Advances In Multidisciplinary & Scientific Research
A Multidisciplinary & Interdisciplinary Journal

Vol. 2 No. 2, Issue 1, May 2016

Solving for Computational Intelligence the Timetable-Problem

A.A. Ojugo, D. A. Oyemade, & D. Allenotor
Department of Math/Computer Science
Federal University of Petroleum Resources
Effurun, Warri, Nigeria.
ojugo.arnold@fupre.edu.ng, oyemade.david@fupre.edu.ng, allenotor.david@fupre.edu.ng

ABSTRACT

Timetable problem is a constraint satisfaction problem that aims to resolve timetable scheduling conflicts via models or algorithmic heuristics. The algorithms search through the domain space for the best (goal) state that satisfies all the problem constraints and/or criteria. It aims to guarantee that the reasoning process is explicit enough with a simple structure that conveys enough data about the problem or task to the search algorithm/model. Our study aims at the complete, feasible assignment that satisfies its constraints to yield a valid academic schedule for the University of Benin, Nigeria using 3-variants of simulated annealing: (a) with reheating, (b) with adaptive and (c) with exponential cooling strategies; while adopting a rule-based processor to yield an initial solution for the models. Results obtained were thus: SA with reheating took 6hours and 45minutes to find solution after 8iterations (at best). SA with adaptive cooling (at best) took 8hours after 12iterations; while SA with exponential cooling arrived at solution 2.112seconds after 401 iterations. The convergence time for SA is dependent on parameters such as start population, temperature (initial and final) and random swaps as applied in the cooling strategies. These models yielded a valid schedule for the University of Benin in Nigeria considering student preference as medium constraints of high priority.

Keywords: stochastic, network, function, optimization, search space, solution, models

Aims Research Journal Reference Format:
A.A. Ojugo, D. A. Oyemade, & D. Allenotor (2016): Solving For Computational Intelligence the Timetable-Problem Advances in Multidisciplinary Research Journal. Vol. 2. No. 2, Issue 1 Pp 67-84.

1. INTRODUCTION

The constraint satisfaction problem (CSP) consists of 3-components: (a) variables, (b) values for each variables, and (c) set of constraints for the various variables. Thus, a CSP aims to find a solution that yields a value for each variable so as to satisfy all constraints via search method [35, 43]. A state is an assignment of values to some or all variables, and an assignment is complete if all variable have values. An assignment is partial if some variables have no values; while, a consistent assignment does not violate the constraints. CSP solution must be complete and have a consistent assignment [11, 35].

A constraint is a relation between a local set of variables that satisfies or restricts the values that variables can simultaneously have. For example, the term different(A1,A2,A3) can imply constraints A1, A2 and A3 must take different values. Thus, its solution consists of: (a) each constraint is over local collection of variables, finding a global assignment to all variables that satisfies all constraints are hard, (b) such task is called NP-complete for which the solution must employ a search method that cleverly explores the space for possible assignments of values to all variables, and (c) each of Y variables has E values and EY possible assignments [1, 20].

Examples of CSP include N-Queen task, Sudoku etc. Various search methods are used to find solutions to such tasks, and these include depth search, breadth search, greedy search, iterative deepening, steepest descent, etc. Some CSPs require a solution that maximizes its objective function as applied in scheduling, cryptography, image interpretation to mention a few [17].

While the search for CSPs solution must be feasible (achievable) and optimal (near or closest to the best in the space); some tasks have dynamic feats that make the use of primitive search techniques a bit tedious and inexplicable to resolve. These are best resolved via statistical-optimization methods. These optimization methods search for optimal solution(s) in a given task, chosen from set of possible search space. It aims to relate the problem data as input with uncontrollable feats and/or parameters to be modeled, so as to satisfy all possible constraints and yield an output through its mathematical model/structure that is flexible, easily adapted and robust. These, in time can be easily tuned and is often inspired by biological, stochastic and evolutionary models whose implementation can span across fields such as medicine, electronics, etc. Examples include genetic algorithm, fuzzy logic, particle swarm, gravitational search algorithm etc – and is today, termed soft-computing (SC) [18].

1.1. Classical Problem Domain: Timetabling

[9,26,35,39,41] notes that the timetable problem is a classical CSP, that aims to derive an approximate solution as thus: given a set of lecture classes and their days, student number/enrollment, rooms and their capacities, instructors and their ranks, class types and locations, distances between buildings, priorities of each building for different departments, students and their preferences amongst others – it constructs a feasibly, optimal lecture schedule for students, taken by lecturers that satisfies all hard constraints, and minimizes all soft and medium constraints.

[9] Constraints are more generally grouped into: (a) modular constraints (lectures scheduled to clash, as students chose a subset of options from a given set, (b) time constraint (certain lectures taught by an external instructor are constrained to a particular day), (c) smoothness constraints (reduce maximum number of lectures taken simultaneously for exhaustive smooth room usage), and (d) spread out constraints (lectures belonging to the same course are timetabled on separate days).

TTP is a high-dimensioned, non-Euclidean, multi-search constraint optimization, whose dynamism is an NP-assignment task of timeslots to an event set that is subject to constraints made up of N_i instructors, N_l lectures, N_r rooms and N_s students. It aims to schedule N_i instructor-class pair within N_t timeslots to yield a schedule – such that no instructor or student is in more than one lecture class at a time, and no room is expected to accommodate more than its capacity and more than a lecture at a time [3,6,19,26,39,41].

Various studies and different heuristics have been employed for such scheduling. For medium-sized tasks such as exam/course scheduling in the university – many methods work well; whereas, scheduling in large universities has no particular method yield good results for real-world apps on larger scale, the nearest of which is in [2,4,19,26,39,41]. Its constraints are divided into [20-21, 26] thus:

a. Hard constraints are specific to space and time. They cannot be physically violated. Thus, such event constraints must be satisfied and cannot overlap in time. We can schedule only one class at a time for any student, teacher or room thus: (a) classes taught by the same instructor, (b) classes held in the same room, (c) lecture class or laboratory of the same class, (d) class cannot be assigned to a particular room unless the room capacity is greater than/equal to number of student enrollment for that class, and (e) laboratory classes require a certain type of room.
b. Medium constraints are student preference and they fall between hard and soft constraints – as they are based on time and space conflict that also cannot be physically violated (e.g. a student cannot be in more than one class at same time). However, it is not possible to, and we are not expected to satisfy all the students always. Thus, they can be avoided by making adjustments – so that as we are not expected to satisfy all student-class preferences, in some cases, students can adjust their preferences to satisfy student-class preference where such classes clash or is oversubscribed. Some students will also have carry-overs. These must be catered for. Medium constraints have, though not a high a penalty as hard constraints, attached to them – such that in a final schedule, the penalty of which must be minimized and preferably, reduced to zero where possible. Examples are: (a) avoid time conflicts for lecture and classes with students in common, (b) eligibility criteria for the class must be met, and (c) do not enroll athletes in classes that conflict with their sport practice time.

c. Soft constraints are preferences of time conflicts as they have the lowest priority and penalty associated to them. We aim to minimize such cost, but may not reduce them to zero. Examples are (a) for each student, balance the 3-days (Mon, Wed and Fri) and 2-days (Tue and Thu) schedules, (b) spread lectures over the week, (c) lecture classes may request contiguous time slots, (d) balance the student enrollment in multi-section classes, (e) lunch/breaks may be specified, (f) instructor may request periods in which their lectures and classes are not taught, (g) instructors have preference for specific rooms or types of rooms, and (h) minimize distance between room where classes are taught or assigned and the building housing the home department.

Hard constraints must be satisfied and their associated cost must reduce to zero. Medium and soft constraints are student and teacher preferences that should be satisfied if possible or minimize the cost associated with them. Preferences involving teachers will have higher priority over those of students. A feasible, optimal schedule is one that satisfies all hard constraints, minimizes all medium and soft constraints where they are not satisfied. This study was conducted and achieved via a semi-automated prediction [1,26,39,41]. The cost function measures quality of the current schedule that involves the weighted sum of penalties associated with the different constraint or violation types. The study thus, aims to find the optimal solution using stochastic, evolutionary methods that minimizes the cost function [19-20].

1.2. Stochastic Models and Fitness Function

Most stochastic models use hill climbing technique that yields the following demerits: (a) they are time consuming, (b) their speed often shrink as they approach maxima and thus, get stuck at local minima, (c) requires extra computational power to search its space/domain, and (d) are computationally intensive procedure and thus, expensive in implementation.

A good fitness function determines if an optimal is found, as model learns data relationships and compares predicted to observed values. Its performance measure contributes to fitness value. Study implements a semi-automated prediction with substantial manual effort to iterate final solution. To schedule a certain semester, template of the previous semester is used as part of the input [26].

2. THE RULE BASED SYSTEM

[26] A rule-based system is implemented for 3 reasons: (a) serves as benchmark to measure how well other methods do in comparison, (b) its simplified version yields a sensible choice for moves in models used, rather than choosing completely random swaps, which greatly improves proportion of moves accepted, and (c) used as preprocessor to all other models and to yield a good start solution.

The model consists of heuristic and conventional recursion routines to assist in carrying out class/lecture assignments, as suited for the problem domain at hand where the following holds [26,39,41]:

a. Distance matrix of values between each academic department and every other building under use for scheduling.
b. The class data structure of each class or lecture scheduled anywhere in campus is adopted that is capable of lining with each other
c. The room data structure of each room (regardless of type) involved in the scheduling process such that like classes/lectures and room structures are linked together.
d. Data structure for time periods to keep track of which time slot is occupied or not.
e. Department data structure for department inclusion within other larger departments or colleges
f. Students structure indicating classes or lectures of various degrees of requirements and preferences for each student.

The model's basics function is: given data files of lectures or classes, rooms and buildings, department-to-building distances matrix, student data and the inclusion data – using all these structures above, the system builds an internal database used in carrying out the scheduling process. This process involve a number of essential sub-processes such as checking the distances between buildings, room type and hours occupied, checking and comparing time slots for any specific conflicts, checking rooms for any space conflicts, and keeping track of and updating the hour already scheduled [26,39,41].

It is an iterative approach and the basic procedure for each move is as thus. Lecture scheduling is done by the department, so that each iteration or generation consists of a loop over all the departments. The departments are chosen in order of size, with those having the most classes being scheduled first. The model first scans all currently unscheduled lectures. It then attempts to assign them to the first unoccupied rooms and timeslots that satisfy the rules governing constraints. Since the constraints for capacity of rooms are very difficult to satisfy, larger lectures are scheduled first to avoid not having lectures with larger number of students without appropriate room capacity to cater for them [19, 26,39,41].

In most cases, only rooms and timeslots satisfying all the rules will already by occupied by previously scheduled lecture classes; And in such case, the will system attempt to move one of these lecture or class into a room and timeslot, to allow the unscheduled class or lecture to be scheduled. Next, the system searches through all scheduled lecture or class and selects those with higher cost by checking the medium and soft constraints (such as how close a room size matches lecture class size, how many students have time conflicts with lectures clashing, which sporting activity clashes with lecture time and how many students are affected, whether the lecture is in a preferred time slot or building, and so on. Selecting threshold values for defining what is considered a high cost for each case is subjective procedure – though straightforward to choose a reasonable value. When a poorly scheduled lecture class is identified, the model searches the space and maps and/or swaps it into a more comfortable position or timeslot – so that the hard constraint are still satisfied, but the overall cost of the medium and soft constraints are reduced [26,39,41].

Room swapping process continues, provided all the rules are satisfied and no cycling (swapping of the same lecture) occurs. Once all the departments have been schedule, iteration cycle is complete. The system continues in this process until complete iteration yields no further change in a schedule. There are many rules dealing with the space and hour continuum, room type, and priority of room. Many are quite complex, but the basic rules such as those implementing hard constraints are quite straightforward [13, 16, 19, 26-27, 39,41, 16,20,22,26,39,41]

Rules adopted for the states:

a. IF room (capacity) > class (space requested) AND {no time conflict in room} THEN assign the room to the class.
b. IF student (capacity) > room (space requested) THEN assign NOT room to lecture class.
c. IF {(Lecturer/Instructor = (Professor > Reader > Senior Lecturer > Lecturer1 > Lecturer2 > Assistant lecturer)} AND {no time conflict for room} THEN (allot space-requested to senior lecturer as preference is to most senior).
d. IF (time = allotted lecture) AND Student (time = sport) THEN Student (adjust) OR Class Time (adjust).

Our rule-based model will yield a partial schedule as output (as it is unable to assign all the given lectures to rooms and timeslots). Output is divided into: (a) lectures and its associated lecturer/students, assigned to various rooms, (b) list of unassigned lecture classes from constraint conflicts [26,39,41].

3. SIMULATED ANNEALING (SA)

[40] SA is inspired by the annealing technique that aims to strengthen glass or crystals by heating them till it liquefies and is, allowed to slowly cool so that the molecules settles into states with lower energies. With this, the model tracks and alters the state of an individual, constantly evaluating its energy via its energy function. Its optimal point is found by running series of Markov chain under various thermodynamic state [44]. The neighbouring state determined by randomly changing an individual's current state via a neighbourhood function. If a state with lower energy is found, individual moves to it; else, if neighbourhood state has a higher energy, individual moves to that state only, if an acceptance probability condition is met. If not met, individual remains at current state [25, 32-34,44]. [24-25,31,40,44] The acceptance probability is difference in energies between current and neighbouring states, and temperatures. Temperature is initially set high, so individual is more inclined towards higher energy state – allowing individuals to explore a greater portion of the space and preventing it from being trapped in local optima. As model progresses – temperature reduces with cooling and individuals converge towards lowest energy states till an optimum point. [19-20]

Our rule-based SA algorithm is given by [12,25-26,41,44]:

1. Select initial solution via preprocessor rule-based system
2. Select the temperature change counter H = 0
3. Select a temperature cooling schedule,
4. Generate Initial schedule for individual state, energy and temperature S
5. Set initial best schedule S* = S.
6. Compute cost of S : C(S)
7. Compute initial temperature T_o
8. Set temperature T = T_o
9. Loop until temperature is at minimum
10. Loop until maximum number of iterations reached
11. While stop criterion is not satisfied or reached, Do:
12. Repeat Markov chain length (M) times
13. Select random neighbor S' to current schedule, (S' N_s)
14. Find neighbour state via neighbour function
15. If neighbourhood state has lower energy than current
16. Then change current state to neighbouring state
17. Else if the acceptance probability is fulfilled
18. Then move to the neighbouring state
19. Else retain the current state
20. Keep track of state with lowest energy
21. End inner loop: End outer loop

A major advantage of SA over other heuristic methods is in its ability to avoid becoming trapped in local minima. It uses a random search, that not only accepts changes that decrease the objective function f (for a minimization task), but also allows some changes that increases it [14,17-18]. The latter is accepted with a probability defined as thus:

$$P = exp\left(\frac{-\delta f}{T}\right) \quad (1)$$

df is the increase in f, T is a control parameter that by analogy with the original application is known as the system temperature irrespective of the objective function involved [27,30]. SA's implementation is quite straightforward and its structure must have the following elements provided for [27, 30, 37]:

a. a representation of possible solutions
b. a generator of random changes in solutions
c. a means of evaluating the problem functions
d. an annealing schedule - initial temperature and rules for lowering it as the search progresses.

This model employs exploratory search via multiple individuals and flexibility in finding a better optimal point, even when a local minimum are found and present [9,11,40]. The rule-based system initialized, yields candidates with low fitness. If a better individual is not found, best individual is chosen after a number of runs for a series of random walks until an optimal solution is found. SA is run on the chosen "fittest" candidates or individual until a solution is found on the neighbourhood size and function [23-24,40].

To randomly re-initialize the space for a series of Markov chain to be run, the temperature schedule is applied. After which, the neighbourhood function is then applied to randomly change individual energy states and compute best fitness with such individual tracked until a fitness of 0.6 is found. Model finds individuals of low energy to enter SA cycle early enough to apply the temperature schedule as needed [26,28]. Thus, a moderated Markov chain that accepts states with energies of lower or equal to current state's energy, is used. The runs continues till state of 0 energy is reached, to imply that solution is found [31-34,40].

Mapping SA to the TTP task at hand, involves a construct as thus [19-20,26,39,41]:

1. A state is a timetable containing the set:
   I: a set of instructors, L: set of lecture classes
   S: set of students, R: set of rooms
   T: set of timeslot and intervals
2. A cost or energy E(I, C, S, R, T) such that:
   a. E(I) is cost of assigning more than maximum number of allowed class lectures.
   b. E(L) is the cost of assigning certain lecture class within/at same timeslot in violation of the exclusion constraint, for example.
   c. E(S): cost of assigning a student to two or more lecture class that are in time conflict; plus cost of scheduling one/more lectures that do not meet student's major, lectures requested, or lectures requirements; plus the cost of lectures evenly spread over a week.
   d. E(R): cost of assigning rooms of wrong size and/or type to a certain lecture class.
   e. E(I): cost of assigning more/less time than required plus cost of imbalanced lecture assignment (certain periods will have more lectures assigned to them than others) etc.
3. A swap (move) is exchange of one or more of the following: lectures L_i with lecture L_j in a set L with respect to periods T_i and T_j, and/or with respect to classrooms R_i and R_j respectively – and is referred to as lecture class swapping.

Along with these necessary constraints, SA as input data the following: the rule-based system preprocessed output in form of non-scheduled and scheduled such as list of room types, list of lecture, their associated instructors and room types, a department to building distance matrix, list of student number and their lecture preferences, and a list of lectures that are not allowed to be scheduled simultaneously. For effective SA, it is crucial to use a good method for choosing new trials and a good cooling schedule that aids effective search [26,39,41].

3.1. SA with Exponential Cooling

[29,42] Simulated Annealing as a heuristic model has 3-feats, which are of fixed definitions namely: state space, move class and cost function. Its temperature is the only variable during its computation (a feat for all SA variants). Attempts to derive good schedule first, melts the system at a high temperature. Then, repeatedly lowering temperature by the constant ($0<\alpha<1$). Enough steps at each temperature is taken to keep system close to equilibrium, until system approaches ground state. This results in an exponential schedule that is further explained as:

a. Initial Temperature T_0 is estimated by conducting an initial search in which all increases are accepted and calculating the average objective increase observed □f+. This results in an average increase of acceptance probability $P_0 = 0.8$ – implying, there is 80% chance that a change to increases its objective function will be accepted. T_0 clearly depends on scaling f, and is problem-specific defined by [29, 37,42]:

$$T_0 = \frac{-\delta f +}{Ln(P_o)} \quad (2)$$

b. Final Temperature: In some implementations, is determined by scaling: (a) number of temperature values used, and/or (b) total number of solutions to be generated. Alternatively, the model is halted when makes no further progress. Lack of progress is defined in many ways and it generally implies that no improvement or new best solution is being found in the chain at one temperature. Combined with acceptance ratio falling below a given (small) value pf [32].

c. Length of Markov Chains (L_j): The choice of L_j depends on the task's size, which is not same selecting length of the j^{th} Markov chain. Thus, L_j is independent of k. Also, minimum number of transitions N_{min} should be accepted at each temperature. Thus, as Tj approaches 0, transitions are accepted with decreasing probability so that the number of trials required to achieve N_{min} acceptances tends to 1. In practice, each Markov chain is terminated after: (a) L_j transitions, or (b) N_{min} acceptances, whichever comes first, is a suitable compromise [33].

d. Decrementing Temperature: Depends on cooling strategy in use, and we adopts the form:

$$T_{j+1} = \propto T_j \quad (3)$$

□□is constant close to, but smaller than 1. The scheme [34] was first proposed with □= 0.95. It differs from the linear cooling scheme that allows T to be reduced at every L trials via:

$$T_{j+1} = T_j - T \quad (4)$$

Reductions achieved using exponential and linear schemes have been found to be comparable near to each other, and the final value of f, generally improves with slower cooling rates at the expense of greater computational effort. The algorithm performance depends more on cooling rate □T/L than on individual values of □T and L. Obviously, care is taken to avoid negative temperatures when using the linear scheme.

3.2. SA with Reheating as a Cost Function

As adopted [4,26,39,41] the ideas behind is due to [18,34] that stressed specific heating as a measure of the variance of cost (energy) values of states at a given temperature. The higher the variance, longer time it takes to reach equilibrium, so the longer time one spends at temperature or alternatively put, the slower one should lower the temperature.

In such combinatorial optimization such as TTP and traveling salesman, phase transition should be resolved. Related studies show that resolution of the overall structure of the solution occurs at high temperatures; while the fine details of the solution is resolved at low temperatures. Applying reheating depends on the phase transition, allowing the model to spend more time in low temperature phases. Thus, reducing the total amount of time required to solve the task. To find temperature at which phase transition occurs, we must compute specific heat. Phase transition occurs at T($C\,max_H$)when specific heat is maximal, which triggers the change in state ordering – so that if best solution found has a high cost (energy), then this super structure may require re-arrangement by raising temperature higher than the phase transition temperature T($C\,max_H$). Thus, the higher the current best cost, the higher the temperature required to escape local minima. We then compute the maximum specific heat via the steps as adopted in [3,26,28,34,39,41] given by Eq. 9.

[17-18,26] SA yields a set of configuration C(T). If C_i is cost of configuration I, C(T) is average cost at temperature T, and σ(T) is standard deviation of cost T. At each temperature T, the probability distribution for configuration becomes:

$$P_i(T) = \frac{e^{\frac{-C_i}{kT}}}{\sum_j e^{\frac{-C_i}{kT}}} \quad (5)$$

The average cost is computed as:

$$< C(T) > = \sum_{i \in C} C_i P_i(T) \quad (6)$$

Thus, the average square cost is given by:

$$C^2(T) > = \sum_{i \in C} C_i^2 P_i(T) \quad (7)$$

The variance of the cost:

$$\sigma^2(T) = < C^2(T) > - < C(T) >^2 \quad (8)$$

And, the specific heat is given by:

$$C_H(T) = \frac{\sigma^2(T)}{T^2} \quad (9)$$

Temperature T($C\,max_H$)at which max specific heat occurs, or at which model undergoes a phase transition can be found as thus [26,39,41]:

$$T = K * C_b + T(C\,max_H) \quad (10)$$

K is the tunable parameter, C_b is current best cost. Reheating is done when temperature drops below phase transition or max specific heat; and there has been no decrease in the cost of specified number of iterations (point model gets stuck at local minima). Reheating increases temperature above transition phase of Eq. 7 to yield enough configuration change that allows it explore other minima, when the temperature is reduced again [26,39,41].

3.3. SA with Adaptive Cooling

[26,39,41] Computes new temperature from specific heat (standard deviation of costs obtained at the current T) – so that it keeps the model close to equilibrium by cooling slower close to the phase transition, where specific heat is large. Adaptive cooling is implemented in various ways but, we choose the ideas of [26,39,41,45-46], which yields better cooling schedule. T_j is current temperature at j. With $\sigma(T_j)$ in Eq. 5 – new temp T_{j+1} becomes:

$$T_{j+1} = T_j * e^{\frac{-aT_j}{\sigma(T_j)}} \quad (11)$$

a is the tunable parameter. From [14,26,39,45], the function σ(Tj) is smoothed out to avoid dependencies of temperature decrement on large change in standard deviation. Eq. 12 smoothes σ as thus:

$$\gamma(T_{j+1}) = (1-\omega) * \sigma(T_{j+1}) + \omega\sigma(T_j)\frac{T_{j+1}}{T_j} \quad (12)$$

Our smoothing function sets w = 0.95, to follow the Boltzmann distribution that preserves the relation:

$$\frac{d}{dT}C(T) = \frac{\gamma^2(T)}{T^2} = C_H \quad (13)$$

4. METHODS AND MATERIALS

Study aims to schedule only two semesters with different types of room (like Auditoriums, studios, pavilions, classrooms, conference rooms, laboratories, theatres and unspecified such as pavilion, incomplete and make-shifts rooms).

4.1. Model Performance Evaluation

Model's performance is evaluated via coefficients of efficiency (R) and determination (r^2) with ideal values of 1 [21, 23-24,38]:

$$MSE = {}^{1}/_{n} \sum_{i=1}^{m} \{(Ypi - Yio)2\}^{1/2} \quad (14)$$

$$MAE = {}^{1}/_{n} \sum_{i=1}^{m} |Ypi - Yio| \quad (15)$$

$$MRE = {}^{1}/_{n} \sum_{i=1}^{m} \frac{|Ypi - Yio|}{Yio} \quad (16)$$

$$COE(R) = 1 - \frac{(Q_{obs} - Q_{sim})^2}{(Q_{sim} - Q_{obs})^2} \quad (17)$$

$$r^2 = \frac{[(Q_{obs} - (1 - Q_{obs}))(Q_{obs} - (1 - Q_{obs}))]^2}{(Q_{obs} - (1 - Q_{obs}))^2 (Q_{sim} - (1 - Q_{sim}))^2} \quad (18)$$

4.2. Experimental Practice

[14,26,39,45-46] We note that SA's performance in any application is highly dependent on the method used to select the new trial configuration of the model – so that it can effectively sample all parameters in the domain space, achieved via efficient moves. The simplest way for choosing a move is to swap the rooms or timeslots of two randomly selected lectures. This is quite inefficient as random swapping of lectures will increase the overall cost, especially if it had yielded a valid solution (i.e. low temperature). This low acceptance implies that the simple method is very inefficient – since it requires a lot to compute the change in cost.

[26,39,41] A strategy is needed to choose moves that are more likely accepted. If choice of room is made and we randomly choose a new room from list of rooms, it will most likely be rejected (as such a choice may be too small or too large for the class). One possibility is to create a subset of all the rooms that fulfills the hard constraints on the rooms for the particular lecture such as size and room type. After which, we can make random selection of a room for that lecture only, from the subset of feasible rooms with an acceptance probability that is sure to be much higher. [26,39,41] Also, note that each class has space-type-required tag (in dataset) that embodies other information to assign a lecture to a right room, that will effectively separate and update the independent sets based on the room type. Laboratories are scheduled separately from regular lectures. In our method, we carry out the scheduling of lectures first, followed by laboratories – so that we have made sure that no lecture and its associated laboratory is scheduled at same timeslot [26,39,41].

In effect, the embedded expert system in SA will effectively improve the choice of moves as well as use a more complex expert system as a preprocessor for the SA. When used to choose the moves, the main function of a rule-based system is to ensure that all trial moves satisfy the hard constraints. Many of the rules dealing with the medium and soft constraints are softened or eliminated – as reducing cost of the constraints via the Metropolis update in the SA model [26,39,41].

Another modification is that the rule-based model used is completely deterministic (discrete); while the version used in choosing the moves for SA is probabilistic (rather than random) from the multiple possibilities that satisfies the rules equally well. The extra freedom in choosing a new schedule and extra degree of randomness inherent in annealing update helps prevent model from getting trapped in local minimum before it yields a valid schedule, which is a major problem of a completely discrete, deterministic rule-based system [26]. The reason for making the subset of possible moves choices created for each lecture probabilistic is that choosing from it will yield a continuous schedule [35]. There may be certain kinds of moves that are more likely to be effective so that our move strategy is to select these moves with a higher probability. Swapping a higher level lecture (i.e. graduate) with a lower level lecture (i.e. first or second year) has a more general, higher acceptance – since there is little overlap between students taking these lectures [26,41].

Also, we experimented with two forms of swap: (a) lectures offered by same department/college, and (b) lectures between different departments and colleges – to help effectively prune neighbourhood or domain space, yield a more efficient moves and in turn, overall improvement in result [19,26,39,41].

4.3. Rationale, Feats and Tradeoffs in Study

SA has become a model of choice in dealing with CSPs especially in finance with nested regression and local searches applied in hybrid forms. Complexity and nonlinearity in multi-agent, multivariate systems and the increased interest of data in dynamic modeling of some CSPs as rules begs for more evolved, sophisticated model – since parameters in this study, are a mix of continuous and discrete sets. A strong feat of SA is their flexibility in adapting to many ad hoc constraints, rules, algebraic models, amongst others. The power of SA, coupled with new modeling data can be fitted much better than in previously studies. The model appears well suited for the allocation problems with constraints involved [32].

[12] Its merits are: (a) SA deals with highly nonlinear, chaotic, noisy and dynamic constraints, (b) Its flexibility and search for global optimality makes it a better choice over others, (c) it is a more robust and general model, (d) its versatility allows it not to any restrictive model feat, (e) for stochastic and/or very complex nonlinear models, it is easily tuned to enhance its performance in the shortest time. Conversely, its demerits are: (a) SA is a metaheuristic and choices are required to turn it into an actual algorithm, (b) tradeoff exists between quality of the solutions and compute complete time, (c) tailoring work is required to account for different classes of constraints as well as to tune model parameters is quite delicate, (d) precision in the numbers of parameter used in SA has a significant effect on its solutions' quality.

5. RESULT PRESENTATION

Study aims to satisfy all hard constraints, minimize cost of medium and soft constraints via real dataset from University of Benin (Nigeria), find acceptable parameters that yields appropriate move strategy for a general TTP and study the effect of a rule-based system as a preprocessor to provide the SA with a good starting point. Studies indicate that the number of steps for each temperature needs to be proportional to neighbourhood size, to maintain high quality result (though in use with the TTP) as adapted in [26,39,41].

Study resolved all hard constraints, minimizes all soft and medium constraints to find optima, which improved as the number of iterations in the Markov chain becomes proportional to the combination of the number of lectures, rooms and timeslots. TTP is characterized by sparseness. After required number of lectures N_l has been scheduled, sparseness is $N_{sp} = (N_x N_t - N_l)$ sparseness space-timeslots.

Thus, the sparseness ratio is defined as N_{sp}/N_xN_t – and the denser the task, the lower the sparseness ratio, and the harder the task is to be resolved. Also, for denser tasks, there is an additional correlation involving problem domain size. Student's preferences make task more tedious. Though, viewed as medium constraints and are not necessarily satisfied [26,39].

Table 1: Size of Dataset for each Session

Items	First Semester	Second Semester
Instructor	***307***	***307***
Students	***7005***	7005
Rooms	***23***	***23***
Lecture classes		
Buildings	***4***	***4***
Colleges	***2***	***2***
Departments	***11***	***11***

Table 2. Lectures scheduled using the diffferent SA methods

SA Cooling Strategies	First Semester			Second Semester		
	High	Ave.	Low	High	Ave	Low
Reheating	0.72	0.70	0.72	0.80	0.77	0.76
Adaptive	0.67	0.60	0.50	0.67	0.63	0.59
Exponential	0.56	0.49	0.45	0.56	0.54	0.53
ES	0.91	0.89	0.72	0.93	0.91	0.89

Table 3. Scheduled with Rule-Based model as preprocessor

SA Cooling Strategies	First Semester			Second Semester		
	High	Ave.	Low	High	Ave	Low
Reheating	0.99	0.95	0.95	0.99	0.97	0.97
Adaptive	0.93	0.87	0.87	0.92	0.90	0.90
Exponential	0.87	0.79	0.73	0.88	0.75	0.72

Table 4. Model Performance with Rule-Based model

Cooling Strategy	MSE	MRE	MAE	COE	COD
With Reheating	0.87	0.79	0.75	0.781	0.966
With Adaptive	0.76	0.81	0.62	0.753	0.921
With Exponential	0.76	0.77	0.76	0.688	0.812

For randomly initialized data (as against backdrop of template used here), SA performed poorly. But, use of rule-based model as preprocessor yielded an optimal solution that resolved student preferences of about 70% as a medium constraint of very high priority. This is reasonably good as other approaches do not deal with student preferences. Studies are currently aimed at improving this result to cater for student's preferences as a hard constraint.

With these objectives, a comparison of these cooling method is as thus and supported by [26,39]:

a. SA with reheating took 6hours and 45minutes to find the solution after 8 iterations (at best). It was run 25 times (to eradicate bias), and found optima each time. Time varied between 6hours and 14hours, as this depends on how close the initial population is to solution and neighbourhood function, initial and final temperature.
b. SA with Adaptive cooling (at best) 8hours after 12iterations. It found solution for all 25times, and time range between 8hours and 14hours.
c. SA with exponential cooling arrived at solution 2.112seconds after 401 iterations. SA used a Markov chain of 387 iterations to find a solution. On 25 runs, time is between 3seconds and 3minutes – and its convergence time depends on initialization and random swaps from temperature schedule as applied.

6. SUMMARY AND RECOMMENDATION

Models are used for prediction, serve as educational tools to compile existing knowledge about a task, serve as language to communicate hypotheses and gain better insight and to investigate parameters or input crucial to be estimated accurately. Their failure or sensitivity analysis helps us to better reflect theories on natural systems functioning. A detailed model may not be operationally applicable in larger scale, but allow for study and thus, helps to develop other reasonable, applicable model. Simple models may not yield enough data, whereas complex models may not be fully understood. Its application as an intellectual tool requires less accurate numeric agreement between predictions and observations, but rather seeks the model's feedback as more important. Model complexity must be balanced as only models that are understandable and manageable, are fully explored.

SA is quite time consuming for some CSPs; But use of a rule-based system as a preprocessor with a good initial solution drastically reduces time taken to yield a good solution and improved the quality of the result. In theory, a good initial solution should not be necessary, and any state should give a good solution. In practice, there is no ideal cooling schedule and means of choosing trial moves that efficiently explores the space as there are no restrictions on how long forecast takes. Thus, very hard problems with large parameter space, with difficult search efficiency for which very slow cooling will be time consuming, thus requires an initial, good solution and large enough computing power. This result clearly supports our rationale for academic course scheduling [26,39,41].

REFERENCES

[1] Aarts, E.H., Korst, J and Van Laarhoven., (1997). "Simulated annealing" as in Aarts, E.H and Lenstra, J.K., (eds.) Local Search in combinatorial optimization, John Wiley and Sons.
[2] Abdullah, S., Ahmadi, S., Burke, E.K and Dror, M., (2004). Investigating Ahuja-Orlin's large neighbourhood search for examination timetabling, NOTTCS-TR-2004-8 Computer Technical Report, University of Nottingham.
[3] Abramson, D., (1991). "Constructing school timetables using simulated annealing: sequential and parallel algorithms", Management Science, 37(1), pp 98-113.
[4] Abramson, D., Dang, H and Krishnamoorthy, M., (1996). Simulated annealing cooling schedules for timetabling problem, Asian Operation Research, 3(5), pp 11-24.
[5] Al-Tarawneh, H, Y and Masri, A., (2011). Using a Tabu search with multi neighbourhood structures to solve a university course timetable: UKM case study, Faculty of Engineering (Univeriti Kebangsaan Malaysia) conf. on Data Mining and Optimization, pp208-212.
[6] Anh, D.T, Tam, V.H and Hung, N.Q.V., (2006). Generating complete university course timetables by using local search methods, In Proceedings of IEEE International Conference on Research, Innovations and Vision for the Future, ISBN: 1-4244-0316-2, pp 67-74.
[7] Bacchus, F., (2010). "Constraint satisfaction problem", Univ. of Toronto: Lecture notes on Computer Science www.cs.toronto.edu/~ fbacchus/. Accessed Feb. 13, 2015.
[8] Bayram, H and Sahin, R., (2013). A new simulated annealing approach for traveling salesman problem, Mathematical and Computational Applications, 18(3), 313
[9] Brailsford, S.C., Potts, C.N and Smith, B.M., (1998). Constraint satisfaction problem: algorithms and applications, European J. Operation Research, 119, pp 557-581.
[10] Burke, E and Ross, P. (1996). "Practice and theory of automated timetabling", Selected Papers and Lecture notes in Computer Science, 1153, Springer, NY.
[11] Coddington, P., (2012). "Constraint satisfaction problems", Lecture notes on Computer, cs.adelaide.edu.au. Last accessed Feb 13, 2013.
[12] Darrall, H., Jacobson, S.H and Johnson, A.W., (2003). Theory and practice of simulated annealing, Handbook of Metaheuristics, Springer, ISBN: 978-1-4020-7263-5, pp 287-319.
[13] De Warra, D., (1985). "Introduction to timetabling", European Journal of Operation Research, 19, pp 151-162.
[14] Diekmann, R., Luling, R and Simon, J., (1993). "Problem independent distributed simulated annealing and its applications" in Applied Simulated Annealing, Vidal, R.V., Lecture notes in Economics and Mathematics, Springer.
[15] Dos Passos, W., (2013). Numerical methods, algorithms and tools in C#, Chapter 18: Optimization methods, Taylor and Francis Inc., ISBN: 9780849374791.
[16] Gislen, L., Soderberg, B and Peterson, C., (1989). "Teachers and classes with neural networks", International J. of Neural Systems, 1, pp 167 – 178.
[17] Johnson, D., Aragon, C, McGeoch, L and Schevon, C., (1991). "Optimization by simulated annealing: an experimental evaluation Part II (graph partitioning)", Operation Research, 39(3), pp 865-892
[18] Kirkpatrick, S., (1983). "Optimization by simulated annealing", Science, 220, pp 671-680.
[19] Miners, S., Saleh Elmohamed, M.A and Yau, H., (1995). "Optimizing timetabling solutions using graph coloring, NPAC REU program, Syracuse University: NY, www.npac.syr.edu
[20] Ojugo, A.A., (2005). Comparative study of simulated annealing model to solving optimization problem – case of virus propagation on dynamic networks", Unpublished MSc, Nnamdi Azikiwe University Awka, Nigeria.
[21] Ojugo, A.A., Eboka, A.O and Yoro, R.E., (2007). A hybrid simulated annealing neural network model to solving the Sudoku problem, In Proceedings of IRDI 4th Conf. on Science and Technology, 78 – 102, Uyo: Nigeria.
[22] Ojugo, A.A., Eboka, A.O., Okonta, E.O., Yoro, R.E and Aghware, F., (2012). "Genetic algorithm rule-based intrusion detection system", J. Emerging Trends in Computing and Information Systems, 3(8), 1182 - 1194.

[23] Ojugo, A.A, Emudianughe, J.E, Yoro, R.E, Okonta, E.O and Eboka, A.O, (2013a) "Hybrid ANNGSA for runoff modeling, Progress in Intelligence Computing Applications, 2(1), doi: 10.4156/pica.vol2.issue1.2, pp22 – 33.
[24] Ojugo, A.A., and Yoro, R.E., (2013b). "Computational intelligence in stochastic solution for Queen problem", Progress in Intelligence Computing and Applications, 2(1), doi: 10.4156/pica.vol2.issue1.4, pp 46 – 56.
[25] Perez, M and Marwala, T., (2011). Stochastic optimization approaches for solving Sudoku, Proceeding of IEEE Congress on Evolutionary Computing, pp 256 – 279.
[26] Saleh Elmohamed, M.A, Fox. G and Coddington, P., (1998). "A comparison of annealing techniques for academic course scheduling", Notes on Intelligence Computing, DHCP-045, pp 1-20. www.dhpc.adelaide.edu.au.
[27] Schaerf, A., (2003) "Survey of timetable problem", Dept. of Software Technology, Report CS-R9567, CWI, Amsterdam: Netherlands.
[28] Sontag, E.D., (1998). "Learning for continuous-time recurrent neural networks", Systems and Control Letters, 34, pp. 151-158.
[29] Sorkin, G., (1991). "Theory and practices of SA on special landscape", PhD thesis, Dept of Electrical Engineering Computer Science, University of California, Berkeley
[30] Tarkesh, H., Atighehchian, A and Nookabadi, A.S., (2009). Facility layout design using virtual multi-agent system, Journal of Intelligent Manufacturing, 20(4), Springer, pp 347 – 357.
[31] Thomson, J and Dowsland, K., (1995). "General cooling schedules for simulated annealing based timetable problems, Proceeding of Practice and theory of automated timetabling, Edinburg: Napier University, pp421–444
[32] Van Laarhoven, P.J and Aarts, E.H., (1987). "Simulated annealing: theory and applications", D. Reidel, Dordrecht.
[33] Vidal, R.V. ed., (1993). "Applied simulated annealing", Lecture notes in Economics and Mathematical systems, 396, Springler-Verlag.
[34] White, S.R., (1984). "Concepts of scale in simulated annealing", Proceeding of IEEE Int. conference on Circuit Design, pp 646-651.
[35] Zhang, L and Lau, S.K, (2005). Constructing university timetable using constraint satisfaction programming approach, IEEE Intelligent Agents, Web technology and Internet Commerce, 2, pp 55-60
[36] Bashir, H.A and Neville, R.S., (2013). Hybrid evolutionary computation for continuous optimization, arxiv: 1303.3469, http://arxiv.org>cs
[37] Busetti, F., (1995). Simulated annealing overview,168.18.62.64/wisdom/simulated%20annealing%20overview.pdf .
[38] Guo, Z., Zhao, J., Zhang, W and Wang, J., (2011). A corrected hybrid approach for wind speed prediction in hexi corridor of China, Energy, 36(3), pp 1668 – 1679.
[39] Kumral, M and Dowd, P.A., (2005). A simulated annealing approach to mine production scheduling, J. of Operation Research Society, 56(8), 922-930, www.dhpc.adelaide.edu.au
[40] Lassig, J and Sudholt, D., (2011). Adaptive population model for offspring population and parallel evolutionary algorithms, arxiv: 1102.0588, http://arxiv.org>cs
[41] Michalewicz, Z., (1998). A survey of constraint handling techniques in Evolutionary computation methods, www.dhpc.adelaide.edu.au
[42] Nourani, Y and Andersen, B., (1998). A comparison of simulated annealing cooling strategies, Journal of Physics A: Mathematical and General, 30, pp 8373 – 8385.
[43] Ozcan, E., (2005). Towards an XML-based standard for timetable problem: TTML, Multidisciplinary Scheduling: Theory and Applications, 24, pp163
[44] Perez, M and Marwala, T., (2012). Microarray data feature selection using hybrid genetic algorithm simulated annealing, IEEE conference on Electrical and Electronics Engineers, doi: 10.1109/EEEI.2010.6377146, pp 1 – 5.
[45] Nikolaev, A.G., Jacobson, S.H., Hall, S.N and Henderson, D., (2011). A framework for analyzing sub-optimal performance of local search algorithms, Computational Optimization and Applications, 49(3), pp407.
[46] Iovleff, S and Perrin, O., (2004). Estimating a non-stationary spatial structure using simulated annealing, Computational and Graphical Statistics, 13(1), doi: 10.1198/1061860043100.

SEMESTER LECTURE TIME TABLE, 2014/2015 ACADEMIC SESSION
100 LEVEL

DAY/TIME	8-8:50AM	9-9:50AM	10-10:50AM	11-11:50AM	12-12:50PM	1-1:50PM	2-2:50PM	3-3:50PM	4-4:50PM
MONDAY	PHY 121 (LT1)		MAT 122 (LT1) / EVS 121 (LR5)		BIO 121 (LT1) PHY 123 (LR5)		PHY 129 (PLB2)		
TUESDAY	MAT 121 (LT1)		BIO 129 (ELB) PHY123[LT1]		BIO 129(ELB) MAT123[LT1]	CHM 121 (LT1)		GSE 121 (LT1)	
WEDNESDAY		BIO 121 (LT1) PHY 122 (LR5)		CSC 121 (LT1)	MAT123[LT1]		SPORT		
THURSDAY	CHM121 (LT1)	PHY121[LT1]		CHM 109 (CLB2)			CSC 121 (LT1)		
FRIDAY	MAT 121 (LT1)	CHM 122 (LT1)	PHY 122 (LT1)	PHY 122 (LT1)	GLY121 (LR5)		MAT 122 (LT1)	EVS 121 (LR5)	

200 LEVEL

DAY/TIME	8-8:50AM	9-9:50AM	10-10:50AM	11-11:50AM	12-12:50PM	1-1:50PM	2-2:50PM	3-3:50PM	4-4:50PM
MON	MAT 225 (LR1) EEE 221[LT2] CHM225[LR4]		PHY221(LR2)	MAT 224 (LR1) PHY 221 (LR2) CSC223 (LT2)	MAT 224 (LR1) CSC223(LT2) CHM 222 (LR3)	**EVS 222 (LR1)** GLY 221 (LR2) GET 221 [LT2] CHM 222 (LR3)	EVS 222 (LR1) MAT 226(LR2) GET 221[LT2]	CSC 221 (LR1) CVE 221[LT2] CHM226[CLB2]	
TUES	MAT222 (LR1) CHM221 (LR2) PHY 223 (LR3) GET 222[LT2]		EVS 221 (LR2) GLY 221 (LR3) MEE 224 [LT2] PHY 222 (LR1)		MAT 221 (LR1) EVS 224 (LR2) GLY 222 (LR3) MEE 222[LT2]		CHM 229 (CLB2)	CHM 229 (CLB2) MEE 221[LT2]	
WEDN	MAT221(LR1) PHY 222(LR1)	MAT 222(LR1) PHY 223 (LR2) GLY 224 (LR3) GET 222 [LT2]	CHM 223 (LR1) EVS 223 (LR3) MAT 226 (LR1)		MAT 225 (LR1) GLY 222 (LR2) CHM224[LR3]	GSE 221 (LR1)	GSE 221 (LR1)	SPORTS	
THURSDAY	EVS 225 (LR1) GLY 223(LR3)	MAT 224(LR1) EVS 225 (LR3)	MAT 223 (LR1) GLY 224 (LR2) MEE 223[LT2] EVS 229 (EVL)		AGP 221 (LR1) EVS 229 (EVL) CHM224[LR3]	AGP 221 (LR1) CHM224[LR3]	AGP222 (LR3) PHY229 (LB) CSC222 (LR2)	AGP222(LR3) PHY229 (LB) CSC222(LR2)	PHY 229(LB)
FRIDAY	CSC 221 (LR2)	CSC 221(LR2) GLY223 (LR1)	GET229[LH1] GLY223(LR1)	GET229[LH1]	GET229[LH1]				

300 LEVEL

DAY/TIME	8-8:50AM	9-9:50AM	10-10:50AM	11-11:50AM	12-12:50PM	1-1:50PM	2-2:50PM	3-3:50PM	4-4:50PM
MONDAY	EEE321[LT2] CHE320[LR4] MEE323[LR3] PNG322[LH]			CHE321[LH] / MEE321[LR4] PNG320[LR3] / GLY322[LR2]		EEE322[LR3] PNG321[LH]	MAR321[LR3]	GET322[LH] MAR321[LR3]	
TUESDAY	CHE322[LR3] / PNG321[LH]		EEE320[LH] CHE320[LR3] MEE323[LR4] GLY326[LR2] PNG322[LH]	EEE320[LH] MEE323[LR4] CHE320[LR3]	AGP222/PNG325[LH] GLY322[LR3] MEE327[LT2]	EEE323[LR3] AGP222/PNG325[LH] MEE327[LT2]	EEE323[LR3] MAR321[LH]	CHE323[LR4] MAR321[LH]	
WEDNESDAY	GET321[LH]	EEE321[LH] CHE323[LR3]	GET323[LH] / GLY326[LR3]		CHE322[LR4] MEE322[LH]	EEE322[LR1] PNG320[LR3] MEE322[LH]	CHE321[LR4]	SPORTS	
THURSDAY	GET 321[LH]		EEE320[LH]	CHE320[LH] MEE324[LR4]	EEE325[LR3] CHE302[LH] MEE324[LR4] CHE325[LT2]	EEE325[LR3] MEE326[LH] CHE325[LT2]	MEE326[LH]	EEE324[LH]	
FRIDAY	MEE329[LR4] EEE329[LH] MAR329[LB]	MEE329[LR4] EEE329[LH] MAR329[LB]	MEE329[LR4] EEE329[LH] MAR329[LB]				CHE324[LH] PNG324[LR4]	CHE324[LH] PNG324[LR4]	CHE324[LH] PNG324[LR4]

400 LEVEL

DAY/TIME	8-8:50AM	9-9:50AM	10-10:50AM	11-11:50AM	12-12:50PM	1-1:50PM	2-2:50PM	3-3:50 PM	4-4:50PM
MONDAY	EVS 421 [EVL] MAT 421 [CSL] CHM421[LR5] GLY 421 [GLL] PHY 421 (PLB)		CHM 424 [LR5] CSC 425 [CSL] AGP 423 [GLL]	CSC 421 [CSL] EVS 422 [EVL] CHM 422 [LR4] GLY 422 [GLL] PHY 422 (PLB) MAT425[LR5]		CSC 424 [CSL] CHM 425 [LR5] GLY 426 [GLL] PHY 429 (LR5)	CSC 423 [CSL] CHM 425 [LR5] GLY 426 [GLL] PHY 429 (LR5)	MAT 423 [LR4] GLY 423 [GLL]	MAT423[LR4] CSC424[CSL]
TUESDAY	CSC 421 [CSL] CHM 422 [LR4] EVS 423 [EVL] GLY 425 [GLL] MAT425[LR5]	EVS 421 [EVL] PHY 421 [LR4] MAT 421 [CSL] CHM421 [LR5] GLY 425 [GLL]	CSC 422 [LR5] PHY 423 [PHL] GLY 422 [GLL] EVS 427 [ELB] MAT 424 [LR4]	MAT 424 [LR4] EVS 423 [EVL] PHY 423 [PHL] GLY 428 [GLL] EVS 427 [ELB]	EVS 423 [EVL] GLY 428 [GLL] PHY 421 (LR4) CSC428[CSL]	CSC 424 [CSL] CHM426[CLB2] AGP 422 [GLL] PHY 424 [PHL]	CSC 424 [CSL] CHM426[CLB2] AGP 422 [GLL] PHY 424 [PHL]	CSC 426 [LR5] CHM 427 [CLB2] GLY 423 [GLL]	
WEDNESDAY	MAT 422 [LR4] EVS 424 [EVL] GLY 427 [GLL]	MAT 422 [LR4] CHM423[CLB2] EVS 424 [EVL]	CHM 423 [CLB2] MAT 421 [LR4] PHY 423 (LB)	MAT 424 [LR4] CHM 424 [LR5] GLY 424 [GLL]	MAT 424 [LR4] CHM 424 [LR5] GLY 424 [GLL]	CSC 426 [LR4] AGP 424 [GLL] PHY 425 (PHL)	AGP 424 [GLL] PHY 425 (PHL) EVS424[EVL]	CSC 427 [CSL] CHM 429 [CLB2] GLY 426 [GLL]	
THURSDAY	MAT 422 [LR5] AGP 426 [GLL]	MAT 422 [LR5] EVS 425 [EVL] GLY427[GLL]	EVS 425 [EVL] AGP 426 [GLL] PHY 428 (PHL) CHM430[CLB]	EVS 428 [LR5] AGP 423 [GLL] PHY 428 (PHL) CHM430[CLB]	CSC 425 [CSL] EVS 428 [EVL] AGP 423 [GLL] PHY 426 (PHL	CSC 423 [CSL] AGP 421 [GLL] PHY 426 (PHL) CSC 425 [LR4]	AGP 425 [GLL]	CSC 422 [LR5] GLY 429 [GLL]	
FRIDAY	MAT 423 [LR4] PHY 427 [PHL] AGP 421 [GLL]	MAT 423 [LR4] PHY 427 [PHL] AGP 421 [GLL]	CSC 427 [CSL] AGP 426[GLL]	CHM 428[CLB2] GLY 427 [GLL] CSC428[CSL]	CHM 428[CLB2] GLY 427 [GLL] CSC428[CSL]			AGP 421 [GLL]	

500 LEVEL

DAY/TIME	8-8:50AM	9-9:50AM	10-10:50AM	11-11:50AM	12-12:50PM	1-1:50PM	2-2:50PM	3-3:50PM	4-4:50PM
MONDAY	CHE520[LR1] MEE521[LR6] EEE524[LR3] MAR534[LR4] EEE546[LR6] PNG520[LR2]		EEE525[LR6] MEE526[LR5]	MEE537[LR4] EEE523[LR3] MAR535[LR1] EEE533[LR5] EEE544[LR6] PNG521[LR2]		EEE550[LT2] MAR521[LR1] PNG521[LR2]	EEE550[LT2] MAR521[LR1]	CHE524[LR1] MEE527[LR3] MAR527[LR2] EEE543[LR5] EEE532[LR4] PNG523[LR3]	MEE525[LR4]
TUESDAY	GET521[LT2]	GET521[LT2]	CHE521[LR2] MAR522[LR1] EEE532[LR6] EEE543[LR5] PNG523[LR3]	CHE521[LR2] MAR522[LR1] EEE532[LR6] EEE543[LR5] PNG523[LR3]	MAR523[LR1] EEE535[LR4] PNG521[LR2]	MAR523[LR1] EEE535[LR4]	CHE525[LR2] MEE521[LR6] MAR533[LR4] PNG520[LR2]	PNG525[LR6] MEE536[FLB]	MEE536[FLB]
WEDNESDAY	EEE521[LR6] MAR531[LR1] EEE531[LR5] EEE541[LR4] PNG521[LR2]	EEE521[LR6] MAR531[LR1] EEE531[LR5] EEE541[LR4] PNG521[LR2]	CHE525[LR2] MEE527[LR3] MAR526[LR1]	CHE525[LR2] MEE527[LR3] MAR526[LR1]	PNG526[LR4] MEE526[LR2]	CHE520[LR1] EEE525[LR3] MEE526[LR2]	CHE522[LR1] MAR532[LR2] EEE528[LR4] EEE526[LR3] PNG525[LR6] MEE524[FLB]	MEE524[LR1] MAR532[LR2] EEE528[LR4] EEE526[LR3] PNG525[LR6] MEE536[LH]	GET521[LT2]
THURSDAY	CHE524[LR3] MEE523[LR1] MAR526[LR5] PNG526[LR4]	CHE524[LR3] MEE523[LR1] PNG526[LR4]	MEE525[LR4]	MAR531[LR1] MEE525[LR4]	EEE521[LR6] MAR533[LR4] EEE531[LR5] EEE541[LR4]	MEE537[LR4] MAR527[LR2}	CHE502[LR1] MEE522[LR6] MAR527[LR2}	MAR525[LR1] MEE524[LR2]	MAR525[LR1]
FRIDAY									