of these results, the figures reported in Table-**??** seem to be quite consistent with the figures of Table-**??** of the two methods, simulated annealing and MFA on the class (represented by the TSP) of non-Euclidean CO problems, which the scheduling problem belongs to.

We plan to carry out more experiments using the techniques reported in this paper, perhaps using an improved pre-processor system.

References

[HKP-91] Hertz, Krogh, and Palmer, *Introduction to the theory of neural computation*, Addison-Wesley, Redwood City, 1991.

[Abramson-94] Abramson, Dang, and Krishnamoorthy, *An Empirical Study of Simulated Annealing Cooling Schedules*, Griffith Univ. report, Nathan, Qld, Aus. 1994.

[Dowsland-93] Dowsland, Kat, Using Simulated Annealing for Efficient Allocation of Students to Practical Classes. Stat and OR Group, EBMS, Univ College of Swansea, UK, 1993.

[TD-94] J. Thompson, and K. Dowsland, "Variants of Simulated Annealing for the Examination Timetabling Problem", Working Paper, Statistics and OR Group, European Business Management School, Swansea SA2 8PP, UK. (1994)

[GT-86] K. Gosselin and M. Truchon, "Allocation of Classrooms by Linear Programming", Journal of Operational Research Society 37, 561 (1986).

[Abramson-94] Abramson, Dang, and Krishnamoorthy, "An Empirical Study of Simulated Annealing Cooling Schedules", Griffith Univ. report, Nathan, Qld, Aus. (1994).

[Lister-93] R. Lister,"Annealing Networks and Fractal Landscapes", Proc. IEEE Int. Conf. on Neural Nets, March 1993, Vol. I, pp 257-262.

[Mouritsen-84] Mouritsen, O. G, Computer Studies of Phase Transitions and Critical Phenomena. Springer-Verlag, Berlin, 1984.

[Montanari] Montanari, U, "Networks of constraints: fundamental properties and applications to picture processing", *Information Sciences* 7 (1974), pp. 95-132.

[Ladkin-Maddux] Ladkin, P.B., and Maddux, R.D., "On binary constraint problems", *Journal of the ACM* 41 (1994), pp. 435-469.

[Mackworth] Mackworth, A. K., "Consistency in networks of relations", *Artificial Intelligence* 8 (1977) pp. 99-118.

[GJ-79] Gary and Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco (1979).

[GS-92] Sorkin, Gregory, *Theory and Practice of Simulated Annealing on Special Energy Landscapes*. PhD Thesis, UC-Berkeley, Dept of EE and CS, 1992.

[Shrijver-86] Shrijver, A, *Theory of Linear and Integer Programming*, John Wiley and Sons, Chichester, UK, 1986.

[Shrijver-95] Shrijver, A, *A Course in Combinatorial Optimization*, to be published in 1995.

[Frost-94] Forest, Richard, *Ensemble of Simulated Annealing*, SDSC, California, 1994.

[van.Laarhoven-87] Van Laarhoven, and Aarts, *Simulated Annealing: Theory and Applications*. D. Reidel, Dordrecht (1987).

[van-den-Bout.90] 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.

[Johnson-89] Johnson, Aragon, McGeoch, and Schevon, Optimization by Simulated Annealing: an Experimental Evaluation, Part I (Graph Partitioning). OR 37, 865-892 (1989).

[Johnson-90] David Johnson, Local Optimization and the TSP, Springer Lecture Notes in CS 443, 446-461, Springer-Verlag, 1990.

[Johnson-92] Johnson, Aragon, McGeoch, and Schevon, Optimization by Simulated Annealing: an Experimental Evaluation, Part III (TSP). Work in progress.

[Johnson-94] David Johnson, Data Structures for the TSP, to appear in the Jr. of Algorithms.

[PS-88] F. Praparata, M. Shamos, *Computational Geometry*, Springer-Verlag, 1988.

[Min-88] Minsky, Marvin and S Papert, *Perceptron* , 1988 edition.

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

[PS-89] Peterson, Söderberg, "A New Method for Mapping Optimization Problems onto Neural Nets", International Jr. of Neural Systems 1, 3 (1989).

[GSP-89.1] Gislén, Söderberg, Peterson, "Teachers and Classes with Neural Nets", Inter. Jr. of Neural Systems 1, 167 (1989).

[GSP-91] Gislén, Söderberg, Peterson, "Scheduling High Schools with Neural Nets", Lund University Preprint LU-TP-91-9, Lund, Sweden (1991).

[Peterson-90] Carsten Peterson, "Parallel Distributed Approaches to Combinatorial Optimization Problems - Benchmark Studies on TSP", Neural Computation 2, 261 (1990).

[PS-92] Peterson, Söderberg, "Artificial neural Nets and Combinatorial Optimization Problems", To appear in *Local Search in Combinatorial Optimization*, edited by Aarts and Lenstra , second half of 1993.

[S-91] Paul Stolorz, "Abusing Statistical Mechanics to do Adaptive Learning and Combinatorial Optimization", Preprint, Santa Fe Institute, Dec 1991.

[HT-85] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", Biological Cybernetics 52, 141 (1985).

[Hopfield-84] J. J. Hopfield, "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).

[Simic-90] P. Simic, "Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimizations", Network 1, 89-103 (1990).

[Simic-91] P. Simic, "Constrained Nets for Graph Matching and Other Quadratic Assignment Problems", Neural Computation, v.3, n.2, 268-281 (1991).

[Wu-82] F. Y. Wu, "The Potts model", *Reviews of Modern Physics* 54, 235-268 (1982).

[CDM-92] A. Colorni, 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.

[Hertz-92] Alain Hertz, "Finding a feasible course schedule using Tabu search", Discrete Applied Mathematics 35, 255-270, 1992.

[Vidal-93] *Applied Simulated Annealing*, edited by Rene Vidal, Lecture Notes in Economics and Mathematical Systems no. 396, Springer-Verlag, 1993.

[Gee-93.1] Andrew Gee, Problem Solving with Optimization Networks, PhD Thesis, The Univ. of Cambridge, Dept. of Engineering, July 1993.

[White-84] S. R. White, "Concepts of scale in simulated annealing". In Proceedings of the IEEE Int. Conference on Circuit Design, pp 646-651, 1984.

[Simmen-93.1] Martin Simmen, PhD Thesis, Univ. of Edinburgh, 1993.

[Simic-93] P. Simic, private communication.

[Simmen-93.2] Martin Simmen, private communication.

[Gee-93.2] Andrew Gee, private communication.

[Peterson-92-93] Carsten Peterson, private communications.

[Carter-93] Mike Carter, private communications.