A NEGATIVE SELECTION ALGORITHM FOR UNIVERSITY EXAMINATION TIMETABLING

MUHAMMAD ROZI MALIM, AHAMAD TAJUDIN KHADER, ADLI MUSTAFA

Faculty of Information Technology & Quantitative Science, UiTM, 40450 Shah Alam, Malaysia
rozi@tmsk.uitm.edy.my

School of Computer Science, USM, 11800 Minden, Penang, Malaysia
tajudin@cs.usm.my

School of Mathematical Science, USM, 11800 Minden, Penang, Malaysia
adli@cs.usm.my

Abstract: The university examination timetabling is known to be a highly constrained optimization problem. Metaheuristic approaches, such as simulated annealing, tabu search, and genetic algorithm, have successfully been applied to solve the problem. The negative selection algorithm, an algorithm inspired by the immune system, has successfully been applied to detect computer viruses, tool breakage detection, anomaly detection, and network intrusion detection. This paper presents a negative selection algorithm for the university examination timetabling problems with the main objective to show that the algorithm may be tailored for educational timetabling. Another objective is to show that the algorithm may produce good quality exam timetables, as good as other optimization algorithms (metaheurictics). The experimental results, using the benchmark datasets, have shown the effectiveness of the algorithm by producing good quality exam timetables, as good as metaheuristic approaches. For future work, the negative selection algorithm will be compared with other artificial immune algorithms using the same datasets.

Keywords: Examination timetabling, artificial immune system, Negative Selection Algorithm.

1. Introduction

The construction of an *examination timetable* is a common problem for all institutions of higher education. Usually it involves taking the previous semester's timetable and modifying it so it will work for the new semester. The examination timetabling is known to be a highly constrained combinatorial optimization problem. The common approaches to this problem may be classified as *global algorithms* such as integer programming and goal programming, *constructive heuristics* such as sequential heuristics and constraint logic programming, and *metaheuristics* such as simulated annealing (SA), tabu search (TS), and evolutionary algorithms (EA) (Carter and Laporte, 1998).

Artificial immune system (AIS), a new branch of Artificial Intelligence, is a new intelligent problem-solving technique that being used in optimization and scheduling problems (Hart and Ross, 1999). In the literature, the authors have shown that the AIS algorithms are more efficient than the classical heuristic scheduling algorithms such as SA, TS, and genetic algorithm (GA) (Malim *et al.*, 2004). Artificial immune systems have been more successful than GA and other methods in applications of 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 (Hart *et al.*, 1998).

This paper presents an artificial immune algorithm called *negative selection algorithm* (NSAET) for the university examination timetabling problems. The main objective is to show that the algorithm may be tailored for educational timetabling. Another objective is to show that the algorithm may produce good quality examination timetables, as good as other optimization algorithms such as SA, TS, and GA. Twelve benchmark examination datasets have been used to implement and test the algorithm. The experimental results have significantly shown that the NSAET is an effective optimization algorithm. A comparison with other solution methods (metaheuristics) have shown the effectiveness of the algorithm by producing good quality examination timetables, as good as metaheuristic approaches.

2. Examination Timetabling Problem

Examination timetabling problem (ETP) is a specific case of the more general timetabling problem. The examination timetabling regards the scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for the students as much as possible (Di Gaspero and Schaerf, 2001). Given a set of exams, a set of (contiguous) timeslots, a set of students, and a set of student enrollments to exams, the problem is to assign exams to timeslots subject to a variety of *hard* and *soft* constraints.
Hard constraints must be satisfied in order to produce a *feasible* timetable. The main hard constraints in examination timetabling are usually represented by the following:

i) Every exam in the set must be assigned to exactly one timeslot of the timetable.
ii) No individual should be timetabled to be in two different places at once. i.e. any two exams which have students in common must not both be scheduled in the same timeslot.
iii) There must be sufficient resources available in each timeslot for all the exams timetabled, e.g. room capacities must not be violated.

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. The violation of soft constraints should be minimized. 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 examination timetabling are:

i) Spreading exams - students should not have exams in consecutive (adjacent) timeslots.
ii) Time assignment - an exam may need to be scheduled in a specific timeslot.
iii) Time constraints - an exam may need to be scheduled before, after or at the same time as another.
iv) Resource assignment - an exam must be scheduled into a specific room.

The examination timetabling problem can be seen as consisting of *two* subproblems (Anh and Hoa, 2004): (1) assigning exams to timeslots, and (2) assigning exams (with timeslots) to rooms. For real-life situations, these two subproblems can be solved separately.

3. Artificial Immune System and Negative Selection Algorithms

The 'artificial immune system' is an approach which used the natural immune system as a metaphor for solving computational problems, *not* modeling the immune system (Timmis, 2001). 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 (Hart, 2002). 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 (de Castro, 2002). For examples, the *Artificial Immune Networks* by Farmer *et al.* (1986), the *Clonal Selection Algorithm* by de Castro and Von Zuben (2000), and the *Negative Selection Algorithm* by Forrest *et al.* (1994). Immune network models are suitable to deal with dynamic environments and optimization problems, 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.

3.1 *The negative selection of T-cells and B-cells*

The negative selection of T-cells can occur within or outside the *thymus*. Negative thymic selection stems from interactions of immature T-cells with the self-peptides presented by the self-MHC molecules. This process results in the death of an activation-dependent cell, thereby purging potentially autoreactive T-cells from the repertoire. T-cells bearing useless TCRs (T-cells Receptors) that do not exhibit significant interactions with any self-MHC ligands are lost from the repertoire through positive selection. The time and extension of this process of deletion depends upon the *affinity* of the binding between the TCR and the self-antigen. T-cells that bind with higher affinities to the self-antigen are purged more effectively from the repertoire than those with lower affinities. Although the negative selection nearly eliminates the entirety of developing thymocytes, self-reactive T-cells can still escape from the thymus and circulate in the periphery as fully immuno-competent T-cells. These self-reactive T-cells can pose a threat of an autoimmune disease taking hold of the host.

T-cell tolerance alone would be insufficient protection against autoimmunity. Immature B-cells within the *bone marrow* are especially sensitive to tolerance induction. Mature B-cells can also be rendered tolerant if they encounter an antigen in the absence of both T-cell help and co-stimulatory influences. As with the T-cells, self-reactive B-cells can also escape the central B-cell negative selection. In this case, B-cell activation or tolerance will be the result of the number, the strength, and the time when the costimulatory signals arise. A fast and sudden ligation of the receptor to the antigen will generally induce a clonal response. At the same time, a constant and relatively weak stimulation will lead to tolerance, characterized by the inhibition of the clonal response and further cellular apoptosis.

3.2 *Negative Selection Algorithms*

The *negative selection algorithm* (NSA) is one of the most widely used techniques in AIS (Gonzalez *et al.*, 2003). It is primarily used to detect changes in data/behavior patterns by generating detectors in the complementary space. The NSA is based on the principles of self-nonself discrimination (Forrest *et al.*, 1994). The algorithm was inspired by the thymic negative selection process that intrinsic to natural IS, consisting of screening and deleting self-reactive T-cells. The NSA takes considerable time (exponential to the size of the self data) and produce redundant detectors (Gonzalez, 2005). This time/size limitation motivated the development of different approaches to generate the set of candidate detectors. The NSA also termed the exhaustive detector generating algorithm.

The standard NSA proposed by de Castro (2002) can be summarized as follows:

1. *Initialization:* Randomly generate strings and place them in a set *P* of immature T-cells. Assume all molecules (receptors and self-peptides) are represented as binary strings of the same length *L*.
2. *Affinity evaluation:* Determine the affinity of all T-cells in *P* with all elements of the self set *S*.
3. *Generation of the available repertoire:* If the affinity of an immature T-cell with at least one self-peptide is greater than or equal to a given cross-reactive threshold *T*, then the T-cell recognizes this self-peptide and has to be eliminated; otherwise T-cell is introduced into the available repertoire *A*.

The basic algorithmic steps of the NSA by Dasgupta *et al.* (2004) can be presented as follows:

1. Define self as a collection *S* of elements in a feature space *U*, a collection that needs to be monitored.
2. Generate a set *F* of *detectors*, each of which fails to match any string in *S*. An approach that mimics the IS generates random detectors and discards those that match any element in the self set.
3. Monitor *S* for changes by continually matching the detectors in *F* against *S*. If any detector ever matches, then a change is known to have occurred, as the detectors are designed not to match any representative samples of *S*.

Gonzalez and Dasgupta (2003) presented a *real-valued NSA* (RNS) for anomaly detection. The algorithm applies a heuristic process that changes iteratively the position of the detectors driven by two goals: to maximize the coverage of the nonself subspace and to minimize the coverage of the self samples. The RNS detector generation starts with a population of candidate detectors, which are then matured through an iterative process. In particular, the center of each detector is chosen at random and the radius is a variable parameter which determines the size of the detector. At each iteration, the radius of candidate detector is calculated, and the ones that fall inside self region are moved (i.e. its center is successively adjusted by moving it away from training data and existing detectors). The set of nonself detectors are then stored and ranked according to their size (radius). The detectors with larger radii (and smaller overlap with other detectors) are considered as better-fit and selected to go to the next generation. Detectors with very small radii, however, are replaced by the clones of better-fit detectors. The clones of a selected detector are moved at a fixed distance in order to produce new detectors in its close proximity. Moreover, new areas of the nonself space are explored by introducing some random detectors. The whole detector generation process terminates when a set of mature (minimum overlapping) detectors are evolved which can provide significant coverage of the nonself space.

4. Negative Selection Algorithm for Examination Timetabling (NSAET)

Figure 1 shows the negative selection algorithm developed for the examination timetabling problems. This algorithm, called *NSAET* (Negative Selection Algorithm for Examination Timetabling), was developed based on the natural negative selection process (T-cell and B-cell), the standard NSA proposed by de Castro (2002), and the RNS algorithm presented by Gonzalez and Dasgupta (2003).

```
1.   Initialization: initialize a population of candidate detectors (initial feasible timetables):
     randomly generate an initial population of detectors (feasible timetables)
     while number of detectors (timetables) is less than population size
       for each detector (timetable)
         randomly select exam one by one
           assign each exam to the first available timeslot and room satisfying hard constraints
         check for an identical detector (duplicate timetable)
           if the detector (timetable) is identical with another detector
             remove the detector (timetable)
     end while (an initial population is generated)
2.   Population loop:
     for each population of detectors (feasible timetables)
2.1  Censoring: Determine the Fitness of each timetable and Average fitness of current population, and
     eliminate detectors with fitness greater than or equal to average:
       for each detector (timetable) in the current population
         determine the fitness value via a fitness function (soft constraints)
       determine the average fitness for the current population
       for each detector
         if the fitness >= average
           eliminate the detector (timetable)
       if all fitness values are equal
         eliminate only the second half of the detectors (timetables)
2.2  Monitoring: Generate new detectors (feasible timetables) to replace the eliminated detectors by cloning
     and mutating the remaining detectors. Only new detectors with fitness less than or equal to average will be
     accepted for the new population:
       while the number of detectors (timetables) < population size
         randomly select a detector (timetable) according to fitness using roulette wheel
           clone the detector (timetable)
           while mutation = failure
             mutate the clone to produce a new detector (feasible timetable)
               randomly select two exams and exchange the timeslots and rooms
             if all hard constraints are satisfied
               if the new detector is not identical with other detectors (no duplicate timetables)
                 mutation = success, and determine the fitness
           end while
         if the fitness of the new detector <= average
           add the new detector to the new population
         else eliminate the new detector
       end while
3.   Cycle:
     repeat population loop until a given convergence criterion is met.
```

Fig. 1. Negative Selection Algorithm for Examination Timetabling (NSAET)

5. Benchmark Datasets

The *twelve* examination timetabling datasets used in the implementation of the NSAET are available on Internet from ftp://ftp.mie.utoronto.ca/pub/carter/testprob/, called *Carter datasets*. These datasets provide a reasonable benchmark problems for comparison of different methods or algorithms. The datasets and characteristics are shown in Table 1. Each of the datasets come in two files, one file (*course data file*) contains the list of courses, and the other (*student data file*) contains a list of student-course selections. The courses and student-course selections are sorted in ascending order.

Table 1. Examination Datasets and Characteristics

Code	University	No. of Exams	No. of Students	No. of Enrollments	Timeslot Capacity
car-f-92	Carleton University 1992	543	18419	55522	2000
car-s-91	Carleton University 1991	682	16925	56877	1550
ear-f-83	Earl Haig Collegiate 1983	190	1125	8109	350
hec-s-92	Ecole des Hautes Etudes Com. 1992	81	2823	10632	650
kfu-s-93	King Fahd University 1993	461	5349	25113	1955
lse-f-91	London Sch. of Econ. 1991	381	2726	10918	635
rye-s-93	Ryerson University 1993	486	11483	45051	2055
sta-f-83	St. Andrews High Sch. 1983	139	611	5751	465
tre-s-92	Trent University 1992	261	4360	14901	655
uta-s-92	Uni. of Toronto, Arts & Science 1992	622	21266	58979	2800
ute-s-92	Uni. of Toronto, Engineering 1992	184	2750	11793	1240
yor-f-83	York Mills Collegiate 1983	181	941	6034	300

6. Implementation

First of all, before implementing the algorithm (NSAET), each of the examination timetabling problems will be formulated as a 0-1 integer programming model. The model would assist in encoding the algorithm into C++ programming codes.

6.1 *Mathematical model*

Using the general model by Malim *et al.* (2005), the formulation may be carried out as follows:

- Since there are *three* sets of variables (exam, student, timeslot), only the matrices exam-student (E), student-conflict (C), and exam-timeslot (Q) need to be constructed; the first two are input matrices and the other is the output matrix. The *room assignment* is not considered.
- *Three* hard constraints are considered for each of the datasets:
  i) No students must be assigned to two different exams at the same timeslot (first-order conflict);
  ii) Timeslot capacity must not be exceeded;
  iii) Each exam must be assigned to exactly one timeslot (all exams are scheduled).
- Only *one* soft constraint is considered for all datasets. This constraints will be used to evaluate the fitness function of each feasible timetable:
  'No students should be assigned to two exams in adjacent timeslots (second-order conflict)'.
  The penalty value (fitness function) for each student that violated this constraint is '1'.
- The 0-1 integer programming model (*exam-timeslot assignment*) for each of the datasets may be formulated as follows:

$$\text{minimize} \quad \sum_{i=1}^{n_1-1}\sum_{j=i+1}^{n_1} c_{ij}\cdot prox(t_{(e_i)},t_{(e_j)}) \tag{1}$$

$$\text{subject to} \quad \sum_{k=1}^{n_2}\sum_{j=1}^{n_1}\sum_{i=1}^{n_1} c_{ij}\cdot q_{ik}\cdot q_{jk}=0 \tag{2}$$

$$\sum_{j=1}^{n_2} x_c(e_i,t_j)=0 \tag{3}$$

$$\sum_{i=1}^{n_1} x(e_i,t_j)=0 \tag{4}$$

all variables are integers 0-1;

where c_{ij} is the number of students taking both exams e_i and e_j; $prox(t_{(e_i)},t_{(e_j)})=1$ if $|t_{(e_i)}-t_{(e_j)}|=1$, 0 otherwise; $t_{(e_k)}$ specifies the assigned timeslot for exam e_k; q_{ij} are the matrix entries Q taking values 0 or 1; $x_c(e_i, t_j) = 0$ if $\sum_{i=1}^{n_1} n_s(e_i) \cdot q_{ij} \leq n_c(t_j)$, 1 otherwise; $n_s(e_i)$ is the number students in exam e_i; $n_c(t_j)$ is the maximum-capacity of students for each timeslot; $x(e_i, t_j) = 0$ if $\sum_{j=1}^{n_2} q_{ij} = 1$, 1 otherwise; n_1 = number of exams; and n_2 = number of timeslots.

6.2 *Implementation of the Negative Selection Algorithm (NSAET)*

For each dataset, the NSAET, presented in Figure 1, may be implemented as follows:

1. Initial timetables:
Ten (10) different initial feasible timetables are generated using a *simple random selection algorithm*, i.e. each exam is selected at random and assigned to a randomly selected timeslot satisfying all hard constraints (no student conflicts, timeslot-capacity not exceeded, and all exams are scheduled). For each initial timetable, if an identical timetable already generated in the initial population, the timetable must be eliminated and the algorithm will generate a new feasible timetable.

2. Population loop:
Censoring process - the *fitness* value of each timetable and the *average* fitness of the current population are determined, and the timetables with fitness values greater than or equal to average are eliminated.

During the censoring process, the fitness value of each timetable in the current population is evaluated using the fitness function; this value represents the number of students having two exams in adjacent timeslots. Total fitness represents the sum of fitness values of all timetables in the current population. Then the average fitness for the current population is determined. The average fitness is equal to total fitness divide by the number of timetables or population size. For each timetable, if the fitness is less than average, the timetable will be accepted for the new population, otherwise it will be eliminated.

Monitoring (reproduction) process - new feasible timetables are generated to replace the eliminated timetables by *cloning* and *mutating* the remaining timetables.

During the monitoring process, only new timetables with fitness less than or equal to average will be accepted for the new population. A timetable is randomly selected from the current population for cloning according to fitness using a *Roulette Wheel* selection method. Cloning copies good timetables from current population to next generation population. It is expected that the timetables with low fitness will be selected for cloning. Only one clone is produced for each selected timetable. This give rise to a duplicates timetable. This duplicate timetable needs mutation to form a new feasible timetable. The mutation operator works by taking two exams at random and exchange the timeslots of the two exams, always maintaining a feasible timetable. The mutation process is repeated until the cloned timetable has satisfied all hard constraints and no duplicates. If the fitness of the new timetable is less than of equal to average, the new timetable will be added to the new population. Otherwise, if the fitness is larger than average, the new timetable will be eliminated. The monitoring process will be repeated until the number of timetables for the new population is equal to population size.

3. Cycle:
The process (*population loop*) will be repeated until the maximum number of generations is exceeded, or until the maximum number of none improvement generations is equal to 100.

7. Experimental Results

7.1 *First results for Negative Selection Algorithm (NSAET)*

The NSAET has been implemented on the twelve Carter datasets. The following (Table 2) are the *first* experimental results on solving examination timetabling problems using NSA. The algorithm was run on each dataset for *five* trials, and the maximum number of generations 500 was used. The best fitness value and the average fitness value for each dataset, based on the five trials, have been recorded.

Table 2. First Results on Solving UETP using NSAET

Institution	No. of Exams	No. of Students	No. of Student Enrollments	Timeslot Capacity	No. of Timeslots	Fitness Values	
						Best	Average
car-f-92	543	18419	55522	2000	31	357	411.6
car-s-91	682	16925	56877	1550	40	406	454.2
ear-f-83	190	1125	8109	350	24	65	85.6
hec-s-92	81	2823	10632	650	19	7	13.2
kfu-s-93	461	5349	25113	1955	20	1	2.6
lse-f-91	381	2726	10918	635	18	162	188
rye-s-93	486	11483	45051	2055	24	161	221.6
sta-f-83	139	611	5751	465	14	0 (169)	1
tre-s-92	261	4360	14901	655	25	40	61.6
uta-s-92	622	21266	58979	2800	32	209	240.6
ute-s-92	184	2750	11793	1240	10	0 (269)	3.2
yor-f-83	181	941	6034	300	22	1	2.4

The *number of timeslots* used for all datasets were imposed according to those given in Carter's website. However, the number of timeslots for all datasets may be further reduced if necessary. The *fitness value* (soft constraint violations) is the minimum number of students having two exams in adjacent timeslots at generation 500 or less. The best fitness values of the datasets *sta-f-83* and *ute-s-92* have converged to '0' at generations 169 and 269, respectively. The fitness values for other datasets may converge to '0' if a number of generations much larger than 500 is used.

The results from 12 different examination datasets have significantly shown that the NSAET is an effective optimization algorithm; can successfully be applied to solve (and optimize) various kinds of university examination timetabling problems.

7.2 *Comparing NSAET with other solution methods*

A comparison with other published results was also conducted. This is to access the effectiveness of the NSAET against other optimization algorithms. Only *five* datasets were considered; car-f-92, car-s-91, kfu-s-93, tre-s-92, and uta-s-92. The following are the authors and metaheuristic approaches used in the published results:

(A) Burke *et al.* (1996) – Memetic Algorithm.
(B) Di Gaspero and Schaerf (2001) – Tabu Search.
(C) Caramia *et al.* (2001) – A set of heuristics (Greedy Assignment, and Spreading Heuristic).
(D) Merlot *et al.* (2003) – Hybrid Algorithm (Constraint Programming, Simulated Annealing, and Hill-climbing.

All authors have considered the same hard and soft constraints, hence a direct comparison based on the fitness values (number of students having two exams in adjacent timeslots) may be carried out. The main goal is *not* to show that the NSAET is better than metaheuristic approaches, rather to show the algorithm may produce good quality examination timetables as good as metaheuristics. Table 3 summarizes the results.

Table 3. Comparison with Other Solution Methods

Code	No. of Exams	No. of Students	Timeslot Capacity	Timeslots	Fitness Values					
					A	B	C	D	NSAET	
									Best	Average
car-f-92	543	18419	2000	40	331	424	268	158	1 (1057)	4.4
car-s-91	682	16925	1550	51	81	88	74	31	3 (935)	5.8
kfu-s-93	461	5349	1955	20	974	512	912	247	0 (530)	2
tre-s-92	261	4360	655	35	3	4	2	0	0 (420)	1.8
uta-s-92	622	21266	2800	38	772	554	680	334	2 (1058)	3.8

The number of timeslots used for all datasets were imposed according to the papers of the published results. Hence, based on the results in Table 3, the NSAET obtained the best (average) fitness values in *four* datasets. The best fitness values of the datasets *kfu-s-93* and *tre-s-92* have converged to '0' at generations 530 and 420, respectively. However, the fitness value of the dataset tre-s-92, on average, has converged to '1.8' compared to '0' by Merlot's multi-stage method.

It may be concluded that the NSAET (negative selection algorithm) is capable of producing good quality examination timetables as good as metaheuristic approaches.

8. Conclusion and Future Work

This paper has successfully presented an artificial immune algorithm for the university examination timetabling problems, called NSAET (negative selection algorithm for examination 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 examination timetabling problems. The algorithm can handle the hard constraints and soft constraints very well. The experimental results on twelve benchmark datasets (Carter datasets), available on the Internet, have significantly shown that the NSAET is an effective optimization algorithm; can successfully be applied to solve various kinds of university examination timetabling problems.

A comparison with other optimization algorithms (published results) has significantly shown the effectiveness of NSAET by producing good quality (low fitness) examination timetables, as good as metaheuristic approaches. This algorithm may be accepted as a new member of optimization algorithms for solving examination timetabling problems.

For future work, the negative selection algorithm (NSAET) will be compared with other artificial immune algorithms, such as clonal selection algorithm (CSAET) and immune network algorithm (INAET), using the same examination timetabling datasets. These artificial immune algorithms may also be applied to university *course* timetabling.

References

Anh D.T., and Hoa L.K. (2004). Combining Constraint Programming and Simulated Annealing on University Exam Timetabling. *Proceedings International Conference RIVF'04*, February 2-5, Hanoi, Vietnam, pp:205-210.

Burke, E.K., Newell, J.P., and Weare, R.F. (1996) A memetic algorithm for university exam timetabling. In *Practice and Theory of Automated Timetabling, First International Conference*, E.K. Burke and P. Ross, Eds. Edinburgh, U.K.: Lecture Notes in Computer Science 1153, Springer-Verlag, 1996, pp:241-250.

Caramia, M., Dell'Olmo, P., and Italiano, G.F. (2001) New algorithms for examination timetabling. In *Algorithm Engineering 4th International Workshop, WAE 2000*, S. Nher and D. Wagner, Eds. Saarbrucken, Germany: Lecture Notes in Computer Science 1982, Springer-Verlag, 2001, pp:230-241.

Carter, M.W. and Laporte, G. (1998) Recent Developments in Practical Course Timetabling. In *Practice and Theory of Automated Timetabling, Second International Conference*, E.K. Burke and M.W. Carter, Eds. Lecture Notes in Computer Science 1408, Springer-Verlag, 1998, pp:3-19.

de Castro, Leandro Nunes (2002) Immune, Swarm, and Evolutionary Algorithms, Part I: Basic Models. *Proceeding of the ICONIP, Workshop on Artificial Immune Systems*. Vol. 3, pages: 1464-1468, Singapore 18-22 November 2002.

de Castro, L.N. and Von Zuben, F.J. (2000) The Clonal Selection Algorithm with Engineering Applications. *Proceedings of GECCO'OO* (Workshop Proceedings), pp:36-37.

Di Gaspero, L. and Schaerf, A. (2001) Tabu search techniques for examination timetabling. In *Practice and Theory of Automated Timetabling, Third International Conference*, E.K. Burke and W. Erben, Eds. Konstanz, Germany: Lecture Notes in Computer Science 2079, Springer-Verlag, 2001, pp:104-117.

Farmer, J.D., Packard, N.H., and Perelson, A.S. (1986) The immune system, adaptation, and machine learning. *Physica, 22D*, pp: 187-204.

Forrest, S., Perelson, A.S., Allen, L., and Cherukuri, R. (1994) Self-nonself discrimination in a computer. *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pp:202-212. Los Alamitos, CA: IEEE Computer Society Press.

Gonzalez, L. J. (2005) A Self-Adaptive Evolutionary Negative Selection Approach for Anomaly Detection. *PhD Thesis*, Nova Southeastern University, January 2005.

Gonzalez, F. and Dasgupta, D. (2003) Anomaly Detection Using Real-Valued Negative Selection. *Proceedings Genetic Programming and Evolvable Machines*, GPEM2003. 2004 Kluwer Academic Publishing, Netherlands, pages 383-403.

Gonzalez, F., Dasgupta, D., and Nino, L.F. (2003) A Randomized Real-Valued Negative Selection Algorithm. *Proceeding of Second International Conference on Artificial Immune Systems* (ICARIS), September 1-3, Napier University, Edinburgh, UK, pp: 261-272.

Hart, Emma. (2002) Immunology as a Metaphor for Computational Information Processing: Fact or Fiction? *PhD Thesis*, University of Edinburgh 2002.

Hart, E. and Ross, P. (1999) An Immune System Approach to Scheduling in Changing Environments. *Proceedings of the GECCO '99*, pp. 1559-1566.

Hart, E., Ross, P., and Nelson, J. (1998) Producing Robust Schedules Via An Artificial Immune System. *Proceedings of the ICEC'98*.

Malim, M.R., Khader, A.T., and Mustafa, A. (2004) Artificial Immune Systems for Scheduling Problems. *Proceedings of the National Seminar on Decision Science 2004 (SKSP2004)*, Faculty of Quantitative Sciences, UUM; 15-17 Dec. 2004, Penang.

Malim, M.R., Khader, A.T., and Mustafa, A. (2005) University Examination Timetabling: A General Model. *Proceedings of the National Symposium on Mathematical Sciences XIII (SKSMXIII)*, 31/5-2/6 2005, UUM, Holiday Villa, Alor Star, Kedah.

Merlot, L.T.G., Boland, N., Hughes, B.D., and Stuckey, P.J. (2003) A Hybrid Algorithm for the Examination Timetabling Problem. In *Practice and Theory of Automated Timetabling, Fourth International Conference*, E.K. Burke and P. De Causmaecker, Eds. Gent, Belgium: Lecture Notes in Computer Science 2740, Springer-Verlag, 2003, pp:207-231.

Timmis, Jonathan (2001) An Introduction to Artificial Immune Systems. *Research Notes*. ES2001, Cambridge, December 2001. University of Kent at Canterbury, England, UK.