A PSO algorithm to solve a Real Course+Exam Timetabling Problem(1)

Elizabeth Montero[1,2] (2), María-Cristina Riff[2], Leopoldo Altamirano[2]

[1] Université de Nice
2000 route des Lucioles, 06903 Sophia Antipolis Cédex, France
montero.ureta.elizabeth@etu.unice.fr

[2] Universidad Técnica Federico Santa María
Avenida España 1680, Valparaíso, Chile
{elizabeth.montero, maria-cristina.riff, leopoldo.altamirano}@inf.utfsm.cl

Abstract

University course and exams timetabling is a very well-known constrained problem. The problem becomes more difficult when we have to face a dynamic timetabling problem where new courses and exams can appear during the semester. In this paper, we propose a particle swarm approach to simultaneously solve both problems as well as a local search procedure to handle the dynamism. We show that our approach can find high quality solutions within minutes, where a traditional forward checking requires hours to obtain comparable ones.

Keywords

University course timetabling, dynamic exam timetabling, particle swarm optimization

1 Introduction

Dynamic course and exams timetabling is a well-known and very difficult constrained problem. The course timetabling and dynamic exam timetabling problems at the Chilean Technical University Federico Santa María (UTFSM) have been manually managed until now. Because of the growing number of educational programs and the increasing number of students this method has collapsed, and many dynamic requirements can not be satisfied. In this study we have identified some special features of our problem. At the beginning, to construct a course timetabling each department of the University communicates its courses requirements for the semester, giving the time-slots required. Our problem is, thus, an allocation problem where a classroom must be assigned to each course satisfying the associated facility and capacity constraints. Given the dimensionality of our course timetabling problem we can not ensure optimality in a reasonable time. Here, we consider optimality as either a planning that produces a minimum number of unused seats each week or a planning that generates the most equitable use of classrooms. For the first phase, that corresponds to the course timetabling problem we have designed a particle swarm optimization algorithm (PSO).

During the semester, the solution found for the course timetabling problem must be modified in order to satisfy new requirements as exams. Exams are not concentrated on one or two weeks, and no special classrooms are reserved for exams. Exams are individually planned by each lecturer according to his course requirements (date on which the exam will take place, classroom capacity, duration). Section 2 gives full details of our real world problem at UTFSM. When the time-slot of a given exam is fixed, the problem consists in finding a classroom with a capacity of at least twice the number of students in the corresponding course. At this point, most of the classrooms are usually assigned, and finding an empty one for the exam's time-slot with the required capacity is difficult. Therefore, perturbations must be made to move one or more courses from their originally assigned classroom to a new one. The new solution must be obtained as fast as possible, and obviously with the lowest number of perturbations. This phase is handled by a local search procedure. It is first applied to the static initial solution for the course timetabling problem obtained from the PSO algorithm, and thereafter to the current solution, each time a new exam is requested by a lecturer.

We explain our PSO algorithm and local search procedure in section 3. We compare our approach against two methods. We first show that our approach gives high quality solutions, as well as does forward checking, but in a dramatically smaller CPU time. Then we show that our proposition obtains much better results than those given by the manual method currently applied in our university. Section 4 gives the comparison results, and an analysis of the solutions quality and algorithm performance. We finally present an overview of existing approaches to solve this kind of problems in section 5, and some conclusions and future work in section 6.

[1]Partially supported by Centro Científico Tecnológico de Valparaíso (CCT-Val) FB0821.
[2]Partially supported by Project ECOS/CONICYT 2010-038.

2 Problem Statement at UTFSM

At the Technical University Federico Santa María we have two main problems:

2.1 Static Problem

At the beginning, we know all courses to be scheduled including their capacity and facility requirements (multimedia). A problem solution must satisfy the following hard constraints:

- Lectures having students in common can not take place at the same time-slot;
- Lectures must take place in a classroom suitable for them in terms of facilities and capacity;
- Two lectures of the same course scheduled in consecutive time-slots must be assigned to the same classroom;
- Two lectures can not take place at the same time-slot in the same classroom; and
- Each lecture must be assigned to a classroom in its corresponding time-slot;

A timetable in which all lectures have been assigned to a time-slot and a classroom, such that no hard constraints are violated, is said to be feasible. The aim of the problem is to find a feasible solution that do the best use of facilities. In our case, the *H1* constraint is satisfied by a preprocessing step accomplished by each University Department. The most difficult constraints to be satisfied are *H3*. These constraints mean that when a course is planned in two consecutive time-slots, they must be scheduled in the same classroom. The idea is neither the students nor the teacher moves to another classroom. For this problem we have studied two evaluation functions:

- *O1*: To minimize the number of unused seats in each classroom during the week
- *O2*: To balance the number of times that each classroom is used during the week

The idea of the evaluation function O_1 is to obtain a solution where the classrooms satisfy the capacity and facility constraints considering a minimum number of free seats. The second one tries to obtain a solution which allows a fairly use of the classrooms. This is important from the economical point of view, since it allows a distribution of the workload of each classroom, where projector lamp and other classrooms furniture are utilized equitably .

2.2 Dynamic Problem

During the semester new requirements must be satisfied, as exams and new courses, thus the solution found for the static problem must be perturbed. We distinguish two main situations:

- There is an available classroom that satisfy the constraints, so it can be assigned to this new activity.
- When a classroom is not available, some re-assignments are required in order to update the original course planning.

For this dynamic problem we consider as evaluation function the minimization of the number of moves to be done in order to satisfy the new requirements. This evaluation function is used in order to minimize the number of perturbations of the original planning (static solution).

2.3 Mathematical Model

Parameters

- Set of **lectures** $\mathcal{L} = \{l_1, l_2, \ldots, l_L\}$; where $|\mathcal{L}| = L$
- Set of **classrooms** $\mathcal{R} = \{r_1, r_2, \ldots, r_R\}$; where $|\mathcal{R}| = R$
- Set of **time-slots** $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$; where $|\mathcal{T}| = T$
- p_l: number of participants of lecture l, where $p_l \geq 0, \forall l \in L$
- c_r: capacity of classroom r, where $c_r \geq 0, \forall r \in R$
- ta_l: time-slot required by lecture l, where $1 \leq ta_l \leq T, \forall l \in L$
- $f_l = \begin{cases} 1 & \text{If lecture } l \text{ requires facilities} \\ 0 & \text{otherwise} \end{cases}$
- $e_r = \begin{cases} 1 & \text{If classroom } r \text{ posses facilities} \\ 0 & \text{otherwise} \end{cases}$
- $g_{l,m} = \begin{cases} 1 & \text{If lecture } l \text{ and } m \text{ are consecutive} \\ 0 & \text{otherwise} \end{cases}$

Decision variable

- $X_{l,r,ta_l} = \begin{cases} 1 & \text{If classroom } r \text{ is assigned to lecture } l \text{ in the required time-slot } ta_l \\ 0 & \text{otherwise} \end{cases}$

Constraints

- Each classroom posses the required capacity of the assigned lectures

$$X_{l,r,ta_l} \cdot p_l \leq c_r \; ; \qquad \forall l = 1 : L \; , \forall r = 1 : R \tag{1}$$

- Each classroom posses the required facilities of the assigned lectures

$$X_{l,r,ta_l} \cdot f_r \leq e_r \; ; \qquad \forall l = 1 : L, \forall r = 1 : R \tag{2}$$

- Two consecutive lectures must be assigned to the same classroom

$$X_{l,r,ta_l} + X_{m,r,ta_m} \geq 2 \cdot g_{l,m}; \qquad \forall l, m = 1 : L, l \neq m, \forall r = 1 : R \tag{3}$$

- Each classroom can be assigned only to one lecture each time-slot

$$\sum_{l=1}^{L} X_{l,r,ta_l} \leq 1; \qquad \forall r = 1 : R \tag{4}$$

- Each lecture is assigned to one room in its time-slot

$$\sum_{r=1}^{R} X_{l,r,ta_l} = 1; \qquad \forall l = 1 : L \tag{5}$$

Objectives For this problem we define two different evaluation functions according to the University requirements:

- *O1*: Minimize the total free seats. It measures the amount of seats unused during a week

$$\sum_{l=1}^{L} \sum_{r=1}^{R} X_{l,r,ta_l} \cdot (c_r - p_l) \tag{6}$$

- *O2*: Minimize the number of times the classrooms are used during a week

$$\frac{\sum_{r=1}^{R} | \sum_{l=1}^{L} X_{l,r,ta_l} - \bar{X} |}{R} \tag{7}$$

where $\bar{X} = \frac{\sum_{l=1}^{L} \sum_{r=1}^{R} X_{l,r,ta_l}}{R}$

3 The Two-phase Algorithm

To face this problem, we consider an algorithm with two phases: a static and a dynamic phase.

3.1 Static Approach - Particle Swarm algorithm

For the static problem we propose an algorithm based on particle swarm optimization. We have selected the particle swarm method because it is an easy to implement algorithm. Moreover, it incorporates few parameters and it has been able to get good quality solutions in timetabling problems.
For this problem, we have modified the basic approach in order to apply it in a discrete domain problem [5], [3]. The algorithm manages all the constraints and the moves are done on the discrete space.
In our approach, the position (X) and the velocity (V) of each particle are represented as matrices of size $R \times T$. The position uses a binary representation, where element X_{rt} represents if classroom r is being used in the time-slot t or not. Representation also includes the information of the lecture assigned to classroom r in time-slot t.
Velocity uses a real-valued representation, where V_{rt} represents the probability of change of the X_{rt} position element. Velocity values vary between 0 and V_{max}. The particle swarm algorithm utilizes a star topology.
The algorithm works with feasible solutions, which satisfies the capability, facility and consecutiveness constraints. It starts with a random population of feasible solutions. Algorithm 1 shows the procedure used to create the initial feasible particles.
Velocity matrix is randomly initialized with values in range $[0; 1]$.
Particles are evaluated according to one of the following evaluation criteria:

Algorithm 1 Feasible solutions construction procedure

```
Procedure $feasible\_solutions\_construction()$
repeat
    a random lecture $l_i$ is selected
    for each feasible classroom available $r$ in $l_i$ time-slot in increasing size order do
        if it is possible to assign all $l_i$'s consecutive lectures to $r$ then
            break;
until all the lectures are assigned to a classroom
```

- $O1$: For each time-slot, we compute the sum of all the used classrooms seats and the sum of all the seats required by the lectures in each time-slot. The difference between these two values represent the total unused seats of each time-slot. The sum of the free seats for all time-slots is the number of free seats during each week.

- $O2$: For each classroom we calculate the number of times it is used during the week. We use this information in equation 7.

According to their quality, l_{best} and g_{best} particles are identified.
At each iteration, position of particles are modified using three different movements. For each time-slot one of the next movements is performed:

- *swap*: In this case, a random occupied classroom is selected and verified for changes according to its velocity, then it is exchanged by another random selected classroom in the same time-slot. The objective of this movement is to incorporate diversity to the search process.

- *Guided swap to* l_{best} [4]: In this case, a random classroom is selected and verified for changes according to its velocity, then the same position value in l_{best} is assigned to it. This change can produce one or several swap movements of lectures in a single step in order to ensure feasibility of solutions.

- *Guided swap to* g_{best} [4]: The same procedure of *Guided swap to* l_{best} is applied in this case, but it is performed considering the g_{best} particle.

The movements produce feasible solutions. Swaps of consecutive lectures are performed only if all consecutive lectures can be exchanged to the new classroom. Capacity and facilities requirements are always checked before changes.
The velocity of each particle is used in the movements in order to verify for each element if it is able to be changed or not. Velocity is updated each iteration according to equation 8.

$$V^{(t+1)} = \omega * V^t + rand(0, c_1) * |(l_{best})^t - X^t| + rand(0, c_2) * |(g_{best})^t - X^t| \tag{8}$$

The algorithm performs 10000 iterations with a population of 6 particles, an inertial weight ω of 0.5 cognitive ratio c_1 of 0.5, a social ratio c_2 of 0.5 and a maximum velocity V_{max} of 1.0.

3.2 Dynamic Approach

For the dynamic problem we propose a local-search approach that works on the solution found by the particle swarm algorithm. For this algorithm we define two main parameters of the search: the maximum number of tries to find a solution and the maximum number of perturbations they can produce in the schedule.

Local-search procedure This approach performs an incremental hill-climbing procedure. It uses as representation a matrix of size $R \times T_r$, where T_r corresponds to the number of time-slots needed by the requirement:

1. It starts trying to assign the new requirement to an empty classroom. If it is possible, then a solution with no swaps has been found.

2. It tries to move one lecture assigned to a classroom which satisfies the requirement to an empty classroom. If it is possible, then a solution with one movement has been found.

3. It tries to swap a lecture assigned to a classroom which satisfies the constraints with another lecture in the time-slot. If it is possible, this assignment will produce a solution with two perturbations.

4. At each new step, the hill-climbing method increments the maximum number of swaps allowed in the static planning. This incremental feature of the algorithm is focused on reducing the number of updates required to satisfy the constraints. The search process ends when a maximum number of solutions is reached or it is not possible to find more solutions with the maximum number of swaps allowed.

4 Experiments and Results

For our experiments, we use a real problem data obtained from the UTFSM. We consider around 950 lectures, 49 classrooms and 48 time-slots in a week.

4.1 Static problem results

For the first static solution, we compare our particle swarm approach with both, the manual one used at the UTFSM, and a forward checking based algorithm, which is a complete technique. In the following, we first give a short overview of the forward checking algorithm and then we present the results obtained.

4.1.1 Forward Checking algorithm

In the forward checking approach [6], we consider as variables the set of all lectures which require classrooms: $\mathcal{L} = \{l_1 \dots l_L\}$. The domain of each lecture is composed by the set of classrooms of the University: $\mathcal{R} = \{r_1 \dots r_R\}$. At the beginning of the algorithm a node consistency process is performed in order to delete from the domain of each lecture the classrooms which do not satisfy the capacity and facilities requirements. After the node consistency process, variables are ordered according to their constraints and domain sizes. During the variable ordering step, consecutive lectures of the same course are kept together because they have the same capacity and facilities constraints. The domains are ordered according to the capacity of the classrooms, starting with the smallest ones, in order to produce good quality solutions at the first steps of the search, when the evaluation function *O1* is used. The forward checking method we use includes an optimization criteria that allows to develop the tree search just when the algorithm detects that it can obtain a better solution than the previous one. The quality of a solution is measured according to the evaluation functions *O1* or *O2* previously defined. Given the size of our problem it is impossible to verify optimality within hours.
At each step, the method assigns the current lecture to the first available classroom in its domain and performs the three following processes:

- checking of the optimality criteria in order to continue the construction of the current solution just when it is possible to find a better quality solution than the current one.
- filtering of the domain of all the non-assigned lectures connected to the actual lecture, i.e. the lectures which have in common the time-slot and the consecutive lectures it could have.
- assignment of the same classroom to all the consecutive lectures the actual lecture could have. The assignment of the consecutive lectures implies a second filtering process which could eventually detect an unfeasible assignation.

The filter processes can produce empty domains, which means that a solution can not be constructed with the current assignments. In this case, a backtrack jump is required. A Graph Based Jump (GBJ) is used to directly go back to the last lecture which has in common the time-slot. When the jump is directed to a lecture which has consecutive lectures, then the jump is re-directed to the first consecutive lecture in the tree. The corresponding filters are cleared and the process continues.
When all the lectures are assigned to a classroom, then a new best quality solution has been found.
In this case we use as the termination criteria the maximum amount of time the algorithm is able to search for better solutions.

4.1.2 Results

Graph in Fig. 1 shows the frequency classroom use for the current manual solution. Axis y shows the frequency of use of each classroom to be assigned and axis x shows the classrooms of the university ordered by capacity, starting with the smallest rooms. Graphs in Fig. 2 and 3, show the frequency of use of classrooms of the best solution found by forward checking and the best solution found by particle swarm optimization using the first objective criterion ($O1$).
The solutions obtained considering this criterion, are quite different from the manual obtained. The new solutions try to use in a more efficient way the resources “rooms”. We can see that the algorithms trend to obtain solutions which strongly use the smaller classrooms. This gives more opportunities to the dynamic step for finding a big classroom for an exam. Our PSO based approach is faster than the forward checking based algorithm. The forward checking based method requires around 0.01 seconds to find the first feasible solution and the average time required to find a new better quality solution is around 3000 seconds. The solution in figure 2 is the best solution found by the forward checking method after 10000 seconds. Given the size of our problem, around 1000 variables and 49 domain size it was not possible to prove optimality. Our technique requires 500% less CPU time to obtain an equivalent quality solution. Moreover, PSO is able to find various feasible solutions when forward checking has obtained at most three feasible solutions in the same time. The problem has many feasible alternative solutions and some of the intermediate solutions can be used given some particular requirements, for example a blocked classroom. It is important to note that these results allow an easier re-assignment of the exams requirements on the dynamic step.
Graph in Fig. 4, shows a typical solution obtained using the optimization criterion $O2$. In this case, both algorithms are able to quickly find the best solution. We can observe that the objective is reached, assigning each classroom to $19-20$ lectures each week.

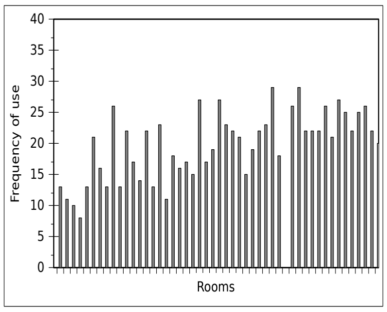

Figure 1: Manual built solution

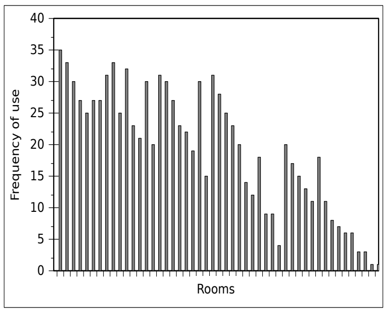

Figure 2: Forward Checking solution

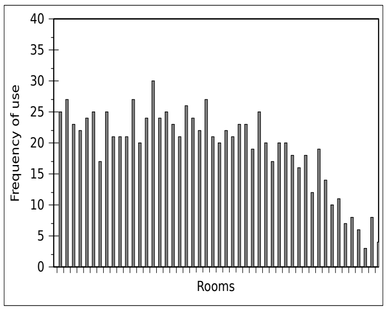

Figure 3: Particle Swarm solution

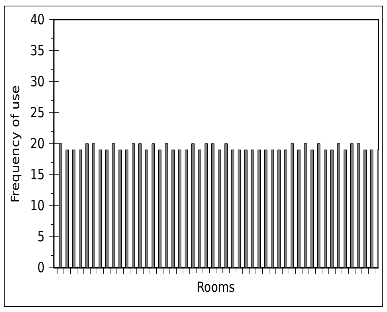

Figure 4: Solution considering objective *O2*

Occupancy rate	rooms-request	feasible-rooms	free-rooms	swaps
0.87	7×110	8	1	2
0.87	8×80	15	2	2
0.87	8×50	46	6	1
0.80	7×110	8	1	1
0.80	8×80	15	4	1
0.80	8×50	46	9	0
0.73	7×110	8	3	1
0.73	8×80	15	4	1
0.73	8×50	46	10	0

Table 1: Number of swaps performed to solve the dynamic problem

4.2 Dynamic problem results

For the dynamic phase we perform experiments defining several typical requirements of rooms for exams in the most constrained conditions. The experiments are performed considering the manual built solution (shown in Fig. 1) because it considers the current situation of the dynamic problem. Table 1 shows the three defined scenarios. These scenarios are defined according to the occupancy rate of each time-slot. The occupancy rate is defined as the rate between the number of occupied classrooms and the total number of rooms for each time-slot. Table 1 shows scenarios with occupancy rate ranging from 0.73 to 0.87. For each occupancy rate we analyze three typical rooms request. The first one requires 8 rooms of 110 seats capacity. The second one requires 8 rooms of 80 seats capacity and the last one requires 8 rooms of 50 seats capacity. The third and fourth columns in Table 1 show the number of feasible classrooms for the capacity required and the number of them that are available in the time-slot of the planing. The last column shows the number of perturbations made by our approach to find a feasible solution. Our algorithm is able to obtain solutions with little perturbations in just a few seconds. In the results of Table 1 we observe that, according to the difficulty of the problem the algorithm is able to find solutions that satisfy the rooms requirements using the idea of swaps and satisfying the high classroom demand in the University. The results have been obtained considering the current manual classroom allocation, which represents a more complex scenario compared to the classroom allocation obtained using the methods for the static problem where biggest rooms are practically free and directly available to be used to satisfy new examination requirements.

5 Related Work

The timetabling problem (TP) is a combinatorial problem that can be viewed as an optimization task. It consists of assigning schedules to several workers or students, which also require some resources [11]. In order to get a feasible timetable, a set of hard constraints must be satisfied (most of them technical constraints); moreover, a good timetable must satisfy some soft constraints (frequently, comfort-related constraints), and if all soft constraints are met, we can consider the solution as optimal. This NP-hard problem presents several variants, such as the school [9], [13]; employee [7], [10]; exam [2], [8] and university timetabling problems [1], [12].

6 Conclusions

In this paper we have proposed a particle swarm approach, usually used to solve continuous problems, to the dynamic course and exam timetabling problem. Our approach can efficiently handle the creation of new courses or exams after the initial static start-up planning. When applied to a real-world problem at the UTFSM, our particle swarm approach has shown to solve very quickly the initial static problem, optimizing the use of small classrooms, to let the biggest ones available for exams. Therefore, finding a classroom with a big capacity for an exam becomes easier for the local search procedure during the dynamic phase of the timetabling. We have shown that our approach can find very good quality solutions within minutes, where a traditional forward checking method needs hours to obtain comparable quality solutions.

References

[1] E. Burke, D. de Werra, and J. Kingston. Applications to Timetabling. In J. L. Gross and J. Yellen, editors, *Handbook of Graph Theory*, pages 445–474. CRC Press London, 2004.

[2] E. K. Burke, D. Elliman, P. H. Ford, and R. F. Weare. Examination Timetabling in British Universities: A Survey. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 76–90. Springer-Verlag, 1996.

[3] S.-C. Chu, Y.-T. Chen, and J.-H. Ho. Timetable Scheduling Using Particle Swarm Optimization. In *Proceedings of the First International Conference on Innovative Computing, Information and Control*, pages 324–327. IEEE Computer Society, 2006.

[4] X. Hu, R.C. Eberhart, and Y. Shi. Swarm Intelligence for Permutation Optimization: A case study of N-queens problem. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 243–246, 2003.

[5] J. Kennedy and R.C. Eberhart. A Discrete Binary version of the Particle Swarm Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 4104–4108, 1997.

[6] V. Kumar. Algorithms for Constraint-satisfaction Problems: A Survey. *AI Magazine*, 13(1):32–44, 1992.

[7] A. Meisels and A. Schaerf. Modelling and Solving Employee Timetabling Problems. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):41–59, 2003.

[8] R. Qu, E. K. Burke, B. Mccollum, L. T. Merlot, and S. Y. Lee. A Survey of Search Methodologies and Automated System Development for Examination Timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.

[9] A. Schaerf and L. Di Gaspero. Local Search Techniques for Educational Timetabling Problems. In *Proceedings of the 6th International Symposium on Operations Research*, pages 13–23, 2001.

[10] A. Schaerf and A. Meisels. Solving Employee Timetabling Problems by Generalized Local Search. In *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, pages 380–389. Springer-Verlag, 2000.

[11] G. Schmidt and T. Ströhlein. Timetable Construction - An Annotated Bibliography. *Computer Journal*, 23(4):307–316, 1980.

[12] K. Socha, M. Sampels, and M. Manfrin. Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In *Proceedings of Evolutionary Computation in Combinatorial Optimization*, volume 2611 of *LNCS*, pages 334–345. Springer-Verlag, 2003.

[13] R. Willemen. *School Timetable Construction: Algorithms and Complexity*. PhD thesis, Technische Universiteit Eindhoven, 2002.