Solving an Uncapacitated Exam Timetabling Problem Instance using a Hybrid NSGA-II

Nuno Leite[1]*, Rui Neves[2], Nuno Horta[2], Fernando Melício[1] and Agostinho Rosa[3]†
[1]*ISEL - Lisbon Polytechnic Institute, Rua Conselheiro Emídio Navarro 1, 1959-007 Lisboa, Portugal*
[2]*Telecommunications Institute/IST, TU-Lisbon, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal*
[3]*LaSEEB-System and Robotics Institute, Department of Bioengineering/IST, TU-Lisbon, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal*

Keywords: Exam Timetabling Problem, Evolutionary Algorithms, Multi-objective Optimization, Combinatorial Problems.

Abstract: This paper describes the construction of an university examination timetable using a hybrid multi-objective evolutionary algorithm. The problem instance that is considered is the timetable of the Electrical, Telecommunications and Computer Department at the Lisbon Polytechnic Institute, which comprises three bachelor degree programs and two master degree programs, having about 80 courses offered and 1200 students enrolled. The task of manually construct the exam timetable for this instance is a complex one due essentially to the high number of combined degree courses. This manual process takes, considering a two-person team, about one week long. A hybrid multi-objective evolutionary algorithm, based on the Non-dominated Sorting Genetic Algorithm-II (NSGA-II), is proposed for solving this problem instance, incorporating two distinct objectives: one concerning the minimization of the number of occurrences of students having to take exams in consecutive days, and a second one concerning the minimization of the timetable length. The computational results show that the automatic algorithm achieves better results compared to the manual solution, and in negligible time.

1 INTRODUCTION

The construction of school and university examination timetables is one of the most important tasks taking place in educational institutions. Many institutions still elaborate their timetables in a manual form, involving a great deal of time and human resources and leading to suboptimal solutions. The task of automatically constructing examination timetables is known as the Exam Timetabling Problem (ETTP), and is an extensive studied optimization problem. The basic problem consists in distributing a set of exams by temporal periods, satisfying a set of hard and second order (or soft) constraints. Constraints of the first type cannot be violated as this results in an infeasible timetable. Constraints of the second type represent institution's view of what makes a good timetable and should be satisfied as many as possible. Examples of constraints include: not scheduling exams with common students in the same period (hard constraint); sufficient seating capacity for all exams (hard constraint); leave at least a two day interval between exams for all students (soft constraint). Moreover, depending if seating capacity hard constraint is considered or not, the ETTP is further classified in Capacitated ETTP and Uncapacitated ETTP, respectively. The actual program curricula seen at universities are designed to offer a great degree of diversity and flexibility to the students, letting them choose a considerable number of free or optional courses. In order to make this possible with available teacher, faculty staff, and university resources (rooms, equipment, etc.), courses are being offered in multiple related programs. This growing number of combined courses imposes extra difficulties in solving the ETTP.

The development of systems to automate the construction of university examination timetables has begun in the 1960 decade. The paper (Qu et al., 2009) constitute a survey of the recent (from 1995 to 2008) techniques and algorithmic approaches used to solve this problem. These techniques are clas-

*Work partially supported by the FCT SFRH/PROTEC/67953/2010 grant.
†Work partially supported by the FCT SFRH/BSAB/1101/2010 and PEst-OE/EEI/LA0009/2011 grants.

Leite N., Neves R., Horta N., Melício F. and Rosa A..
Solving an Uncapacitated Exam Timetabling Problem Instance using a Hybrid NSGA-II.
DOI: 10.5220/0004166001060115
In *Proceedings of the 4th International Joint Conference on Computational Intelligence* (ECTA-2012), pages 106-115
ISBN: 978-989-8565-33-4

sified in the following groups: Graph based sequential techniques, Constraint based, Local search based (e.g. Tabu search, Simulated Annealing), Population based (e.g. Evolutionary algorithms, Ant algorithms), Multi-criteria techniques, Hyper-heuristics and Decomposition/clustering techniques. More recently, the ETTP has been approached like a Multi-objective/Multi-criteria Optimization problem, acknowledging the true dimensions of real world problems, that typically have many facets to consider (proximity costs between student exams, timetable lengths, room assignment, invigilator availability, etc.). Multi-criteria techniques were proposed in (Burke et al., 2001) and (Petrovic and Bykov, 2003). Other recent works (Côté et al., 2004), (Wong et al., 2004), (Cheong et al., 2009) and (Mumford, 2010), applied Multi-Objective Evolutionary Algorithms (MOEAs) to solve the ETTP. Evolutionary approaches are well suited to solve Multi-objective Optimization (MOO) problems because a population of solutions is already being manipulated in each iteration of the evolutionary algorithm. Therefore, the population-approach of evolutionary algorithms can be effectively used to find the multiple trade-off solutions of MOO problems. In MOO the solutions are characterized by optimum sets of alternative non-dominated solutions, known as *Pareto sets*. Several MOEA have been proposed in the literature (Deb, 2001). It is known that metaheuristics, like evolutionary algorithms, work better if hybridized with other techniques (Raidl, 2006). In fact, the most successful applications of MOEA to the ETTP are hybrid approaches, being usually hybridized with some form of Local Search procedures. Moscato and Norman (Moscato and Norman, 1992) introduced the term *memetic algorithm* to describe evolutionary algorithms in which local search is used. Following this stream several authors developed hybridizations of MOEA with other metaheuristics (Ehrgott and Gandibleux, 2008). In (Burke and Landa Silva, 2005) the authors present design guidelines of memetic algorithms for scheduling and timetabling problems.

In this work we propose a novel hybrid MOEA and show its application on a real world ETTP instance. The considered problem instance is the examination timetable of the Electrical, Telecommunications and Computer Department (DEETC) at the Lisbon Polytechnic Institute. The proposed MOEA is based on the Elitist Non-dominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al., 2002). The NSGA-II procedure is one of the popularly used MOEA which attempt to find multiple Pareto-optimal solutions in a multi-objective optimization problem. Like the works (Côté et al., 2004), (Wong et al., 2004) (Cheong et al., 2009) and (Mumford, 2010), we also consider two objectives: one that maximizes each student free time between exams, and a second objective that considers the minimization of the timetable length.

Table 1: Characteristics of the DEETC dataset.

Exams	Students	Enrolment	Periods
80	1238	4637	18

Table 2: Number of exams per program in the winter semester of DEETC department.

LEETC	LEIC	LERCM	MEIC	MEET
32	30	29	19	25

The paper is organized as follows: the next section describes the DEETC department ETTP instance and its formulation as a multi-objective optimization problem. Section 3 presents the algorithmic flow of the proposed MOEA. Section 4 presents simulation results and analysis of the proposed algorithm. Finally, conclusions and future work are presented in Section 5.

2 PROBLEM DESCRIPTION

The problem instance considered in this work is the DEETC timetable of the winter semester of the last academic year. The DEETC timetable comprises five programs: three B.Sc. programs (named LEETC, LEIC and LERCM) and two M.Sc. programs (named MEIC and MEET). B.Sc. and M.Sc. programs have six and four semesters duration, respectively. The DEETC dataset characteristics are listed in Table 1.

The number of exams per program is listed in Table 2. About 34 of the 80 courses lectured in DEETC are shared by different programs, as depicted in Table 3. The high complexity of the timetable is due mainly to two reasons: (1) high degree of course sharing in different programs and different semesters (e.g. LSD course is offered in the 1st and 2nd semesters of LEIC and LEETC programs, respectively); (2) the courses of the even semesters (summer semesters) are also being lectured in the winter semester, thus increasing the timetable complexity, because there are students attending courses in the even and odd semesters. To get an idea of the number of students involved in each semester, we present in Table 4 the number of classes proposed for the winter semester for each program. Each class of the 1st to the 3rd semester has on average 30 students and the remainder semesters have 20 students per class on average.

Table 3: Courses shared among the five programs offered in the DEETC. The number of shared courses sums to 34 (out of 80 courses with exam). The first five columns contain the semesters where the course is offered. Semesters in M.Sc. courses are numbered 7 to 10 (four semester master program).

MEIC	MEET	LERCM	LEIC	LEETC	Course	Acronym
		1	1	1	Linear Algebra	ALGA
		1		1	Mathematical Analysis I	AM1
		1	1	1	Programming	Pg
			1	2	Logic and Digital Systems	LSD
		2		2	Mathematical Analysis II	AM2
		2	2	2	Object Oriented Programming	POO
		2	2	3	Probability and Statistics	PE
			2	3	Computer Architecture	ACp
			3 and 5		Computer Graphics	CG
			3 and 5		Computation and Logic	LC
			3 and 5		Functional Programming	PF
		3	3	4	Imperative Programming in C/C++	PICC/CPg
	7	3		5	Digital Communication Systems	SCDig
		4	4	4	Computer Networks	RCp
	7		4	5	Virtual Execution Systems	AVE
	8	4			Multimedia Signal Codification	CSM
		4		5	Operating Systems	SOt
7		5			Unsupervised Learning	AA
	8	5			Database Systems	BD
	8		5	6	Internet Programming	PI
	8	5		6	Distributed Computational Systems	SCDist
7	7	5	5	5	Internet Networks	RI
7			5		Compilers	Cpl
	7			5	Control	Ctrl
	7			5	Radio Communications	RCom
7			5		Security Informatics	SI
	7			5	Telecommunication Systems	ST
7	7		5	5	Embedded Systems I	SE1
7	7	6			Multim. Comm. Networks & Services	RSCM
7			6		Distributed Systems	SD
7		6			Software Engineering	ES
7 to 9	8	6	3 to 6	6	Project Management and Economics	EGP
7 to 9	8	6	3 to 6	6	Enterprise Management	OGE
7 to 9	8	6	3 to 6	6	Management Systems	SG

Table 4: Number of classes proposed for the winter semester for each program.

Sem.	LEIC	LEETC	LERCM	MEIC	MEET
1st	5	5	3	2	2
2nd	3	3	1	-	-
3rd	3	3	2	2	2
4th	2	2	1	-	-
5th	3	3	1		
6th	-	-	-		
Total	16	16	8	4	4

2.1 Problem Formulation

This paper considers an instance of the ETTP that was first formulated in (Burke et al., 1996). In their formulation, if a student is scheduled to take two exams in any one day there should be a free period between the two exams. Violation of this constraint is referred as a *clash*. In their work it is considered the Capacitated problem, whereas in our work we ignore the capacity constraint. The corresponding Uncapacitated problem is formulated as:

$$\text{Minimize} \quad f_1 = \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|-1} a_{ip}\, a_{j(p+1)}\, c_{ij} \quad (1)$$

$$f_2 = |P| \quad (2)$$

$$\text{subject to} \quad \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|} a_{ip}\, a_{jp}\, c_{ij} = 0, \quad (3)$$

$$\sum_{p=1}^{|P|} a_{ip} = 1, \forall i \in \{1, \ldots |E|\}, \quad (4)$$

where:

- $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is the set of exams to be scheduled,
- $P = \{1, 2, \ldots, |P|\}$ is the set of periods,
- a_{ip} is one if exam e_i is allocated to period p, zero otherwise.
- c_{ij} (Conflict matrix) is the number of students registered for exams e_i and e_j.

Eqs. (1) and (2) are the two objectives of minimizing the number of clashes and timetable length, respectively. Constraint (3) is the (hard) constraint that no student is to be scheduled to take two exams in the same period. Constraint (4) indicates that every exam can only be scheduled once in any timetable.

3 HYBRID MULTI-OBJECTIVE GENETIC ALGORITHM

As mentioned in the introduction, we solve the DEETC ETTP instance using a hybrid MOEA based on the NSGA-II algorithm. NSGA-II has the following features: (1) it uses an elitist principle, (2) it uses an explicit diversity preserving mechanism, and (3) it emphasizes non-dominated solutions. The basic NSGA-II was further transformed to include a step where a Local Search procedure is performed. The general steps of the hybrid algorithm (named HMOEA) are depicted in Figure 1. In the following subsections we describe each block of the HMOEA in detail.

Procedure HMOEA
$P^{(t)}$: parent population at iteration t
$Q^{(t)}$: offspring population at iteration t
$R^{(t)}$: combined population at iteration t
L: local search operator
OUTPUT
$N^{(t)}$: archive of non dominated timetables

Initialize P^0 and Q^0 of size N with random timetables
For each iteration $t \leftarrow 0, 1, \ldots, I_{max} - 1$ do
(**Step 1**) Form the combined population, $R^{(t)} = P^{(t)} \bigcup Q^{(t)}$, of size $2N$.
(**Step 2**) Classify $R^{(t)}$ into different non-domination classes.
(**Step 3**) If $t \geq 1$, use local search procedure L to improve elements of $R^{(t)}$.
(**Step 4**) Form the new population $P^{(t+1)}$ with solutions of different non-dominated fronts, sequentially, and use the *crowding sort* procedure to choose the solutions of the last front that can be accommodated.
(**Step 5**) $N^{(t+1)} \leftarrow NonDominated(P^{(t+1)})$.
If $t = I_{max} - 1$ then Stop.
(**Step 6**) Create offspring population $Q^{(t+1)}$ from $P^{(t+1)}$ by using the crowded tournament selection, crossover and mutation operators.
(**Step 7**) Repair infeasible timetables.

Figure 1: Hybrid NSGA-II procedure.

3.1 Chromosome Encoding

In order to optimize for the second objective (see Eq. (2)), each timetable is represented by a variable-length chromosome as proposed by (Cheong et al., 2009), and illustrated in Figure 2. A chromosome encodes a complete and feasible timetable, and contain the periods and exams scheduled in each period. Valid timetables should have a number of periods belonging to a valid interval, initially given by the timetable planner. However, the operation of crossover and mutation could produce invalid timetables, because of extra periods added to the timetable as a result of these operations. Thus, a repairing scheme must be applied in order to repair infeasible timetables. The adopted scheme is explained in detail in Section 3.4.

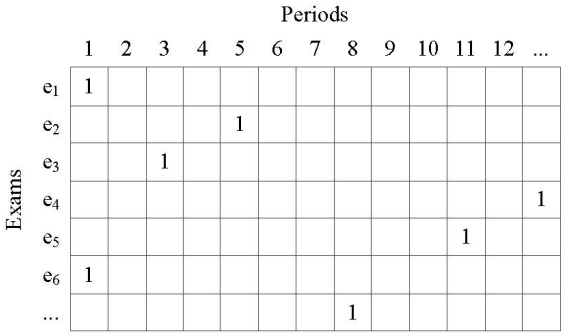

Figure 2: Variable length chromosome representation. A chromosome encodes a complete and feasible timetable.

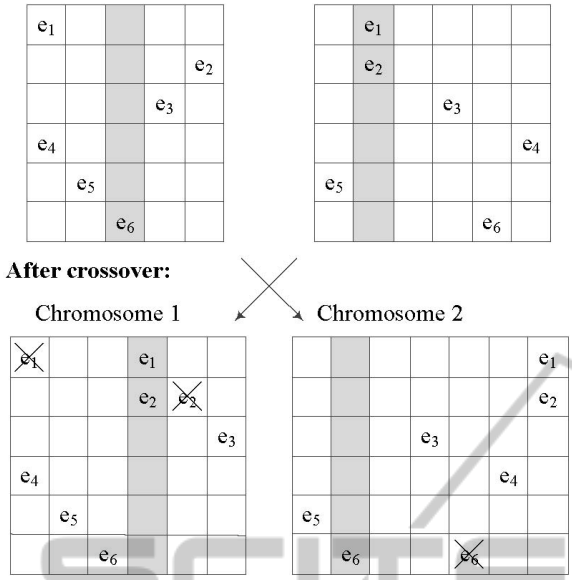

Figure 3: Illustration of Day-exchange crossover based on (Cheong et al., 2009). The shaded periods represent the chromosomes *best days*. These are exchanged between chromosomes, being inserted into randomly chosen periods. Duplicated exams are then removed.

3.2 Population Initialization

It is known that the basic examination timetabling, of minimizing the number of slots considering the hard constraint of not having students with overlapping exams, is equivalent to the graph colouring problem (Côté et al., 2004). As such, several heuristics of graph colouring have been applied to the ETTP. These heuristics influence the order in which exams are inserted in the timetable. In this work, we use the following two heuristics, in the initialization and mutation processes:

- Saturation Degree (SD): Exams with the fewest valid periods, in terms of satisfying the hard constraints, remaining in the timetable are reinserted first.
- Extended Saturation Degree (ESD): Exams with the fewest valid periods, in terms of satisfying both hard and soft constraints, remaining in the timetable are reinserted first.

The ESD heuristic is used in the population initialization procedure, while the SD heuristic is used in the reinsertion process of the mutation operator (detailed in Section 3.3). These two procedures are similar to the procedures applied in (Cheong et al., 2009). The use of the SD heuristic in the initialization process has been experimented but with worse results than the ESD heuristic.

Table 5: HMOEA parameters and environmental settings.

Parameter	Value
Population size	40
Number of iterations	I_{max} =125
Crossover probability	1.0
Mutation probability	0.2
Reinsertion rate	0.02
SHC no. iterations	$t_{max} = 5$
SHC temperature	T = 0.0001
Computer	Intel Core i7-2630QM, 2.0 GHz, 8 GB RAM
OS	Win 7.0 Pro, 64 bit
Matlab version	Version 7.9 (R2009b)

Table 6: Clashes per course for the manual and automatic solutions with 18 periods.

	Number of clashes	
Timetable	Manual sol.	Automatic sol.
LEETC	287	215
LEIC	197	163
LERCM	114	71
MEIC	33	23
MEET	50	40
Combined	**549**	**393**

In the initialization process, a timetable with a random (valid) length is generated for each chromosome. Then, the unscheduled exams are ordered according to the ESD heuristic and a candidate exam is selected randomly being then scheduled into a randomly chosen period (chosen from the set of periods with available capacity while respecting the feasibility constraint). If no such period exists, a new period is added to the end of the timetable to accommodate the exam. In the ESD heuristic used, a candidate exam can be scheduled in a period if it does not violate feasibility and if the number of clashes is bellow or equal to 70. This process is repeated until all exams have been scheduled.

3.3 Selection, Crossover and Mutation

The offspring population is created from the parent population by using the crowded tournament selection operator (Deb et al., 2002). This operator compares two solutions and returns as the winner of the tournament the one which as a better rank, or if the solutions have the same rank, the one who has a better crowding distance (the one which is more far apart from their direct neighbours).

The crossover and mutation operations were adapted from the ones introduced in (Cheong et al.,

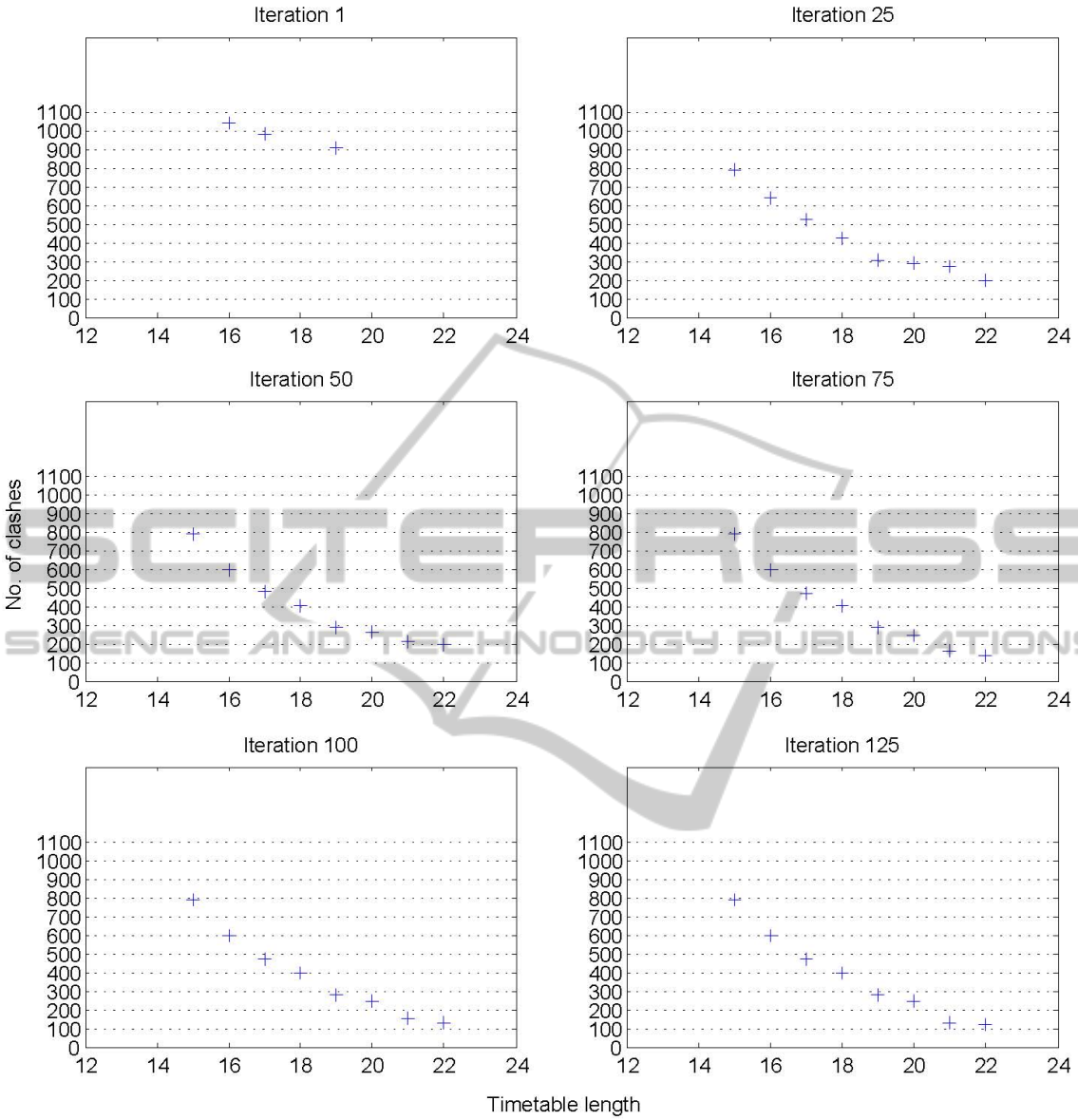

Figure 4: Evolution of the Pareto front.

2009). In the crossover operator, termed Day-exchange crossover, the best days, selected based on the crossover rate, are exchanged between chromosomes. The best day of a chromosome consist of the day (a period, in our case) which has the lowest number of clashes per student. This operation is illustrated in Figure 3. To ensure feasibility after the crossover operation, the duplicated exams are deleted. Notice that, as mentioned before, the result of inserting a new period in a chromosome could produce a timetable with a number of periods larger than the valid upper limit. If this is the case, a repair scheme is applied in order to compact the timetable.

The mutation operator removes a number of exams, selected based on the reinsertion rate, and reinserts them into other randomly selected periods while maintaining feasibility. We use the SD graph colouring heuristic to reorder the exams, prior to reinserting them. As in the case of the crossover operator, the mutation operator could also add extra periods to the timetable, for the exams that could not be rescheduled without violating the hard constraints.

3.4 Repairing Scheme

The repair scheme adopted is similar to the period control operator of (Cheong et al., 2009), consisting of the following two operations: (1) *Period expan-*

Table 7: Manual solution for the LEETC examination timetable. The courses marked in bold face are shared with other programs, as shown in Table 3. The number of clashes of this timetable is 287.

Sem.	Course	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1st	**ALGA**			x															
	Pg							x											
	AM1												x						
	FAE																	x	
	ACir														x				
2nd	**POO**								x										
	AM2											x							
	LSD																		x
	E1															x			
	MAT					x													
3rd	**PE**													x					
	ACp																	x	
	EA									x									
	E2	x																	
	SS						x												
4th	**RCp**											x							
	PICC/CPg															x			
	PR			x															
	FT							x											
	SEAD1																		x
5th	**ST**														x				
	RCom					x													
	RI										x								
	SE1													x					
	AVE								x										
	SCDig												x						
	SOt																	x	
6th	**PI**												x						
	SCDist															x			
	EGP	x																	
	OGE	x																	
	SG	x																	

sion, used when the timetable has a number of periods below the lower limit, and (2) *Period packing*, used when the timetable has a number of periods above the upper limit. In the period expansion operation, empty periods are first added to the end of the timetable such that the timetable length is equal to a random number within the period range. A *clash list*, comprising all exams involved in at least one clash, is maintained. Then, all the exams in the clash list are sweep in a random order and rescheduled into a random period without causing any clashes while maintaining feasibility. Exams which could not be moved are left intact. The period packing operation proceeds as follows: first, the period with the smallest number of students is selected; then the operation searches in order of available period capacity, starting from the smallest, for a period which can accommodate exams from the former while maintaining feasibility and without causing any clashes. The operation stops when the timetable length is reduced to a random number in the desired range or when it goes one cycle through all periods without rescheduling any exam.

3.5 Ranking Computation

The non-dominated sorting procedure used in NSGA-II use the evaluation of the two objective functions to rank the solutions. We adopt a simple penalization scheme in order to penalize solutions with an invalid number of periods. The penalization is enforced according to the following pseudo-code:

If timetable length $>$ max length **Then**

$$f_1^{Pen} = f_1 + \alpha_1(\text{timetable length} - \text{max length})$$

$$f_2^{Pen} = f_2 + \alpha_2(\text{timetable length} - \text{max length})$$

Else If timetable length $<$ min length **Then**

$$f_1^{Pen} = f_1 + \alpha_1(\text{min length} - \text{timetable length})$$

$$f_2^{Pen} = f_2 + \alpha_2(\text{min length} - \text{timetable length}).$$

We set $\alpha_1 = 1000$ and $\alpha_2 = 10$ to introduce a high penalization on the number of clashes and number of periods, respectively.

Table 8: Automatic solution for the LEETC examination timetable. The courses marked in bold face are shared with other programs, as shown in Table 3. The number of clashes of this timetable is 215.

Sem.	Course	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1st	**ALGA**												x						
	Pg	x																	
	AM1							x											
	FAE					x													
	ACir																	x	
2nd	**POO**		x																
	AM2				x														
	LSD										x								
	E1															x			
	MAT													x					
3rd	**PE**						x												
	ACp														x				
	EA								x										
	E2																		x
	SS																x		
4th	**RCp**				x														
	PICC/CPg															x			
	PR										x								
	FT	x																	
	SEAD1					x													
5th	**ST**							x											
	RCom												x						
	RI																	x	
	SE1						x												
	AVE		x																
	SCDig													x					
	SOt									x									
6th	**PI**							x											
	SCDist				x														
	EGP																		x
	OGE																		x
	SG																		x

3.6 Local Exploitation

The Local Exploitation step employs a Local Search procedure to improve locally some elements of the population. First, $2N/4$ groups of fours are formed by randomly selecting into each group elements of the population $R^{(t)}$. Then, tournaments between elements of each group are taken. The chromosome which has the lower rank (the one who belongs to the front with lower number) wins the tournament and is then scheduled for the improve step. With this procedure, about $N/2$ of the chromosomes are selected for improvement. Also, it is guaranteed that at least one element of the non-dominated front is selected for improvement. The selected chromosomes are improved locally using a short iteration Stochastic Hill Climber (SHC) procedure, with objective function $f_1 = \textit{minimization of the number of clashes}$. We set a low temperature T in the SHC. In this way, our SHC works like a standard Hill Climber but with only one neighbour, instead of evaluating a whole neighbourhood of solutions. The random neighbour is selected according to the following operation. Firstly, a clash list for the selected chromosome is built. Then, the neighbour chromosome is the one which results from applying the best move of an exam in the clash list into a feasible period. The best move is the one that leads to the highest decrease in the number of clashes.

4 COMPUTATIONAL RESULTS

In our experiments we applied the proposed HMOEA to the DEETC dataset specified in Section 2. Table 5 gives the algorithmic parameters and environmental settings used in the experiments. The algorithm was programmed in the Matlab language.

Firstly, we present the results on the performance of HMOEA and then compare it with the available manual solution. In the experiment made, the initial period range was set to the interval $[14,22]$, that is, four periods below and upper the number of periods

set in the manual solution. The performance of the HMOEA in terms of the evolution of the Pareto is illustrated in Figure 4. We can observe that the algorithm converges rapidly as in iteration 25 it has already a complete first front that is a good approximation of the final Pareto front. After that iteration, the individual solutions are further optimized but to a lesser extent. The running time for this experience (the best result out of five runs) was 296 seconds or $\approx$ 5 minutes.

In Table 6 we compare the number of clashes per course obtained by the manual and automatic (considering the obtained solution with 18 periods) procedures. As we can conclude from this table, the automatic solution improved the number of clashes in all the timetables, which corresponds to a lower number of clashes in the optimized merged timetable. Tables 7 and 8 present the timetables for the most difficult program: the LEETC program. We can see that, qualitatively, the timetable produced by the automatic procedure has a reasonable layout as the exams within the same semester are well distributed.

5 CONCLUSIONS

In this paper we solved a real instance of the exam timetabling problem using a hybrid multi-objective evolutionary algorithm. The instance considered comprises five programs with high degree of course sharing between programs, which difficult the manual construction of the timetable. In the manual elaboration of the timetable actually five timetables are optimized concurrently, one for each program. The automatic algorithm solves this instance by optimizing the combined timetable. With the application of the proposed hybrid MOEA, the present instance was solved effectively, with lower number clash conflicts compared to the manual solution and in negligible time. The current results were obtained without special fine tuning. Moreover, in experiences made, we obtained lower number of clashes than the actual results, but the optimization in each timetable was not balanced, as some timetables were more optimized than others. This is explained by the intrinsic difficulty in optimizing each timetable, e.g. the LEETC is more difficult to optimize than the the LERCM timetable, because it has a greater number of shared courses and more students registered on those courses.

5.1 Future Work

Several improvements could be made to the algorithm. Some are listed next:

- In order to prevent the algorithm to optimize in an unbalanced way, we could considering adding has an objective a measure of program balance, in order to guide the algorithm to prefer solutions were the number of clashes is minimized *and* the balance in programs is achieved.
- Consider room assignment, by solving the Capacitated ETTP.

Finally, in order to evaluate the performance of the HMOEA, we intend to run the algorithm in the set of ETTP benchmarks available - the Toronto and Nottingham benchmarks (Qu et al., 2009) - and compare with other approaches.

REFERENCES

Burke, E., Bykov, Y., and Petrovic, S. (2001). A multicriteria approach to examination timetabling. In Burke, E. and Erben, W., editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 118–131. Springer Berlin / Heidelberg.

Burke, E. and Landa Silva, J. (2005). The design of memetic algorithms for scheduling and timetabling problems. In Hart, W., Smith, J., and Krasnogor, N., editors, *Recent Advances in Memetic Algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*, pages 289–311. Springer Berlin / Heidelberg.

Burke, E. K., Newall, J. P., and Weare, R. F. (1996). A memetic algorithm for university exam timetabling. In (Eds.), E. K. B. . P. R., editor, *Lecture notes in computer science: Proceedings of the 1st international conference on the practice and theory of automated timetabling, PATAT 1995*, volume 1153, pages 241–250, Edinburgh, Scotland. Berlin: Springer.

Cheong, C., Tan, K., and Veeravalli, B. (2009). A multi-objective evolutionary algorithm for examination timetabling. *Journal of Scheduling*, 12:121–146.

Côté, P., Wong, T., and Sabourin, R. (2004). Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. In *eds): Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling (PATAT 2004)*, pages 151–167.

Deb, K. D. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester: Wiley.

Deb, K. D., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Ehrgott, M. and Gandibleux, X. (2008). Hybrid metaheuristics for multi-objective combinatorial optimization. In Blum, C., Aguilera, M., Roli, A., and Sampels, M., editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 221–259. Springer Berlin / Heidelberg.

Moscato, P. and Norman, M. G. (1992). A "memetic" approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, pages 177–186. IOS Press.

Mumford, C. (2010). A multiobjective framework for heavily constrained examination timetabling problems. *Annals of Operations Research*, 180:3–31.

Petrovic, S. and Bykov, Y. (2003). A multiobjective optimisation technique for exam timetabling based on trajectories. In Burke, E. and Causmaecker, P., editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 181–194. Springer Berlin Heidelberg.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., and Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12:55–89.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In *Hybrid Metaheuristics*, pages 1–12.

Wong, T., Côté, P., and Sabourin, R. (2004). A hybrid MOEA for the capacitated exam proximity problem. In *Congress on Evolutionary Computation, 2004. CEC2004.*, volume 2, pages 1495 – 1501 Vol.2.