An Immune-Based Approach to University Course Timetabling:
Immune Network Algorithm

Muhammad Rozi Malim
Faculty Info. Technology & Quantitative Science, UiTM
40450 Shah Alam, Malaysia
rozi@tmsk.uitm.edu.my
(Tel: 019-2124374)

Ahmad Tajudin Khader
School of Computer Science
Univ. of Science Malaysia
11800 Penang, Malaysia
tajudin@cs.usm.my
(Tel: 04-6533646)

Adli Mustafa
School of Math. Science
Univ. of Science Malaysia
11800 Penang, Malaysia
adli@cs.usm.my
(Tel: 04-6533968)

Abstract - The university course timetabling is known to be a highly constrained optimization problem. The main difficulty is to obtain a conflict-free schedule within a limited number of timeslots and rooms. Many different approaches, including evolutionary algorithms, tabu search, simulated annealing, and their hybrids are developed for solving many different types of course timetabling problems. The immune network algorithm, an algorithm inspired by the immune system, has successfully been applied to fault recognition, data analysis, and optimization. This paper presents an artificial immune network algorithm for the university course timetabling with the main objective to show that the algorithm may be tailored for educational timetabling. The experimental results, using three benchmark course datasets, have significantly shown the effectiveness of the algorithm by producing good quality course timetables. For future work, other artificial immune algorithms, such as negative selection, will be applied to university course timetabling using the same course datasets.

Keywords - Artificial Intelligence; Course Timetabling; Artificial immune system; Immune Network algorithm.

1. INTRODUCTION

University course timetabling problem is known to be a highly constrained combinatorial optimization problem (NP-hard). The problem is periodically faced by virtually every college and university in the world. Usually it involves taking the previous semester’s timetable and modifying it so it will work for the new semester. The course timetabling problem (CTP) can be viewed as a multi-dimensional assignment problem [1]. In a timetable, a number of courses are assigned into classrooms and a limited number of timeslots (periods of time) within a week. Given a set of courses, a set of lecturers, a set of timeslots, a set of rooms, and a set of student enrollments to courses, the problem is to assign lecturers to courses, courses to timeslots, and courses to rooms satisfying a set of hard and soft constraints. The main difference from examination timetabling that makes the CTP more complex is that in course timetabling there cannot be more than one event per room, but in examination timetabling there can be more than one exam.

Conflicting objectives and the changing set of constraints in different institutions makes the course timetabling problem very challenging. Many different approaches, including evolutionary algorithms (EA), tabu search (TS), simulated annealing (SA), and their hybrids are developed for solving many different types of course timetabling problems.

Artificial immune system (AIS), a new branch of Artificial Intelligence [2], is a new intelligent problem-solving technique that being used in optimization and scheduling problems [10]. AISs have been more successful than genetic algorithms (GA) and other methods in pattern recognition, computer and network security, and dynamic tasks scheduling due to the applicability features of natural immune systems. Furthermore, the solutions produced by the AIS are observed to be robust than solutions produced by a GA [11].

This paper presents an artificial immune algorithm called immune network algorithm for university course timetabling (INACT). The main objective is to show that the algorithm (INA) may be tailored for solving course/lecture timetabling problems. Another objective is to show that INACT is an effective optimization algorithm; capable of producing good quality course timetables. Three benchmark course timetabling datasets have been used to implement and test the algorithm. The experimental results using the datasets have significantly shown that INACT is an effective optimization algorithm; has successfully produced good quality course timetables with ‘low’ fitness values for most trials and datasets.

2. COURSE TIMETABLING PROBLEM

Course timetabling problem is a specific case of the more general timetabling problem. At its simplest, course timetabling is the problem of scheduling a set of events (lectures, tutorials or labs) to a set of classrooms in a set of timeslots, and taught by a set of teachers, such that no student or teacher is expected to be in more than one room at the same time and that there is enough space available in each classroom for the number of students expected to be there. These two main (hard and soft) constraints, and many others, combine to make course timetabling a classically hard problem to solve.

Hard constraints must be satisfied in order to produce a feasible timetable, whilst violation of soft constraints should be minimized. The main hard constraints in course timetabling are usually represented by the following:

i) A teacher or student can only attend one event at a time.
ii) A teacher must not be assigned to events in the timeslots during which he/she is unavailable.
iii) The number of timeslots that assigned to each event must equal to the event’s weekly frequency.
iv) Two events of the same course must not be scheduled at the same timeslot.
v) The number of events occurring simultaneously in a certain timeslot must not exceed the number of available rooms in that timeslot.

vi) A room can only host one event at a time.

Individual institutions may have their own specialized hard constraints based on their needs and requirements. Any timetable which fails to satisfy these constraints is deemed to be infeasible.

Soft constraints are generally more numerous and varied and are far more dependent on the needs of the individual problem than the more obvious hard constraints. It is the soft constraints which effectively define how good a given feasible solution is so that different solutions can be compared and improved via an objective (fitness) function. The common soft constraints in course timetabling are:

i) Each teacher should be assigned to their preferred courses.

ii) Events of the same student group should be assigned in consecutive timeslots.

iii) All tutorials or lab sessions of any course should occur later in the week than the week’s first lecture on that course.

iv) An event should (or should not) take place in a certain timeslot.

v) There should be sufficient seats in each room to house all the students present.

vi) Events are assigned to rooms in such a way that the room utilization can be maximized, or spare seats in each room are minimized.

The course timetabling problem can be seen as consisting of three subproblems; ‘course-teacher assignment’, ‘event-timeslot assignment’, and ‘event-room assignment’ [13]. In ‘course-teacher assignment’, the teachers (lecturers and tutors) are scheduled to a number of events in all the courses; in ‘event-timeslot assignment’, all events for all the courses in all student academic programs are scheduled into a fixed number of timeslots; and in ‘event-room assignment’, these events are assigned to a fixed number of rooms. Hence, in a course timetabling problem, an assignment is an ordered 4-tuple \((a, b, c, d)\), where \(a \in E, b \in T, c \in R\), and \(d \in P\). An assignment has the straightforward general interpretation: ‘event \(a\) starts at timeslot \(b\) in room \(c\), and is taught by teacher \(d\)’. For some institutions, the allocation of courses to teachers is carried out manually, and the allocation events in a given timeslot to rooms is a secondary problem (the number of rooms may be large) and can be done later as a separate activity. For real-life situations, these three subproblems can be solved separately.

Malim et al. [13] presented a general model for university course timetabling. The model may be used to formulate various kinds of course timetabling problems as 0-1 integer programming. The paper has considered all possible hard constraints and soft constraints for course timetabling. This general model will be used to formulate the three benchmark course timetabling problems before applying the INACT.

3. Artificial Immune System and Immune Network Algorithm

The ‘artificial immune system’ is an approach which uses the natural immune system as a metaphor for solving computational problems, not modeling the immune system [14]. The main application domains of AIS are anomaly detection, pattern recognition, computer and network security, fault tolerance, dynamic environments, robotics, data mining, optimization, and scheduling.

The ‘immune system’ (IS) can be considered to be a remarkably efficient and powerful information processing system which operates in a highly parallel and distributed manner [9]. It contains a number of features which potentially can be adapted in computer systems; recognition, feature extraction, diversity, learning, memory, distributed detection, self-regulation, threshold mechanism, co-stimulation, dynamic protection, and probabilistic detection. From the perspective of information processing, it is unnecessary to replicate all of these aspects of the IS in a computer model, rather they should be used as general guidelines in designing a system.

There are a number of different algorithms that can be applied to many domains, from data analysis to autonomous navigation [3]. These immune algorithms were inspired by works on theoretical immunology and several processes that occur within the IS. The AISs lead to the development of different techniques, each one mapping a different mechanism of the system. For examples, the Artificial Immune Networks as proposed by Farmer et al. [7], the Clonal Selection Algorithm proposed by de Castro and Von Zuben [5], and the Negative Selection Algorithm introduced by Forrest et al. [8]. The immune network models are suitable to deal with dynamic environments and optimization, algorithms based upon the clonal selection principle are adequate to solve optimization and scheduling problems, and the negative selection strategies are successfully applied to anomaly detection.

CLONALG [5] is the most abstraction of clonal selection algorithm. de Castro and Von Zuben [6] presented an AIS combining CLONALG with the immune network theory introduced in Jerne [12]. This model named aiNet has been successfully applied to several data compressions and clustering applications. The same rationales that led to the development of CLONALG were motivations for the implementation of an optimization version of aiNet [4]. aiNet is an extension of CLONALG with steps involving the interaction of the network cells with each other. de Castro and Timmis [4] presented an immune network algorithm for multimodal function optimization using the optimization version of aiNet.

3.1 Immune Network Theory

The immune network algorithm (INA) is based on Jerne’s idiotypic network theory [12]. According to this theory, immune cells have portions of their receptor molecules that can be recognized by other immune cells in a way similar to the recognition of an invading antigen. This results in a network of recognition between immune cells. When an immune
cell recognizes an antigen or another immune cell, it is stimulated. On the other hand, when an immune cell is recognized by another immune cell, it is suppressed. The sum of the stimulation and suppression received by the network cells, plus the stimulation by the recognition of an antigen corresponds to the stimulation level $S$ of a cell.

3.2 Immune Network Algorithm for Course Timetabling (INACT)

Figure 1 shows the INA developed for the course timetabling problems. This algorithm, called INACT (Immune Network Algorithm for Course Timetabling), was developed based on the general immune network algorithm proposed by de Castro [3] and the optimization version of aiNet developed by de Castro and Timmis [4].

4. BENCHMARK DATASETS

The three course timetabling datasets used in the implementation are available on Internet from http://www.diegm.uniud.it/schaerf/projects/coursett.

These datasets are the real-world course timetabling instances from the School of Engineering at the University of Udine, called Schaeerf datasets. These datasets provide a reasonable benchmark problems for comparison with other approaches or algorithms. The datasets (and characteristics) are shown in Table 1.

<table>
<thead>
<tr>
<th>Instance</th>
<th>No. of Courses</th>
<th>No. of Rooms</th>
<th>No. of Timeslots</th>
<th>Total Lectures</th>
<th>No. of Teachers</th>
<th>Occupancy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>12</td>
<td>20</td>
<td>207</td>
<td>39</td>
<td>86.25%</td>
</tr>
<tr>
<td>2</td>
<td>52</td>
<td>12</td>
<td>20</td>
<td>223</td>
<td>49</td>
<td>92.92%</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>13</td>
<td>20</td>
<td>252</td>
<td>51</td>
<td>96.92%</td>
</tr>
</tbody>
</table>

Each of the datasets comes in five files; course.dat contains the information about the courses (course, teacher, number of lectures, minimum number of days, and number of students), periods.dat contains the list of timeslots of the timetabling horizon (day, start time, and end time), curricula.dat contains the information about groups of courses that share common students (group, size, and members), constraint.dat contains additional constraints about timeslot unavailabilities (course, and unavailable timeslots), and room.dat contains information about rooms (room, and size). The ‘occupancy’ indicates the percentage of timeslot-room required to schedule all the lectures.

5. IMPLEMENTATION

First of all, before implementing the algorithm, the course timetabling problem may be formulated as 0-1 integer programming model. The model would assist in encoding the algorithm into C++ programming codes.

5.1 Mathematical Model

Using the general model [13], the formulation may be carried out as follows:

- **Five hard constraints are considered for all datasets:**
  i) All lectures of all courses must be scheduled.
  ii) Two distinct lectures cannot take place in the same room in the same timeslot.
  iii) Lectures of courses of the same group (curriculum) must be all scheduled at different timeslots.
  iv) Lectures of courses taught by the same professor must be scheduled at different timeslots.
  v) Lectures of some courses must not be assigned to certain timeslots as given in course-timeslot restriction matrix.

- **Three soft constraint are considered for all datasets.**
  These constraints will be used to evaluate the fitness value of each feasible timetable:
  vi) The number of students that attend a course must be less than or equal to the number of seats of all rooms that host its lectures.
vii) The lectures of each course must be spread into not less than a specified minimum number of days. Each course that assigned to less than the minimum number of days will be incurred a penalty of 5.

viii) The daily schedule of lectures of the same group (curriculum) should be as much compact as possible, avoiding gaps between lectures. Each gap will be incurred a penalty of 2.

- Since there are five sets of variables (course, teacher, group, timeslot, and room), only the matrices course-teacher preassignment, course-group allocation, course-timeslot restriction, group-conflict, teacher-conflict, and course-timeslot-room assignment need to be constructed; the first five are input matrices and the other is the output matrix (timetable). The course-teacher assignment has been carried earlier and given in course-teacher preassignment matrix.

- The 0-1 integer programming model (course-timeslot-room assignment) for each of the datasets may be formulated as follows:

\[
\text{minimize } \sum_{i=1}^{n_c} x(c_i, t_j) + 5 \times \sum_{i=1}^{n_c} x(c_i, d_i) + \\
2 \times \left( \sum_{i=1}^{n_c} \sum_{j=1}^{n_t} \sum_{k=1}^{n_r} \frac{b_{ik}}{s_i} \frac{b_{jk}}{s_j} x(c_i, t_j, r_k) \right) + \\
\text{(total violations of hard constraints)} \times 1000; \\
\text{subject to } \\
\sum_{i=1}^{n_c} x(c_i, t_j) = 0 \quad (2) \\
\sum_{i=1}^{n_c} \sum_{j=1}^{n_t} x(c_i, t_j, t_k) = 0 \quad (3) \\
\sum_{i=1}^{n_c} x_c(c_i, c_j) = 0 \quad (4) \\
\sum_{i=1}^{n_c} x_p(c_i, c_j) = 0 \quad (5) \\
\sum_{i=1}^{n_c} \sum_{j=1}^{n_t} c_j t_j = 0 \quad (6) \\
\text{all variables are integers 0-1;}
\]

where \( x(c_i, t_j) = \sum_{i=1}^{n_t} t_{ij} \cdot s_i \cdot n_i(t_i) \) if \( \sum_{i=1}^{n_t} t_{ij} \cdot s_i > n_i(t_i) \) (0 otherwise); \( x(c_i, d_i) = 1 \) if the number of days assigned for course \( c_i \) is \( d_i \) (0 otherwise);

\( x(c_i, c_j, t(c_i)) = 1, \ |t(c_i) - t(c_j)| = 2, \ t(c_i) \) indicates the assigned timeslots for events \( c_i \), \( c_j \) and \( t(c) \) belong to the same group; \( x(c_i, t_j) = 1 \) if \( \sum_{j=1}^{n_t} t_{ij} < l_j \);

\( x_c(c_i, c_j) = 1 \) if \( \sum_{i=1}^{n_i} \sum_{j=1}^{n_t} t_{ik} \cdot t_{jk} \cdot d_{ij} > 0 \); and

\( x_p(c_i, c_j) = 1 \) if \( \sum_{i=1}^{n_i} \sum_{j=1}^{n_t} t_{ik} \cdot t_{jk} \cdot e_{ij} = 1 \).

5.2 Implementation of the INACT

For each dataset, the Immune Network Algorithm (INACT), presented in Figure 1, has been implemented as follows:

**Initial timetables:**

Ten (10) initial feasible timetables are generated using a simple random selection algorithm, i.e., each course is selected at random and all lectures of the course are assigned to random selected timeslots and rooms, satisfying all hard constraints. However, the third hard constraint (group-clashing) is relaxed (considered as a soft constraint) since it is almost impossible to generate 10 initial feasible timetables that satisfy all five hard constraints. The third hard constraint will be satisfied during the optimization process (cloning and mutation). For each group, if there are lectures that scheduled at the same timeslot, a penalty of 1000 will be incurred. For the third instance (with highest occupancy rate), the second hard constraint (room-clashing) is also relaxed in order to produce the initial feasible timetables and will be satisfied during the optimization process.

**Population loop:**

Network interactions and stimulation: The fitness value of each timetable in the current population is evaluated via a fitness function. This function represents the total violations of the soft constraints. Since three soft constraints are considered, the total violations is equal to the number of students without a seat, plus the number of courses that assigned to less than the minimum number of days multiply by 5, plus the number of gaps between lectures of the same group on the same day multiply by 2, and plus the total violations of the hard constraints multiply by 1000. Then the stimulation level of each timetable is evaluated by taking the inverse of the fitness value. The stimulation probability is directly proportional to stimulation level; equal to stimulation level divide by the total stimulations of the population.

Metadynamics (Antigens and Genetic variations): Each of the timetables is selected for cloning and mutation. Good timetables (high stimulation/low fitness) will have more clones than bad ones. In the cloning process, the number of clones is equal to population size multiply by stimulation probability. Each cloned timetable will be mutated to produce a new timetable. For a population size 10, on average, the number of clones generated in each generation is equal to 10. But not all clones will form a new population for the next generation, depending on the stimulation level (fitness value) and the mutation rate.

The mutation operator, based on a mutation rate, works by taking one course at random and reallocate all lectures of the course to randomly selected timeslots and best rooms, always maintaining a feasible timetable. The mutation rate is equal to one minus the selection probability of the current cloned timetable. A random probability is generated, then the mutation is performed if the random probability is less than mutation rate. Since the selection probability is small, on average 0.1 for population size 10, this gives a high mutation rate for each immune cell (timetable), called hypermutation. For a population size 10, on average, the mutation rate is 0.9, and hence 90% of the cloned timetables will be mutated. For each mutated timetable, if all hard constraints are satisfied and there are no duplicate timetables in the current population, the mutation is a ‘success’ and the fitness value is then determined. If the fitness value of the current mutated clone is greater than the fitness value of the original clone, the mutation is a ‘success’, and the reassignment is then reset. If the mutation is a ‘failure’, the same mutation process must be repeated until the process is a ‘success’. If no mutation is
performed, the timetable will be assigned a large fitness value (‘0’ stimulation level) so that it will be eliminated during the population update.

Network dynamics (immune cells and antigens interactions, and population update): Finally, gather all current and cloned timetables to form a population of all feasible timetables in the current generation. Sort the timetables according to stimulation level (fitness value) in descending (ascending) order. Select the best 10 timetables (high stimulation/low fitness), and update the population with these selected feasible timetables. A new population of feasible timetables for the next generation is now produced, with one or more (or none) new feasible timetables. Hence, the timetables with low stimulation (high fitness) will be eliminated during the process.

**Cycle:**
The process (population loop) will be repeated until the maximum number of generations (1000) is reached.

6.EXPERIMENTAL RESULTS

The INACT has been implemented on the three datasets (Schaerf datasets). The following (Table 2) are the first experimental results on solving course timetabling problems using immune network algorithm. The algorithm was run on each dataset for five trials, and the maximum number of generations 1000 was used. The fitness values, equation (1), for all trials and the best fitness values (based on the five trials) for all datasets have been recorded.

| Instance | Total Fitness Value [SC1|SC2|SC3] |
|----------|-----------------------------------|
| Trial 1  | Trial 2  | Trial 3  | Trial 4  | Trial 5  | Best    |
| 1        | 265     | 279     | 290     | 332     | 316     | 265     |
| 2        | 11      | 16      | 28      | 23      | 26      | 11      |
| 3        | 53      | 73      | 55      | 50      | 131     | 50      |

In all trials on all datasets, all the five hard constraints were satisfied, and the total fitness value for each trial represents the total violations of soft constraints at generation less than 1000. The best fitness value for each dataset may be further minimized if more trials and a maximum generation number larger than 1000 were considered.

For dataset ‘instance 1’, the best fitness value is 265; i.e. 200 students without a seat, 7 courses scheduled less than a specified minimum days, and 15 free gaps between lectures of the same group on the same day 

\[200 + 7(7) + 2(15) = 265\].

For dataset ‘instance 2’, the best fitness value is 11; i.e. 0 student without a seat, 1 courses scheduled less than a specified minimum days, and 3 free gaps between lectures of the same group on the same day 

\[0 + 5(1) + 2(3) = 11\].

For dataset ‘instance 3’, the best fitness value is 50; i.e. 18 students without a seat, 4 courses scheduled less than a specified minimum days, and 6 free gaps between lectures of the same group on the same day 

\[18 + 5(4) + 2(6) = 50\].

The results from three different course timetabling datasets have significantly shown that INACT is an effective optimization algorithm; can successfully be applied to solve (and optimize) various kinds of university course timetabling problems.

7.DISCUSSION AND FUTURE WORK

This paper has presented an immune-based optimization algorithm for course timetabling, called INACT (immune network algorithm for course timetabling). The algorithm shows great promise in the area of educational timetabling, particularly in its ability to consider, solve, and optimize the wide variety of different course timetabling problems. The algorithm can handle the hard constraints and soft constraints very well.

The experimental results on three benchmark course timetabling datasets, available on the internet, have significantly shown that INACT is an effective optimization algorithm, can successfully be applied to solve various kinds of course timetabling problems. The datasets may also be considered as multiobjective course timetabling problems since three soft constraints were considered. Hence, it may also be concluded that INACT may successfully be applied to solve multiobjective optimization problems.

For future work, other artificial immune algorithms, such as negative selection algorithm, will be applied to university course timetabling using the same course datasets.

REFERENCES


