Automatic Timetabling using Artificial Immune System

Yulan He, Siu Cheung Hui and Edmund Ming-Kit Lai

School of Computer Engineering, Nanyang Technological University
Nanyang Aveue, Singapore 639798
{asylhe,asschui,asmklai}@ntu.edu.sg

Abstract. University timetabling problem is a very common and seemingly simple, but yet very difficult problem to solve in practice. While solution definitely exists (evidenced by the fact that we do hold classes), an automated optimal schedule is very difficult to derive at present. There were successful attempts to address this problem using heuristics search methods. However, until now, university timetabling is still largely done by hand, because a typical university setting requires numerous customized complicated constraints that are difficult to model or automate. In addition, there is a problem of certain constraints being inviolable, while others are merely desirable. This paper intends to address the university timetabling problem that is highly constrained using Artificial Immune System. Empirical study on course timetabling for the School of Computer Engineering (SCE), Nanyang Technological University (NTU), Singapore as well as the benchmark dataset provided by the Metaheuristic Network shows that our proposed approach gives better results than those obtained using the Genetic Algorithm (GA).

1 Introduction

Over four decades, timetabling problem has been a major attraction of scientists from various disciplines. A practical timetabling problem usually involves complex constraints and a large number of events and is considered as *NP-hard* [1]. Traditional methods used to solve the timetabling problem include graph coloring with constraint manipulation [2] and clustering algorithm [3]. The graph coloring algorithm represents the timetabling problem as graphs where events (courses/exams) are represented as vertices and conflicts between the events are represented by edges. The constraint manipulation is done by scheduling the nodes with maximal degree (large number of conflicts) early. The clustering algorithm splits a set of events into groups which satisfy hard constraints and then the groups are assigned to time periods to fulfil the soft constraints. Both methods can only produce sub-optimal solutions.

Over the last two decades a variety of meta-heuristic approaches such as simulated annealing [4], tabu search [5], genetic algorithms (GA) [1, 6] and hybrid approaches [7] have been investigated for timetabling. In meta-heuristic the emphasis is on performing a deep exploration of the most promising regions of the solution space. Meta-heuristic methods begin with one or more initial solutions and employ search strategies to avoid local optima. These meta-heuristic algorithms can produce high quality solutions but often incur high computational cost. Moreover, the procedures are usually context dependent and require finely tuned parameters which may make their extension to other situations difficult [8].

In this paper, the Artificial Immune System (AIS), and in particular the Clonal Selection Algorithm (CLONALG), is investigated to solve the university

timetabling problem. It has certain similarities to Genetic Algorithms (GA) since both of them could be characterized as evolutionary-like algorithms and are inspired by biological metaphors, though CLONALG is based on the biological metaphor of the immune system whilst GA is inspired by the Darwinian evolutionary theory. The preliminary experimental results indicate that CLONALG performs better than GA when tested on the university timetabling benchmark data.

The rest of the paper is organized as follows. Section 2 describes the university timetabling problem. Section 3 presents the background of artificial immune system and specifically the clonal selection algorithm. Applying the CLONALG algorithm on the university timetabling problem is discussed in section 4. Performance comparison of CLONALG and GA by applying these two algorithms on the benchmark data provided by the Metaheuristic Network [9] is discussed in section 5. Finally, section 6 concludes the paper.

2 University Timetabling Problem

A timetabling problem, which is a subset of scheduling problem, can be defined as a combinatorial optimization problem of assigning resources to events being placed in period of time, satisfying a set of predefined constraints. The combinatorial problems are typically classified as *NP-bard* as they take practically infinite time to find any optimal solution and are indeed computationally intractable.

This paper focuses on the university course timetabling problem. In any optimization problem, there are objectives, decisions to make, resources available and related constraints. In a course timetabling problem, resources available are faculty, students, subjects being offered, time periods and rooms. A solution must group these resources together to produce a timetable to adhere to certain conditions set by the timetabler managing the timetable.

A typical university timetable is characterized by the following elements:

- *Timeslot,* $T = \{t_0, t_1, \cdots, t_m\}$. It is defined as the time interval during which a lecture, a tutorial, or a laboratory takes place. Every day has 9 timeslots. Each timeslot has a default value of 1-hour duration and the default starting time is on half an hour boundary. Thus, the total number of timeslots in a 5-day week is 45.
- *Room,* $R = \{r_0, r_1, \cdots, r_n\}$. Each event must take place in a particular room. A room can be classified by their functions or properties. In general, there are three types of room: lecture theater, tutorial room, or laboratory.
- *Subject,* $S = \{s_0, s_1, \cdots, s_p\}$. A subject is described by a name, title, number of students enrolled.
- *Subject Grouping,* $G = \{g_0, g_1, \cdots, g_u\}$. For any set of subjects having students in common, they can be grouped together as one subject group. For example, every course year could have 3 different subject groups, the *main group*, in which students follow a normal path and take four years to get a Bachelor degree, the *accelerated group*, in which students take three and a half years to get the Bachelor degree, and the *take ahead* group, in which students are allowed to take subjects from higher course years.
- *Class groups,* $C = \{c_1, c_2, \cdots, c_v\}$. Students are divided into a set of class groups in order to be accommodated with a particular room's size. Hence, for a particular subject with n students, there will be (usually) 1 lecture group, $n/TutorialRoomCapacity$ tutorial groups, and $n/LaboratoryRoomCapacity$ laboratory groups.

– *Event,* $E = \{e_1, e_2, \cdots, e_w\}$. As mentioned above, a subject group contains a list of subjects, and each subject has i lectures, j class groups of tutorial and/or k laboratory groups. Each event must consist of the following five elements: the subject group, the subject, the class group, the allocated room, and the allocated timeslot.

Two types of constraints are defined for the timetabling problem, *hard constraints* and *soft constraints*. All feasible and meaningful solutions must satisfy hard constraints. However, some or all of the soft constraints may be violated provided that the penalty costs associated with them are kept to minimum.

Typical hard constraints are:

– *Room occupancy.* The first condition for an event to be liable for an allocation in room r_i and at time t_j is that no other events can happen in room r_i at time t_j.
– *Room type.* The type of the room must conform to the type of the event.
– *Room capacity.* The capacity of the room must be large enough to accommodate the event.
– *Conflict.* The conflict constraint only deals with events of the same subject group. Hence, to check the validity of an event e_k (to be allocated) in room r_i, and timeslot t_j, the system must check against any other events of the same subject group in the same timeslot t_j.
– *Lecture spread constraint.* It is only applicable to the lecture event. There can only be one lecture for one subject in a particular day and there must be no more than 3 lectures in total in the same day.

Typical soft constraints are:

– *Event spread constraint.* For each class group c_i, the system will check the number of lecture, tutorial, and also laboratory events from day 0 (Monday) to day 4 (Friday). The penalty will be given if the total number of events per day of each class group c_i is less than 2 or greater than 5.
– *Noon punishment.* Events should be avoided to be scheduled during the lunch time between 12:30pm to 1:30pm.
– *Consecutive event consideration.* It is preferable that the same type of events takes place in the same venue and are scheduled consecutively.
– *Last timeslot punishment.* Events, if possible, should not happen in the last timeslot of the day.

3 Artificial Immune System

Artificial Immune Systems (AIS) are inspired from nature immune systems. The powerful information processing capabilities of the immune system, such as feature extraction, pattern recognition, learning, memory, and its distributive nature provide rich metaphors for its artificial counterpart [10]. Specifically, three immunological principles are primarily used in a piecemeal in AIS methods. These include the immune network theory, the mechanisms of negative selection, and the clonal selection principles. In this paper, only clonal selection principles adapted from [11] will be discussed.

3.1 Clonal Selection Algorithm (CLONALG)

The clonal selection principle is used as the fundamental basis for the development of the clonal selection algorithm (CLONALG) [11]. The algorithm is

presented in Figure 1, and works as follows. First, a initial set of population P_r and an empty memory set M are generated (step 1). Then The *selection* process selects the n best cells or antibodies to generate a new population P_n based on the affinity measure (step 2). These n best individuals of the population are cloned (reproduced) by the *clonal* process, giving rise to a temporary population of clones, C (step 3). The higher the affinity, the larger the number of clones generated for each of the n selected antibodies. The affinity *maturation* process then mutates the antibodies to create the population C^* (step 4). During mutation, it assigns a lower mutation rate for higher affinity antibodies than low affinity antibodies. The idea is that the antibodies close to a local optimum need only be fine-tuned, whereas antibodies far from an optimum should move larger steps towards an optimum or other regions of the affinity landscape [12]. The *reselection* process reselects the improved antibodies from C^* to update the memory set M (step 5). Finally, The *diversity introduction* process replaces d low affinity antibodies with new ones N_d (step 6).

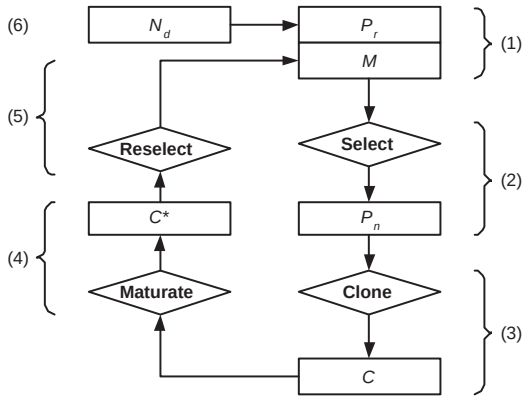


Fig. 1. Clonal Selection Algorithm (CLONALG).

The *selection* and *maturation* processes lead the population towards more stimulated to antigens, while the *clonal* and *diversity introduction* processes help to maintain the diversity of the population.

Although CLONALG with population-based search is characterized as an evolutionary-like algorithm, there are some important differences between CLONALG and genetic algorithm (GA) [4]. Firstly, CONALG was developed with inspiration from the immunological theory whereas GA was based on the Darwinian evolution. Secondly, in CLONALG, the solution is usually extracted from the memory pool constituted during the whole evolutionary process. In GA, the final solution is usually gathered from the population of the last generation of evolution.

4 Applying CLONALG to Timetabling

In order to apply CLONALG to the university timetabling problem, an antibody is represented as an *event* matrix $\mathbf{E_{rooms \times timeslots}}$. There are n number of rooms $\{r_0, r_1, \cdots, r_n\}$, m timeslots $\{t_0, t_1, \cdots, t_m\}$, and w events to be scheduled $\{e_0, e_1, \cdots, e_w\}$. $\mathbf{E[r_i][t_j] = e_k}$, means that an event e_k takes place in room r_i at time t_j.

4.1 Initialization

The CLONALG starts with a creation of an initial population which contains a set of feasible solutions, or antibodies, and is created randomly regardless to their affinity measurement value. Each antibody must follow the scheduling stages defined below. In the first stage, the most constraining events - events involved in largest number of constraints - are to be scheduled. The set of events requiring a special laboratory, or having to appear in the timetable in consecutive periods, or an event whose *room allocated*, *time allocated* attributes are set, have the higher weight and need to be scheduled first. Some events that have more relax constraints are to be scheduled next. For instance, the lecture events that require large-sized classrooms are randomly assigned to the corresponding rooms prior to those requiring small-sized ones. Apart from this, the rest of the events are randomly scheduled accordingly without violating any hard constraint.

4.2 Selection

The *selection* process begins with the measurement of the affinity of each antibody. These antibodies are then ordered according to the affinity calculated. The first antibody in the ordered list has the lowest affinity and the last one has the highest affinity.

Since the checking of hard constraints has been done beforehand, the affinity measurement function only deals with the soft constraints, and penalties will be given to each of the soft constraints violated respectively. The affinity measurement function is defined as

$$f = \frac{1}{1 + \sum_{i=1}^{k} w_i \cdot n_i} \tag{1}$$

where k is the total number of the soft constraints defined, n_i is the number of a certain kind of soft constraints within a particular antibody, w_i is its attached penalty or weight.

After the ordering of antibodies, n highest affinity antibodies are selected to produce a new population P_n. If we choose $n = N$, i.e., the number of highest affinity individuals equals to the number of candidates, each member of the population will constitute a potential candidate solution locally, characterizing a greedy search. In addition, if all the individuals are accounted locally, their clones will have the same size [1]. The value of the parameter n was determined empirically and will be elaborated in section 5.

4.3 Cloning

Antibodies in the population will be duplicated proportional to their affinity and enters the clone population C of size N_c, which is computed by equation 2

$$N_c = \sum_{i=1}^{n} \text{round}(\frac{\beta \cdot N}{i}) \tag{2}$$

where N_c is the total amount of clones generated, β is a multiplying factor, N is the total amount of antibodies and round$(\cdot)$ is the operator that rounds its argument towards the closest integer. Each term of this sum corresponds to the

[1] In order to maintain the best antibodies for each clone during evolution, it is possible to keep one original (parent) antibody for each clone unmutated during the maturation process.

clone size of each selected antibody, e.g., for $N = 100$ and $\beta = 1$, the highest affinity antibody ($i = 1$) will produce 100 clones, while the second highest affinity antibody produces 50 clones, and so on.

4.4 Maturation

In the CLONALG algorithm, the mutation rate of a cell is inversely proportional to the affinity of the cell. It gives the chance for low affinity cells to "mutate" more in order to improve its affinity. Since the mutations always result in better affinity antibodies, the immune system always climbs up the hill towards higher affinity antibody, leading to local optima.

A timetable **E** picked from the *selection* process will be mutated with mutation rate inversely proportional to its affinity measurement function. The mutation rate affects the maximum iteration of the hill climbing procedure. Hill climbing has two criteria to stop: maximum iteration or minimum soft constraints violated (which is set to 0).

At each iteration, the algorithm selects randomly a scheduled event to be moved. The selection of the move can be made either by randomly sampling a set of moves, or by exhaustively exploring the neighborhood looking for the best move (also known as Steepest Descent). In any way, the move should not lead to an infeasible timetable. The cost of applying the move to the chosen event will be calculated, which accounts for the number of violated soft constraints. If it gives a better value, or the same as before, the move will be accepted and applied to the timetable, moving an event from one room-timeslot to another room-timeslot.

4.5 Reselection and Diversity Introduction

There is a slight modification in the *reselection* step of the CLONALG algorithm (Figure 1), instead of choosing n highest affinity clones, the clones will be first compared with their parents. If one of the clones gives a better affinity than its parent, then the clone will be selected to enter the new population of P_n. If the parent is better, then the parent will be selected. Therefore, the n highest affinity antibodies, either the parents or the clones, will be selected to compose the new population of P_n, and d low affinity antibodies are to be replaced after every 5 generations by the *diversity introduction* process. This scheduling is supposed to leave a breathing time in order to allow achievement of the local optima, followed by replacement of the poorer individuals.

5 Experiments

The proposed approach has been developed to solve two problems, the timetabling problem for the School of Computer Engineering (SCE), Nanyang Technological University, and the university course timetabling benchmark problem. To cater for two different inputs and constraints defined by the two problems, two separate systems have been developed based on the CLONALG algorithm.

For the SCE timetabling problem, the system has been tested using a sample dataset of SCE year 1 main group, academic year 2002/2003, semester 1 and incorporating the constraints that have been discussed in section 2. Another system, which was built to solve the benchmark problem, has been tested using the small instances of the datasets which can be downloaded from http://iridia.ulb.ac.be/~msampels/ttmn.data/. The characteristics of both problems are given in

Table 1. Characteristics of the timetabling problems

	SCE Problem	*Benchmark Problem*
Num events	116	100
Num rooms	8	5
Num features	-	5
Approx features per room	-	3
Percent feature use	-	70
Num students	450	80
Max events per student	30	20
Max students per event	200	20
Num subject groupings	5	-

Table 1. Entries represented as "-" means the corresponding characteristic is undefined for that particular problem.

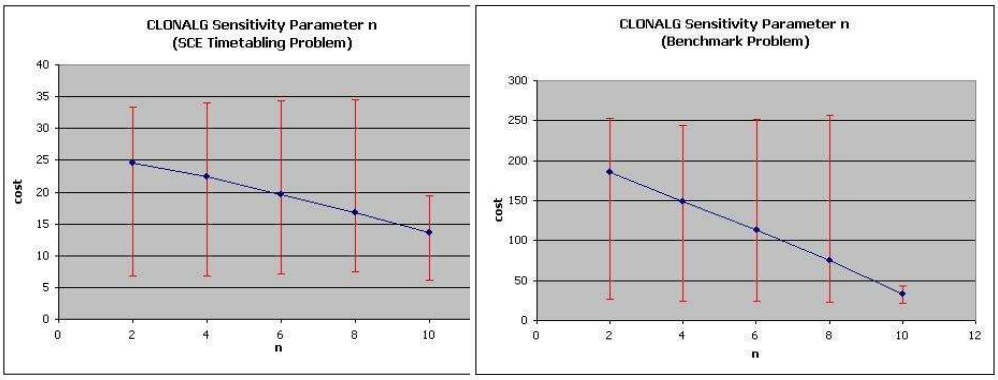
CLONALG has several user-defined parameters and different settings of these parameters would affect the performance of the algorithm. Two parameters are discussed here, namely n, the number of antibodies to be selected for cloning, and β, parameter affecting the number of clones generated from the antibody population.

5.1 Sensitivity with Relation to n

CLONALG sensitivity with relation to n has been evaluated with β (of equation 2) being fixed to 1. Figure 2 presents the results on the maximum, minimum and mean obtained after running CLONALG 10 times with 50 and 100 generations for SCE timetabling and benchmark problem respectively. n takes the value ranging from 2 to 10. The figure shows that the average cost of calculating the antibody affinity in the population decreases as n increases. This behavior was expected, because the higher the value of n, the larger the number of antibodies to be mutated and eventually leads to the better solutions. On the other hand, n has a strong influence on the size of the antibody clone population as described in equation 2, and larger values of n imply a higher computational cost. The computational time for different settings of n is given in Table 2.

Table 2. Computational time versus different settings of n

	Computational Time (sec)	
n	*SCE Problem*	*Benchmark Problem*
2	11.9	16.6
4	15.4	31.8
6	17.6	48.3
8	19.6	64.8
10	20.7	86.9



(a) SCE timetabling problem. (b) Benchmark problem.

Fig. 2. CLONALG sensitivity with relation to n.

5.2 Sensitivity with Relation to β

To study the CLONALG sensitivity with relation to β, n was set to N (population size), and β took the following values, $\beta = \{0.1, 0.2, 0.4, 0.8, 1.5\}$. It can be observed from Figure 3 that the cost of calculating the antibody affinity in the population converges when β is set to 0.8 or higher. This is because the probability of getting better antibodies in a larger clonal population is higher. Nevertheless, the computational time increases linearly with β as can be seen from Table 3.

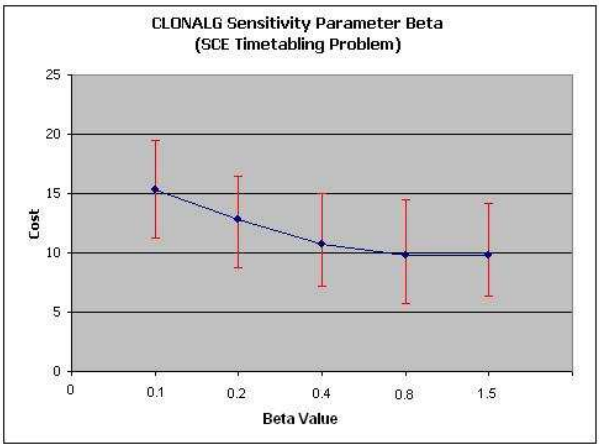


Fig. 3. CLONALG sensitivity parameter β.

5.3 Comparison of CLONALG and GA

The results presented in Table 4 are the best affinity antibodies out of 20 runs of CLONALG, with the number of generations set to 100, population size set to 10, best antibody set to 8, and the number of antibodies to be replaced at each generation set to 2.

Table 3. Computational time versus different settings of β

β	0.1	0.2	0.4	0.8	1.5
Computational Time (sec)	13.6	18.7	29.9	50.3	88.1

The solution score is calculated based on the rules defined in [13] which essentially counts the number of soft constraint violations. The smaller the solution score, the better the algorithm is. It can be observed from Table 4 that CLONALG gives a better result than GA but with longer computational time.

Table 4. Results comparison of different dataset of Benchmark Problem

	CLONALG		**GA**	
	Solution Best	*Computation*	*Solution Best*	*Computation*
Data Set	*Score*	*Time (sec)*	*Score*	*Time (sec)*
small1.tim	19	62.9	36	12.2
small2.tim	24	62.6	37	12.8
small3.tim	20	63.1	38	11.1
small4.tim	14	81.3	51	18.8
small5.tim	12	61.2	31	12.7

By analyzing the two algorithms, CLONALG and GA, and the results obtained, it is noticed that CLONALG maintains a diverse set of local optimal solutions, while GA tends to polarize the whole population of individuals towards the best one. This is mainly due to the selection and reproduction schemes adopted by CLONALG. The coding schemes and evaluation functions of these two algorithms are essentially quite similar, but their evolutionary search is different. Another important aspect of CLONALG compared to GA is the fact that CLONALG takes into the account of the cell affinity, corresponding to an individual's fitness, in order to define the mutation rate applied to each member of the population. GA adopts the genetic operators that disregard the individual fitness.

6 Conclusion

This paper applies the Clonal Selection Algorithm (CLONALG) to solve the course timetabling problem for the School of Computer Engineering (SCE), Nanyang Technological University (NTU), Singapore as well as the university course timetabling problem provided by the Metaheuristics Network.

CLONALG, which is the algorithm based on the biological metaphor of the immune system, is proved to be effective to diverge the population. By comparing CLONALG with GA, it is observed that the main steps composing the GAs are embodied in CLONALG, allowing us to characterize it as an evolutionary-like algorithm. However, while GA uses a vocabulary borrowed from the natural genetics and are inspired in the Darwinian evolution, the CLONALG makes use

immunological terminology to describe the Antigen-Antibody interactions and cellular evolution. CLONALG performs its search through the mechanisms of somatic mutation and receptor editing, balancing the exploitation of the best solutions with the exploration of the neighborhood.

For future work, we will test the system with SCE full data (four years courses) and add the lecturer constraints, etc. Some improvements on the heuristics used in producing the initial candidates could also be made in order to get better final solutions. In addition, implementation of other algorithms is essential for further studies, to compare the results of the two problems investigated here.

References

1. E. Yu and K.S. Sung. A genetic algorithm for a university weekly courses timetabling problem. *International Transactions in Operational Research*, 9(6):703–717, 2002.
2. E.K. Burke, D.G. Elliman, and R.F. Weare. A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, 27(1):1–18, 1994.
3. N. Balakrishnan, A. Lucena, and R. T. Wong. Scheduling examinations to reduce second-order conflicts. *Computers and Operational Research*, 19(5):353–361, 1992.
4. E.K. Burke, A. Eckersley, B. McCollum, S. Petrovic, and R. Qu. Using simulated annealing to study behaviour of various exam timetabling data sets. In *Proceedings of the Fifth Metaheuristics International Conference (MIC 2003)*, Kyoto, Japan, August 2003.
5. B. Jaumard, J.-F. Cordeau, and R. Morales. *Efficient Timetabling Solution with Tabu Search.* http://www.idsia.ch/Files/ttcomp2002/jaumard.pdf, 2003. Avaliable from Metaheuristics Network - Intenational Timetabling Competition.
6. E.K. Burke, D.G. Elliman, and R.F. Weare. A genetic algorithm based university timetabling system. In *Proceedings of the 2nd East-West International Conference on Computer Technologies in Education*, pages 35–40, Crimea, Ukraine, September 1994.
7. M. Chiarandini, K. Socha, M. Birattari, and O. Rossi-Doria. An effective hybrid approach for the university course timetabling problem. *Journal of Scheduling*, 2003. To appear.
8. G. Laporte, M. Gendreau, J-Y.Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7:285–300, 2000.
9. *Metaheuristics Network.* http://www.metaheuristics.org/.
10. D. Dasgupta, Z. Ji, and F. Gonzlez. Artificial immune system (ais) research in the last five years. In *Proceedings of the International Conference on Evolutionary Computation Conference (CEC)*, Canbara, Australia, December 2003.
11. L.N. de Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 6(3):239–251, 2002.
12. L.N. de Castro and J.I. Timmis. Artificial immune system as a novel soft computing paradigm. *Soft Computing*, 7(8):526–544, 2003.
13. *University Course Timetabling Benchmark Solution Score Calculation.* http://www.idsia.ch/Files/ttcomp2002/IC_Problem/node2.html.