FINAL EXAM TIMETABLING: A PRACTICAL APPROACH

Tony Wong[1], Pascal Côté[1], Paul Gely[2]
[1]*Department of automated production engineering,* [2]*Dean of resource management*
École de technologie supérieure, Université du Québec, 1100 rue Notre-Dame Ouest, Montréal (Québec), Canada H3C 1K3
emails: {tony.wong, pascal.cote, paul.gely}@etsmtl.ca

Abstract

This paper details a final exam timetable generator. This generator make use of a genetic algorithm optimizer and has the ability to generate several quality timetables for analysis and selection. This generator has been in use at the École de technologie supérieure with encouraging results. In all exam timetables, there are no exam conflicts, no student having to take 3 consecutive exams and the number of exams per period are always satisfied. There are on average 0.8% of the student body with 2 consecutive exams and about 11% of the student body have 1 pair of exams separated by 1 or 2 free periods. All other students have strictly more than 2 free periods between exams.

***Keywords**: Exam timetabling; Genetic algorithm; Constrained optimization*

1. INTRODUCTION

The final exam timetabling problem requires the assignment of a given number of exams within a fixed amount of time [1]. Our examination timetabling problem has the following characteristics: *i*) there is only one exam for each course; *ii*) no student should have two exams scheduled at the same time slot;. *iii*) exams are spread out temporally to give students more time to study. Therefore we should minimize the number of students with consecutive exams; *iv*) an exam should be assigned to the same time slot as the class schedule. This is helpful to most coop and part-time students since they have to plan their work schedule beforehand with their employers; *v*) due to the finite number of classrooms and other concurrent campus activities, there is a maximum number of exams per time slot; *vi*) some exams need to be preassign to a time slot based on laboratory equipment availability and other logistical constraints; vii) some exams can only be assigned to a subset of time slots because of laboratory equipment availability and other logistical constraints. The timetable generator has to take into account all these characteristics.

This paper is organized as follow. Section 2 presents the final exam timetabling problem model. Section 3 details the genetic algorithm optimizer used in this work. Section 4 presents the mapping of the timetabling problem into a constrained genetic algorithm optimization problem. Some practical results are shown in section 5. We conclude this paper in section 6.

2. PROBLEM MODEL

We characterize our final exam timetabling problem (FETP) as follow. There are q courses $c_1, c_2, \ldots, c_q$ with one exam for each course c_i. These exams are partitioned into r exam-groups $G_1, G_2, \ldots, G_r$ such that for each G_i there are students that take all exams belonging to group G_i.

Every school day of the semester has 3 class-periods. They correspond to the morning, afternoon and night classes. In every student class schedule, each course c_i is given a value such that $c_i \in \{0, 1, 2\}$. This value identifies the membership of c_i to one of the 3 class-periods of a school day. For the exam schedule, there are d days of exams. Thus we have a total of $3d$ exam-periods available for exam assignment.

To take into account the availability of exam-periods and the possibility of preassignment, we define an availability matrix $\mathbf{A}_{q\times 3d}$ such that $a_{i,k} = 1$ if period k is available for exam i and $a_{i,k} = 0$ otherwise. Using a similar logic, we define a preassignment matrix $\mathbf{B}_{q\times 3d}$ such that $b_{i,k} = 1$ if exam i is preassigned to period k and $b_{i,k} = 0$ otherwise. There exists only one preassigned period per exam, that is $\sum_{k=1}^{3d} b_{i,k} = 1$, i = 1, 2, ..., q. Finally, we denote l_k as the maximum number of exam for period k.

Given the above problem setting, our goal is to find a timetable $s_{i,k}$, i = 1, 2, ..., q and k = 1, 2, ..., $3d$ that minimizes

$$H = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3 \tag{1}$$

where α_1, α_2, α_3 are weighting factors and

$$h_1 = \sum_{k=1}^{d} \sum_{l=1}^{r} \sum_{i,j,m \in G_l} \left(s_{i,3k-3} + s_{j,3k-2} + s_{m,3k-1} \right), \tag{2.1}$$

$$h_2 = \sum_{k=1}^{d} \sum_{l=1}^{r} \sum_{i,j \in G_l} s_{i,3k-1} s_{j,3k}, \tag{2.2}$$

$$h_3 = \sum_{i=1}^{q} \sum_{k=1}^{d} \sum_{j=0}^{2} s_{i,3k-j} \left| j - c_i \right|, \tag{2.3}$$

subject to the following constraints:

$$\sum_{k=1}^{3d} s_{i,k} = 1, \quad i = 1,2,\ldots,q, \tag{3.1}$$

$$s_{i,k} \in \{0,1\}, \quad i = 1,2,\ldots,q \quad k = 1,2,\ldots,3d, \tag{3.2}$$

$$\sum_{i \in G_j} s_{i,k} \le 1, \quad j = 1,2,\ldots r \quad k = 1,2,\ldots,3d, \tag{3.3}$$

$$\sum_{i=1}^{q} s_{i,k} \le l_k, \quad k = 1,2,\ldots,3d, \tag{3.4}$$

$$\sum_{i \in G_l} s_{i,k} \le a_{i,k}, \quad l = 1,2,\ldots r \quad k = 1,2,\ldots,3d, \tag{3.5}$$

$$s_{i,k} \ge b_{i,k}, \quad i = 1,2,\ldots,q \quad k = 1,2,\ldots,3d. \tag{3.6}$$

The objective function (2.1) counts the number of three consecutive exams in day for all groups of exams. The objective function (2.2) returns the number of consecutive night and next-day morning exams. The minimization of (2.1) and (2.2) will help spread out the exams across the schedule. The objective function (2.3) counts the number of mismatches between a course's class-period and the assigned exam-period. As pointed out earlier, this is important for most coop and part-time students.

As for the constraints, equation (3.1) states that each exam must be assigned to one et only one exam-period. While (3.2) indicates that $s_{i,k}$ = 1 if exam i is assigned to period k, otherwise $s_{i,k}$ = 0. The constraint (3.3) stipulates that every exam in a given exam-group must not be assigned to a same period. This, in effect, forbides any exam conflicts (two or more exams of the same exam-group assigned to the same period). Finally, inequation (3.4) limits the number of exams per period to a maximum of l exams. The constraints (3.5) and (3.6) force the assignments to satisfy the availability matrix and the preassignment matrix respectively.

We do not attempt to solve this FETP directly because this timetabling problem is known to be NP-complete [2]. Rather, we transform our FETP into a constrained genetic algorithm optimization problem and use a genetic algorithm to find an optimized timetable. The genetic algorithm model and the problem mapping procedure are the subject of the next sections.

3. GENETIC ALGORITHM MODEL

Genetic algorithms (GA) are stochastic optimization methods using the so-called genetic operators [3]. They belong to the family of population-based evolutionary strategies [4]. In our context, the population is a set of candidate solutions to the optimization problem. Equation (4) describes the operating principal of an genetic algorithm. We represent a genetic operator using the form $O_{\underline{\xi/\rho}}$, where O is some genetic operator, ξ represent the operator's application scheme and ρ is the controlling parameter of the scheme. Note that some genetic operators do not have controlling parameters.

Utilizing the above genetic operator representation, the operating principal of an genetic algorithm is simply

$$\mathbf{X}_{g+1} = I_{\underline{\Omega}}\left(M_{\underline{\Pi/p_m}}\left(C_{\underline{\Lambda/p_c}}\left(S_{\underline{\Gamma}}\left(F_{\underline{\Phi}}\left(\mathbf{X}_g \right)\right)\right)\right)\right) \tag{4}$$

where $\mathbf{X}_g$ is the population at iteration g, F is the fitness attribution operator, S is the candidate solutions selection operator, C is the selected solutions crossover operator, M is the new candidate solutions mutation operator and finally, I is the new candidate solutions reinsertion operator. We can consider equation (4) as the successive transformation of a set of candidate solutions (i.e. population $\mathbf{X}$) by a set of genetic operators. The end results of (4) is a new set of candidate solutions with better approximation to the solutions of the optimization problem.

A genetic algorithm iterates until it satisfies a stopping criterion or until it reaches a maximum number of generations. For our timetabling problem, we use the latter strategy so we rewrite (4) to give

$$\mathbf{X}_{g+1} = I_{\underline{\Omega}}\left(M_{\underline{\Pi/p_m}}\left(C_{\underline{\Lambda/p_c}}\left(S_{\underline{\Gamma}}\left(F_{\underline{\Phi}}\left(\mathbf{X}_g \right)\right)\right)\right)\right), \quad g = 0,1,\ldots,g_{\max}. \tag{5}$$

While a genetic algorithm iterates from one iteration to another, the best solution so far is memorized. After $g_{\max}$ generations, the best solution is the best of all solutions found during the execution of the algorithm. Table 1 summarizes the genetic operators, their application scheme and controlling parameters used in our timetabling problem.

Table 1. Genetic operators used in this work.

Fitness attribution operator $F_{\overline{\Phi}}$	
Φ	direct evaluation.
Selection operator $S_{\overline{\Gamma}}$	
Γ	binary tournament [6].
Crossover operator $C_{\overline{\Lambda/p_c}}$	
Λ	uniform crossover [3].
p_c	crossover probability [0, 1].
Mutation operator $M_{\overline{\Pi/p_m}}$	
Π	random mutation with heuristic repair.
p_m	mutation probability [0, 1].
Insertion operator $I_{\overline{\Omega}}$	
Ω	total reinsertion.

4. PROBLEM MAPPING

This section present the mapping of the FETP into a constrained GA optimization problem. The first step is to define the proper candidate solution representation. Recall that GA uses a population of candidate solutions. The other steps of the problem mapping procedure are: *i)* define the fitness calculation for a candidate solution; *ii)* decide on the crossover and mutation rates; *iii)* define the heuristic repair procedure. The following subsections detail these steps and other genetic operators used in this work.

4.1 Candidate solution representation

The natural candidate solution representation is a $1 \times q$ vector [5]. Each element of the vector represent a course with exam. The vector element's value represent the exam-period assigned to that exam. For a population of candidate solutions we have a population matrix defined by

$$\mathbf{X} = \left[x_{i,j}\right], \quad i = 1,2,\ldots,N \quad j = 1,2,\ldots,q \qquad (6)$$

where q is the number of courses with exam and N the number of candidate solutions in the population. The allowable value for each exam is equal to corresponding preassignment matrix **B** element if the exam posesses a preassign period. Otherwise the allowable values is equal to the availability matrix **A** elements. Thus, the allowable values of every candidate solutions are,

$$x_{i,j} = k \quad \text{if } b_{j,k} > 0, \quad k = 1,2,\ldots,3d \qquad (7.1)$$

or

$$x_{i,j} \in \left\{k \mid a_{j,k} > 0, k = 1,2,\ldots,3d\right\}. \qquad (7.2)$$

This encoding ensures that all candidate solutions always satisfy constraints (3.1), (3.5) and (3.6). Since an exam-period is an integer number, we identify our GA as one that operates on a discreet integer-coded population.

4.2 Fitness evaluation

We define a fitness measure in order to evaluate the quality of the candidate solutions. This fitness measure make use of the normalized objective functions. Therefore, each candidate solution $\mathbf{x}_i = [x_{i,j}]$ has a fitness value f_i determined by

$$f_i = \frac{1}{1+\left(\sum_{j=1}^{3} \alpha_j \tilde{h}_j(x_i) + \delta v_1(\mathbf{x}_i) + \gamma v_2(\mathbf{x}_i)\right)}. \qquad (8)$$

The functions $\tilde{h}_1(\cdot)$, $\tilde{h}_2(\cdot)$ and $\tilde{h}_3(\cdot)$ are analogous to the objective functions (2.1) – (2.3) except that they operate on a candidate solution $\mathbf{x}_i$ instead. The weighting factors α_j give the relative importance to different objectives while δ and γ play the role of penalty constants. The $v_1(\cdot)$ and $v_2(\cdot)$ functions counts the number of violations of constraint (3.3) and (3.4) respectively.

As explained in subsection 4.4, exam conflicts may arise after a candidate solution mutation. Also, the candidate solution representation does not eliminate violations of (3.3) and (3.4). It is clear that $f_i \in \,]0, 1]$ with $f_i = 1$ indicating all objective functions are minimized and no exam conflict occurs in $\mathbf{x}_i$. Every candidate solution in a population must be evaluated using (8). This evaluation take place after the production of a new population and is the most time-consuming part of the whole GA optimization process.

4.3 Reproduction process

In order to achieve better approximation to the optimal solutions, a genetic algorithm optimizer needs to produce new candidate solutions from one iteration to another. The production of new candidate solutions relies on the selection, in the current population, of the fittest candidate solutions and share some of their elements. The whole reproduction process involves a selection operator and a crossover operator.

In this work, binary tournament [6] is the application scheme for the selection of the fittest candidate solutions. This scheme randomly select two candidate solutions from the current population and compare their fitness value. The candidate solution with the greatest fitness value wins the tournament. In order to apply the crossover operator, we need to execute the binary tournament twice to obtain two selected candidate solutions.

The uniform crossover operator takes the selected candidate solutions and share their element values with probability p_c. More precisely, the uniform crossover produces a new candidate solution $\mathbf{z} = [z_k]$ from parent solutions $\mathbf{x}_i$ and $\mathbf{x}_j$ according to

$$z_k = \begin{cases} x_{i,k} & \tau \leq p_c \\ x_{j,k} & \text{otherwise,} \end{cases} \quad k = 1,2,\ldots,q \qquad (9)$$

where τ is uniformly distributed random number and $f_i \geq f_j$. Thus, the new candidate solution $\mathbf{z}$ will have elements taken from $\mathbf{x}_i$ and $\mathbf{x}_j$.

4.4 Mutation and repair

The goal of the mutation operator is to provide some measure of diversification within a population [3]. It is a random perturbation that may help the optimizer to reach beyond the local minima. After the application of (9), a new candidate solution $\mathbf{z}$ is produced. We apply the random mutation to $\mathbf{z}$ as follow: *i)* selection a random number $\tau_1 \in [0, 1]$. If $\tau_1 < p_m$ where p_m is the mutation probability, continue to next step *ii.* Otherwise do not apply random mutation to $\mathbf{z}$. *ii)* select a random number τ_2 between 1 to q where q is the number of course with exam; *iii)* select randomly a allowable period for $z_{\tau 2}$. This random perturbation sometimes causes the candidate solutions to have exam conflicts (violation of constraint (3.3)).

In order to correct this anomaly, we scan the mutated candidate solution for any exam conflict. If there exists exam conflicts in more than one exam-group, we choose the exam-group with largest cardinality and mutate the conflicting exam in that group. If the heuristic repair produces a repaired candidate solution that is worst than the original mutated candidate solution, we discard the repaired candidate solution and vice versa.

4.5 Reinsertion

The reproduction and mutation – repair processes continue until all N new candidate solutions are produced. The new candidate solutions form a new population for the GA optimizer. That is, the new population totally replaces the current population and the algorithm is ready to execute another iteration.

4.5 Parameters calibration

In order to apply the GA optimizer to solve the FETP, it is necessary to determine the controlling parameters' value. These parameters are: *i)* N, number of candidate solutions; ii) $g_{\max}$, maximum number of iterations; *iii)* p_c, crossover probability; *iv)* p_m, mutation probability.

There is no analytical relationship between the performance and the population size N of a GA optimizer [7]. However, we do know the effects of the population size on the computation time of the algorithm: the larger is N the greater is the computation load.

Thus, we need to find a balance between the coverage of the search space and the time it takes to complete $g_{\max}$ iterations. In other to determine N and $g_{\max}$, we conducted a number of trial runs using a subset of the FETP with different N and $g_{\max}$ and measured their average value for the best trial-runs. These values are then linearly extrapolated to the size of the true FETP.

We used the same trial-runs to determine the crossover rate and mutation rate. For the crossover rate, we observed that the total number of conflicts decrease more quickly with high crossover rate. However, the standard deviation of the number of conflicts also stabilizes sooner. It means that the population has converged earlier than a lower crossover rate. This can be explained by the way the uniform crossover operates (see equation (9)).

A high crossover rate means that the new candidate solution inherits more and more elements from the best of its two parents. So, in the long run, the population has more and more candidate solutions that are descendants of a few parents. This may lead to the so-called premature convergence problem [6]. From the results shown in figure 1, we should restrict the crossover rate in between 0,95 and 0,75.

We applied similar reasoning in the determination of the mutation rate. In figure 2, no practical effects can be observed when the mutation rate is too low. The total number of conflicts does not change when $p_m = 0.01$. However, noticeable decrease of the number of conflicts occured when $p_m = 0.01$ and $p_m = 0.1$.

5. RESULTS

This section presents the results obtained by the GA optimizer. The FETP data are actual data from École de technologie supérieure. ETS is a mid-size engineering school in Montréal. It has 4 engineering departments, 16 undergraduate and graduate programs and over 3400 students. Table 2 summarizes the parameter setting for all timetable generation. All the timetabling constraints can easily be expressed by the use of a graphical interface. The following figures show some of these settings. For example, in figure 3, the preassigned period or available periods of a given exam are selected.

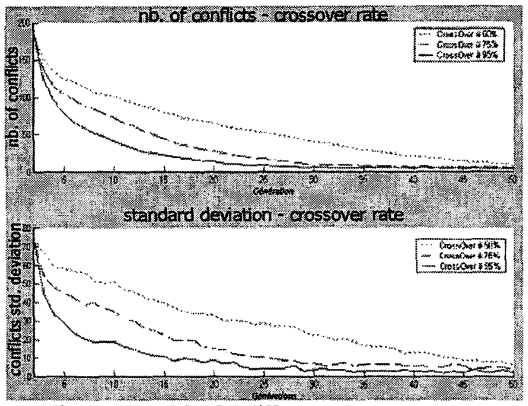


Figure 1. Results for different crossover rates.

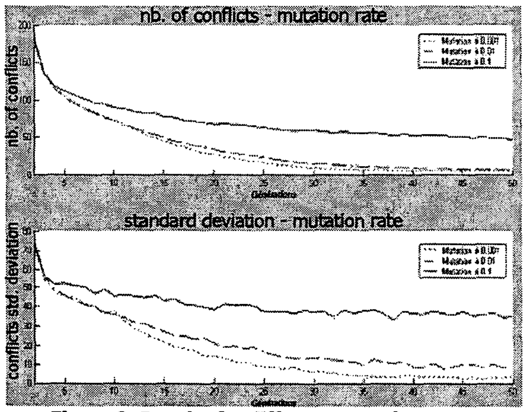


Figure 2. Results for different mutation rates.

Table 3 shows some typical results obtained using the timetable generator. They are very encouraging. In all exam timetables, there are no exam conflicts, no student having to take 3 consecutive exams and the number of exams per period are always satisfied. There are on average 28 students having to take 2 consecutive exams which is about 0.8% of the student body. We also measured the spread of a given timetable by counting the number of course entries that has 1 and 2 free periods between exams. On average there are 370 students (about 11% of the student body) that have 1 pair of exams separated by 1 or 2 free periods. All other students have strictly more than 2 free periods between exams. Finally, the average computation time is 7 hours on a 733 MHz desktop PC for a FETP with 8800 course entries, 173 courses, 32 exam-periods, 150 candidate solutions and 2000 iterations for the GA optimizer.

Table 2. Parameters setting for timetable generation.

Parameter	Value
Course entries†	8800
Courses with exam	173
Exam-period	32
$\alpha_1, \alpha_2, \alpha_3$	0.1, 0.05, 0.01
δ, γ	10, 0.05
Population size N	150
Maximum iteration $g_{\max}$	2000
Crossover rate p_c	0.85
Mutation rate p_m	0.01

†Each student can take multiple courses.

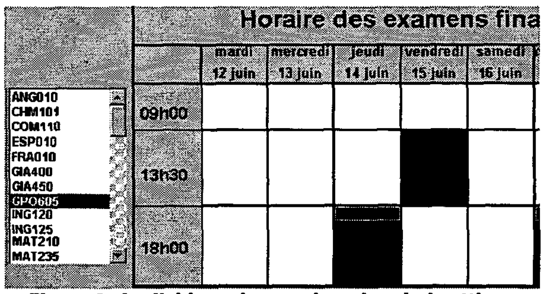


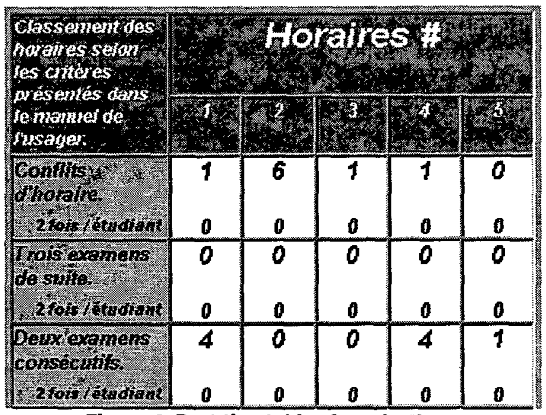
Figure 3. Available and preassigned period setting.

Because GA optimizers are population-based, there may exists more than one timetable that give the same fitness value. Therefore, we always retain the 5 best timetables of each GA run. These timetables are then presented for final selection as shown in figure 4.

It is also interesting to study the quality of a given timetable when one or more exams are assigned to other periods. This can be done without restarting the GA optimizer since it merely reassign a subset of exams of the selected candidate solution. Thus, it only involves the recalculation of the finesse value of the candidate solution. The graphical interface of the impact study is shown in figure 5.

Table 3. Typical results obtained.

Objective - constraint	Nb. students with conflict
3 consecutive exam / day	0
2 consecutive exam / day	28
2 or more exams / period	0
nb. exams / period $k > l_k$	0
Secondary criteria	Nb. of students
1 free period between exams	370
2 free periods between exams	
Other measure	
Computation time	≈ 7h (733 MHz PC)



Classement des horaires selon les critères présentés dans le manuel de l'usager.	Horaires #				
	1	2	3	4	5
Conflits d'horaire.	1	6	1	1	0
2 fois / étudiant	0	0	0	0	0
Trois examens de suite.	0	0	0	0	0
2 fois / étudiant	0	0	0	0	0
Deux examens consécutifs.	4	0	0	4	1
2 fois / étudiant	0	0	0	0	0

Figure 4. Best timetables for selection.

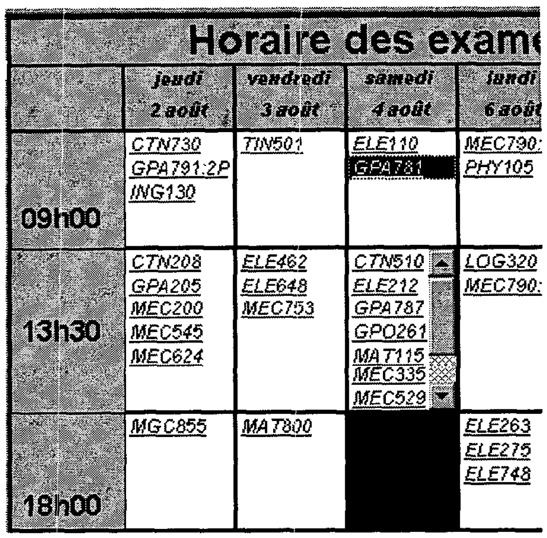
6. CONCLUSION

This paper presented a practical final exam timetable generator using a GA optimizer. Its algorithmic implementation is simple and does not involve any special heuristics. This timetable generator has been in used at École de technologie supérieure since 2001 with encouraging results.

Further studies are already being carried out especially in the area of multiobjective GA optimization. Since the FETP is in fact a multiobjective optimization problem, an multiobjective evolutionary algorithm is a more natural method for this problem.

Acknowledgements

The authors wish to thank Mr. Guy Surprenant, director of SIT (*Services informatiques et télé-communciation*) for granting us access to the ETS student database and Mr. Marcel Bourget for assisting us with the technical details of the student database. The authors also wish to thank the *Décanat de la gestion des ressources* of ETS for their support during this project.



Horaire des exam

	jeudi 2 août	vendredi 3 août	samedi 4 août	lundi 6 août
09h00	CTN730 GPA791:2P ING130	TIN501	ELE110 GPA781	MEC790: PHY105
13h30	CTN208 GPA205 MEC200 MEC545 MEC624	ELE462 ELE648 MEC753	CTN510 ELE212 GPA787 GPO261 MAT115 MEC335 MEC529	LOG320 MEC790:
18h00	MGC855	MAT800		ELE263 ELE275 ELE748

Figure 5. Impact studies of a timetable.

References

[1] D. Johnson, "Timetabling university examinations," *Journal of the Operational Research Society*, vol. 1, no. 41, pp. 39-47, 1990.

[2] M. R. Garey, D. S. Johnson, *Computer and Intractability – A Guide to NP-completeness*. San Francisco: W.H. Freeman and Company, 1979.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley, 1989.

[4] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information Systems*, vol. 1, no.3, pp. 269-308, 1999.

[5] D. Corne, H. L. Fang, C. Mellish, "Evolutionary timetabling: practice, prospects and work in progress," *UK Planning and Scheduling SIG Workshop*, 1994.

[6] D. E. Goldberg, K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in G. Rawlins, ed., *Foundations of Genetic Algorithms*, Morgan Kaufmann, 1991.

[7] A. E. Eiben, R. Hinterding, Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124-141, july 1999.