Analysis of Deterministic Timetabling Strategies for the Generation of University Timetables

[1]Marco A. Peña-Luna, [1,2]Adán Hirales Carbajal, [2]Andrei Tchernykh, [1]Miguel Salinas Yañez,
[1]Center for Applied Engineering, CETYS, Tijuana, B.C. 22210, Mexico
{marco.pena, adan.hirales, miguel.salinas}@cetys.mx
[2]Computer Science Department, CICESE Research Center, Ensenada, B.C. 22860, Mexico
{chernykh, ahiralesc}@cicese.mx

Abstract

We present an experimental study of deterministic timetabling strategies for the generation of university timetables. We distinguish three prioritization criteria, which differ by the type and amount of information required to select the courses to schedule. We analyze the performance of the time tabling strategy and prioritization criteria by using a workload with 126 courses, 65 professors and 12 class rooms from a real scenario. To analyze the performance, we conducted a joint analysis of three metrics. We evaluate the quality of the obtained solutions in terms of their degradation and performance profiles.

Keywords. *University timetabling, Scheduling.*

Análisis de estrategias determinísticas de calendarización, para la generación de horarios universitarios

Resumen.

Presentamos un estudio experimental de una estrategia de calendarización determinística para la generación de horarios universitarios. Distinguimos tres criterios de priorización que se diferencian por el tipo y cantidad de información que requieren para seleccionar materias a calendarizar. Analizamos el desempeño de la estrategia de calendarización y de los criterios de priorización al aplicar una carga de trabajo con 126 cursos, 65 profesores y 12 salones correspondientes a un programa educativo universitario real. Con el fin de analizar su desempeño realizamos un análisis conjunto de tres métricas. Evaluamos la calidad de las soluciones obtenidas en términos de su degradación y perfil de desempeño.

Palabras clave: *Calendarios universitarios, calendarizacion.*

1 Introduction

This work studies the university timetabling problem. It addresses a deterministic version of the problem, since knowledge needed to produce a solution is known. Only valid solutions are considered. Solutions that violate hard constraints are discarded. The quality of solutions is evaluated under the perspective/objectives of two actors: the university as the owner of the infrastructure (classrooms), and the professors. We conduct a joint analysis of metrics, giving equal importance to the interest of both actors.

We propose a heuristic for university timetabling, namely Deterministic Time Tabling Strategy (DTTS). DTTS systematically applies a course reservation mechanism called Resource Reservation Strategy (RRS). RRS creates a set of reservations, if the hard constraints can be satisfied for a given course.

We evaluate DTTS performance in terms of its capacity to schedule a set of 126 courses and 65 professors over 12 resources corresponding to a university department. Solutions are evaluated in terms of three metrics: scheduled courses rate, resource utilization and professor's schedule dispersion. DTTS robustness is analyzed by evaluating its capacity to schedule the workload under different resource capacity conditions.

2 Previous work

The timetabling problem consists of the allocation of a sequence of jobs on a set of resources, while satisfying a set of constraints [21]. In its more generic form, it implies the assignment of courses and professors to classrooms in specific time periods, so that no single professor is assigned to more than one classroom at the same moment, and there is no classroom with more than one course at the same time instance [20]. In summary, the objective of timetabling is to find the maximum set of time periods that satisfy space and time restrictions given by professors, resources, and students.

In the context of university course timetabling, the following elements are disti nguished: academic program, a composition of courses within the same field of knowledge; the course with a series of lectures associated to it. If it is possible, courses are allocated to slots or time windows, and professor, a human resource associated to a university department. The professor teaches one or more courses; student belongs to a university department and can be a part of a group, classroom or physical resource, where lectures take place.

In [18, 7, 21], two types of constraints are distinguished: soft and hard. A solution must meet all hard constraints, but it might omit soft constraints. A solution is considered valid, if all hard constraints are satisfied [7]. It is considered feasible, if it is valid and all courses are scheduled [17]. In addition, it is termed high quality solution, if it satisfies soft constraints [21].

Schaerf [19] distinguished three variants of the timetabling problem: School timetabling (STT), Course Timetabling, and Examination timetabling (ETT). STT deals with weekly scheduling of classes. It considers two hard constraints: a professor cannot conduct simultaneous lectures and concurrent lectures cannot occur in groups. Course Timetabling, deals with weekly scheduling of university courses, it includes STT constraints and aims to minimize overlapping of courses shared by common sets of students. Examination timetabling refers to the scheduling of university departmental exams. ETT aims to avoid overlapping of examination days. Its objective is to distribute exam application dates as much as possible within the exam period.

The timetabling problem has been treated as a deterministic problem, because all characteristics of the problem are known. That is, the properties of courses, professors and resources are known. The timetabling problem, even for simple formulations, is NP-Complete [13, 12, 6, 19, 21]. A solution to the problem can be verified in polynomial time. However, there is no known efficient way to find solutions in polynomial time, in terms of

the size of the problem, unless NP equals to P. To our knowledge, there is no approximation algorithm capable of reducing the solution space. In the literature, the timetabling problem has been addressed via: sequential methods, clustering heuristics, constraint programming, and heuristics, such as genetic algorithms, Tabu search, simulated annealing, and constraint satisfaction [5, 6].

Gans [11] proposed a combinatorial heuristic used to schedule 21 school programs. Foundsolutions satisfied all hard constraints. Glover y McMillan [14] utilized administrative science and AI to schedule a workload with more than 100 employees. Aubin and Ferland [4] proposed the generation of a master timetable by grouping independent timetables. They proposed a heuristic that penalizes overlapping and excessive use of resources. Hertz [15] and Costa [10] used Tabu search to reduce the number of conflicts on shared resources. Colorni [8] evaluated the quality of solutions produced by a genetic algorithm with those of different versions of simulated annealing and Tabu search strategies. Predominant characteristic of previous works are the use of AI techniques and single criteria evaluation of solutions.

Constraint satisfaction approaches have also been applied [16]. They are characterized by the type of constraints (hard and soft), domain (discrete and continuous); and the amount of constrains to satisfy (binary, unitary, multiples and n-ary). CSP are studied using operation research and AI technics, such as: search trees, look back algorithms (back-jumping, conflict-directed back-jumping), look-ahead algorithms, and stochastic methods (hill climbing, Tabu search, simulated annealing, genetic algorithms, among others).

3 Problem definition

University timetabling problem is formulated as follows.

There exists a planning horizon [0,T] divided in periods [t, t + 1], with t = 0,1,⋯,T-l. Period duration is constant, one hour in our study. There exist m resources, each has its own availability and planning horizon, both of size T. We assume full availability of resources.Therefore, no reservations exist prior the scheduling process. Resource capacity is not considered.

n courses j = l,⋯, n are scheduled within the planning horizon. A course is described by the tuple (ps_j, pl_j, pc_j, pt_j, hcc_j, id_j) with duration of lecture $ps_j \geq 1$; lab hours $pl_j \geq 0$, workshop hours $pt_j \geq 0$, clinical hours $pc_j \geq 0$; release time $r_j = 0$, and course identifier id_j Initially all courses are released at time 0. During the scheduling process their release times are set according to availability of the professor and the resources available.

The number of employees $D_j(t)$ who teaches course t_j in the period [t, t + 1] is one. Given a set E of employees, with |E| = k , each employe e ∈E is qualified to teach a subset Q_e of courses. For example, Q_e {1,2} implies that employee is qualified to teach only courses 1 and 2.

The availability pattern of employee e is defined as following.

A binary array $\langle w_e(t)\rangle_{t=0}^{T-1}$. $w_e(t)$ = 1ife is available in the time slot [t, t + 1], otherwise $W_e(t) = 0$, which means that there is no availability.

When a course is allocated into Q_e at a time slot [t, t + 1], the value of $W_e(t)$is changed from 1 to 0.

A planning horizon is a composition of days from Monday to Saturday. Other days combinations are possible. We consider a daily window of 16 hours (1:00am to 11:00pm), hence T = 96 hours.

A working pattern in a time window is a binary array $\alpha = (\pi\ (j,t))$, such as $\pi(j,t) = 1$ if in the time slot $[t,\ t + 1]$ course t_j is imparted, otherwise its value is cero. After the scheduling process, professors and each resource have a working pattern.

The following hard constraints are considered:

1. There is no preference for employees. Employee selection depends on the course list to schedule.

2. Scheduling is done for one university department programs. Resources are not shared among departments.

3. Hours of Consecutive Class HCC_j of a course t_j are finite. Furthermore, we limit the length HCC_j of to be no greater than 16 hours.

4. It is not allowed to have two or more HCC_j in the same day. HCC_j models class hours, lab hours, workshop hours, or clinic hours.

5. Summation of all HCC_j corresponding to course must be equal to the sum of the hours specified in its tuple.

6. An employee $e \in E$ cannot teach courses t_j and t_i, $t_j, \neq t_i$, at the same time period $[t, t + 1]$.

7. Similarly, courses t_j and t_i, $t_j \neq t_i$, cannot be taught during the same time period $[t, t + 1]$ in the same resource $r \in \{1, 2, ..., m\}$.

A solution that satisfies these constraints is valid, even, if it does not schedule all courses in the list. The concept of the group is not considered.

4 Scheduling heuristics

Requirements of a course are satisfied when its vector of durations (ps_j, pl_j, pc_j, pt_j) is assigned to one or more resources. During the scheduling process durations are fragmented into blocks of size HCC_j and scheduled as early as possible in the employee availability window. For each block, a reservation is made in the resource that satisfies constraints 1 to 7. This scheduling process is denominated Resource Reservation Strategy (RRS).RRS selects a resource among the m resources available, by means of a selection criterion. For example: the resource with less reservations, randomly, with less of more capacity, etc.

Given the course t_j, RRS produces a set of reservations, if and only if all of its durationsare scheduled. Otherwise reservations are cancelled.

Pseudocode 1: RSS

1. RRS reads the identifier e of the employee asociated to t_j and makes the release time r_j of t_j equal to the first available time of employee e.
2. Given course t_j, RRS iterates while durations or some constraints of t_j are not satisfied.
3. RRS selects the duration d_j from duration vector $(ps_j, pl_j, pc_j, pt_j)$of t_j. The vector is visited sequentially starting with ps_j. Subsequent duration is not read until the number of hours from the immediate predecessor is cero.
4. If d_j is cero, RRS removes the tag from t_j, return to 2.

5. if duration d_j is bigger than cero. RRS checks if course t_j was previously assigned to a resource. A scheduled task is tagged with the identifier of resource r if it was previously assigned to that resource. Tag is removed when duration class d_j is cero.
6. RSS creates a search area $A = [r_j, r_j + HCC_j]$
7. if$A$ do not belong to the horizon of employee e, $A \notin H_e$, RSS increases $r_j = r_j + 1$, return to 6.
8. if$A$ do belongs to e horizon, RRS checks if there exist availability in time interval A in each resource u. Let $s_u = \{(r_j, r_j + HCC_j), null\}$ and let $disp$ a no null availability vector $disp = \{u | u = 1, \cdots, mys_u \neq nulo\}$. Note that $0 \leq |disp| \leq m$.
9. if $|disp| > 0$, that is, there exists one or more resources available in interval $[r_j, r_j + HCC_j]$
    a. if r is not null and $r \in disp$, RRS selects resource r
    b. if$r$ is not null and $r \notin disp$, RRS selects a resource r applying a selection criteria. For example, resource with minor number of reservations; randomly; resource with smaller or larger capacity, etc.
    c. if$r$ is null, RRS selects a resource r applying a selection criteria (same as 9.b.)
10. RRS makes reservation permanent in r; reduce duration of d_j to $d_j - HCC_j$; eliminates profesor availability in time interval $[r_j, r_j + HCC_j]$; tags t_j with r; and modifies release time r_j of t_j to the next employee e available time (release time r_j should correspond to the next day in the employee availability); set $w_e(t) = 0$, $t = r_j, \cdots, r_j + HCC_j$. Return to 2.

We assume an atomic reservation process. Whenever possible, course t_j reservations are made at the same resource, but at different days. Initially there is no knowledge about resourcesfor reservations.

Once the first HCC_j block is allocated the resulting reservation is tagged with the id of the selected resource. The tag is used to keep memory of previous scheduling decisions. Subsequent reservations, for the same course, are allocated if possible to the tagged resource. The tag is removed if a reservation can no longer be done at the tagged resource. Pseudocode 1 shows the RSS heuristic.

The scheduling heuristic consists of two phases: reservation and confirmation. Given a course, the reservation phase tries to create a set of reservations. Reservations are regarded as tasks in subsequent paragraphs. In the confirmation phase, tasks are sent to their respective resources, thus making reservations permanent.

Given n courses j = 1,⋯, n, the DTTS systematically takes one course t_j, applies RRS and produces a set of reservations or none. DTTS keeps two logs: courses with reservations (TcR) and courses that could not be scheduled (Task with Failures, TcF). RRS generated reservations are registered in TcR, while courses with no reservations are stored in TcF. Once all courses are processed by RRS, DTTS sends the tasks in TcR to their corresponding resources. The local scheduler at each resource receives the task and makes the reservation permanent.

DTTS receives a list of courses to schedule. Each entry of the list contains the following fields: class hours ps_j, lab hours pl_j, workshop hours pt_j , clinic hours pc_j, release time r_j, identifier id_j of course t_j and identifier e ∈ E of the employee qualified to teach t_j. DTTS produces a schedule for each resource. Each schedule contains the following fields: release time r_j, completion time c_j , identifier of the course , identifier of employee e.

Pseudo code 1 describes the execution of the RSS. Alternate flows try to find time intervals compatible between course requirements, employee and resource availability window. When course requirements are not satisfied both availability windows are restored to previous state.

Employee and resource horizons are updated each time a task is assigned to them. Reservations made over one or more horizons are cancelled when, given the requirements of a course, compatible windows between employee and resources do not exist.

Pseudo code 2 describes DTTS execution flow, which uses RRS and generates TcF and TcR logs.

Pseudocode 2. DTTS

```
1. For each course t_j in the workload
2. res = call RSS with t_j
3. if the set res (reservations) is empty then
4.   store t_j in log TcF
5. otherwise
6.   store reservations res in log TcR
7. end
```

5 Methodology

5.1 Ordered lists

Previous to the scheduling process, all courses are inserted into a list. The course in the head of the list is scheduled first. One course is scheduled at a time. Three different orderings are studied: Random (A), there is no preference in the selection of courses; Part Time – Full Time (AP), courses taught by part time professors are scheduled first; Proportion of Hours Required from the Professor (PHRP), courses corresponding to professors with higher proportion of required hours are scheduled first. We evaluate DTTS and RSS analyzing its performance on the three list orderings. We evaluate the algorithm robustness analyzing its performance varying the available number of resources: 6, 8, 10, …, 26 and 28 resources.

5.2 Experimental design

Table 1 summarizes different aspects of the experimental design.The number of faculty and courses to teach is 65 and 126 respectively. 18 full time professors are considered, each with an availability window of 8 hours with recess from 1:00 PM to 3:00 PM. The rest of the professors are part time. Their availability varies. A total of 541 hours should be scheduled. The workload corresponds to real requirements of an Engineering School, Mexicali, CETYS University, January-June 2013 semester.

Table 1. Experimental design

Factors	Levels
Workload	Course Priority list. Three orderings: Random (A), Part Time – Full Time (AP); Proportion of Hours Required from the Professor (PHRP)
# of resources	6, 8, 10, …, 28
Dependent Variables	Utilization, time dispersion and y scheduled courses rate
Replication	400
Design	Factorial (2 factors)
ProblemModel	Deterministic

5.3 Metrics

A good solution aims to satisfy expectations from both the owner of the resources and form faculty. University administrators aim to maximize resource utilization and scheduled courses rate, while faculty prefers to minimize dispersion in their course schedule.

We distinguish three metrics: utilization $U = \frac{1}{HT * m}\sum_{g(t_j)=(m_k)} p_j$, where $p_j = ps_j + pl_j + pc_j + pt_j$

represents the duration of course t_j, HT is the total number of hours to schedule, and $g(t_j)$ is a function that assigns course t_j to a set of resources $\{m_k\}$; scheduled course rate PCC =|{scheduled courses} / |{courses}|; and normalized average time dispersion.

Let t_i y t_j be two tasks corresponding to one or more courses in Q_e such that i α jwhere i ≠ j, that is, is an immediate predecessor of in the same working day. Average time dispersion of employee e is $d_e = \frac{\sum_{(i,j)\ni i\alpha j}(s_j - c_i)}{n_e}$, where s_j is the start time of t_j; c_i is the end time of t_i; and n_e is the number of tasks in the working window of employee . We normalize the employee average time dispersion $dn_e = \frac{\bar{d}_e}{\max_{e \in E}[\bar{d}_e]}$, such that when dn_e tends to one average dispersion in employee *e* working window is big. It is desirable that dn_e tends to cero. Finally, normalized average time dispersion is $dn = \frac{\sum_{e \in E} \overline{dn}_e}{|E|}$.

For professor , we define PHRP as the rate of the number of hours required to teach all courses in Q_e over the number ~~of~~ hours in employee *e* availabi~~lity~~ window.

The workload is composed of a sample of 400 lists of the 126! possiblelists. A degree of variability of 50%, 95% confidence level, and +/- 5% precision level is used to determine the sample size. A list L with AP ordering is generated by creating two sub-lists: L_a for courses taught by part time professors and L_p for courses taught by full time professors.

Elements in lists y are permutated using the algorithm that randomizes in place [9], then L_a y L_p are concatenated in . A list with PHRP ordering is generated using the set of lists $\{L_1, L_2, \ldots, L_k\}$. List L_i contains courses of professors with same PHRP. Lists are sorted in descending order according to their PHRP index and are concatenated into L.

6 Results

6.1 Increasing number of resources

Figure1 shows the average of scheduled course rate. DTTS performance using PHRP is marginally superior to AP and Random. 60% of courses are programed with only six resources, independently of the list ordering. Average of course rate using PHRP approximates 100% by increasing the number of available resources. With 14 resources AP and Random rate approximates 92%, while PHRP attains 100%.

Results presented in Figure 1 consider valid solutions, that is, those which satisfy all hard constraints. However, AP and Random do not find feasible solutions with a reduced number of resources. Result presented in Table 2 show that PHRP, AP, and Random produce feasible solutions with 14, 20 and 24 resources respectively. From the 400 course lists ordered under PHRP, 194 produce feasible solutions. Additionally, PHRP finds the higher number of feasible solutions.

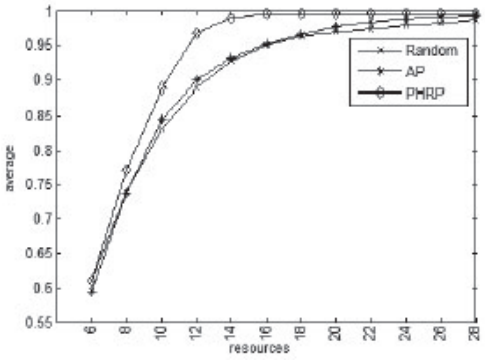

Figure 1. Average scheduled course rate

Table 2. Number of generated solutions

	Feasiblesolutions		
Resources	Random	AP	PHRP
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	71
16	0	0	193
18	0	0	194
20	0	2	194
22	0	16	194
24	3	51	194
26	6	89	194
28	17	131	194

DTTS systematically takes an availability pattern and tries to find a subset of time slots to assign the professor courses. PHRP selects availability patterns where a higher number of courses can be scheduled. We denominate patterns with this characteristic as congested. PHRP produces solutions with higher scheduled course rate by creating opportunities for a higher number of compatible congested patterns. We say that an availability pattern is compatible if it does not overlap with other patterns.

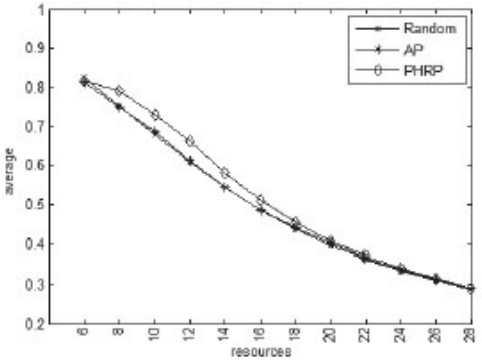

Figure 2. Average resource utilization

Independent of the list ordering, a maximum utilization of 80% is obtained. It is achieved, when 6 resources are used (See Figure 2). As expected, utilization decreases when the number of available resources increases. PHRP utilization is higher than the one obtained using AP and Random in the interval between 8 and 18 resources. This happens because PHRP finds a higher number of compatible congested windows. With 12 and 14 resources DTTS+PHRP achieves utilization between 66% and 58%, which is comparable with results presented in **[Error! Reference source not found., Error! Reference source not found., Error! Reference source not found.]**. Therefore, we can conclude that timetables generated by DTTS produce reasonable resource utilization.

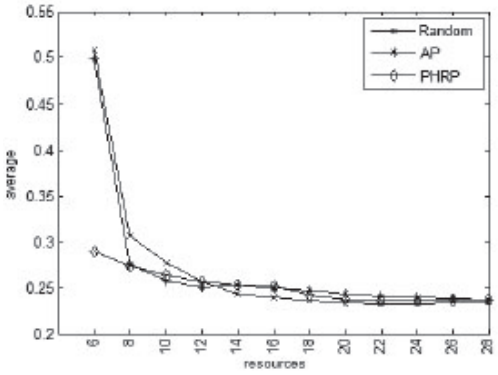

Figure 3. Normalized average professor time dispersion

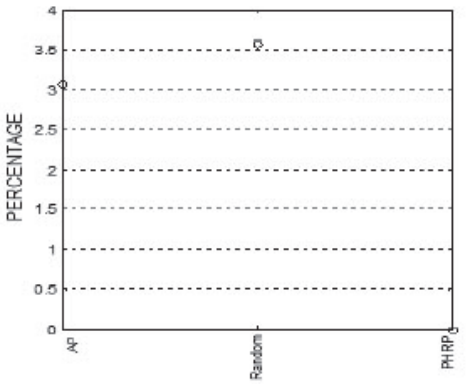

Figure 4. Degradation of Avg. scheduled course rate

Figure 3 shows professor time dispersion. When 6 resources are used, AP and Random produce solutions with normalized average dispersion of approximately 0.5, while PHRP have a dispersion of 0.3. Solutions generated by PHRP have 20% less dispersion than the other two orderings. With 12 resourcesor more, dispersiontends to 0.25.

6.2 Degradation

In this section, we evaluate DTTS degradation using three input list orderings, 126 courses, 65 professors and 12 classrooms (resources). We select 12 resources from real scenario of 12 resources in the semester timetable.

Degradation is defined as the rate between a generated solution and the best generated solution for one metric of interest:

$= \left(\frac{\text{metric value}}{\text{best found metric value}} - 1\right) * 100$. A degradation of cero is desirable.

From Figures 4, 5, and 6, it is not possible to measure, with a reasonable precision, how better one ordering with respect to another one. However, figures show that DTTS produces better solutions using PHRP for all metrics.

Solutions generated by AP and Random are 3% and 3.5% worse than those found by PHRP in terms of scheduled course rate. AP and Random utilization are approximately 3.6% and 4% worse than PHRP. Time dispersion in solutions generated by AP and Random are approximately 7% worse than those generated by PHRP. In summary, AP and Random obtain an average degradation of 4.5 and 5, while PHRP obtains the best average degradation, see Figure7.

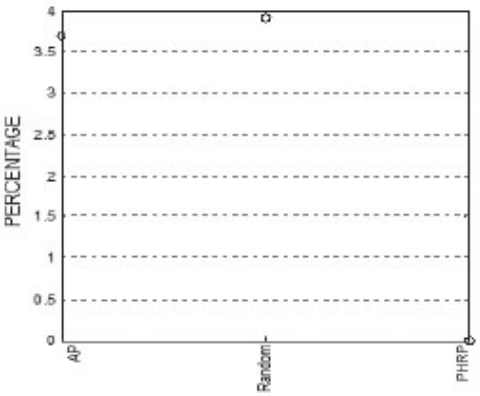

Figure 5. Degradation of Avg. resource utilization

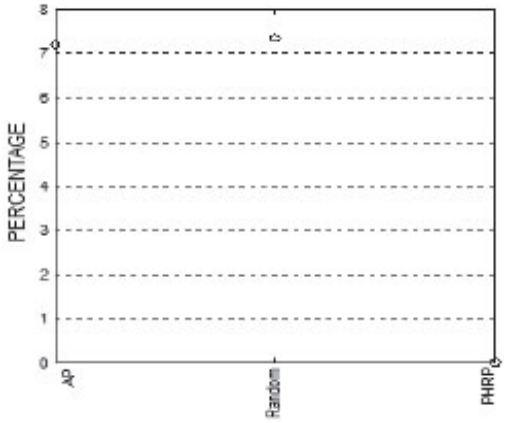

Figure 6. Degradation of Avg. professor time dispersion

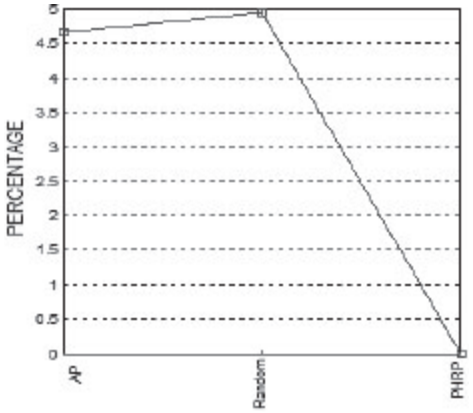

Figure 7. Avg. degradation of the three metrics

6.3 Performance profiles

We analyze DTTS performance profile with Random, AP and PHRP using 12 resources. We found that 50%, 30% and less than 10% of the solutions generated by PHRP, AP and Random are equal to the best generated solution in terms of the scheduled course rate (see Figure 8). However, within a factor of 0.02 from the best generated solution, the percentage of AP and Random generated solutions quickly increase to 80% and 70% respectively. It is important to remark that this happens because of the capacity of the strategies to find valid solutions, although not necessarily feasible solutions.

Generated solutions are marginally 0.02 times worse or equal to the best generated solution. However, it does not imply that solutions are feasible.

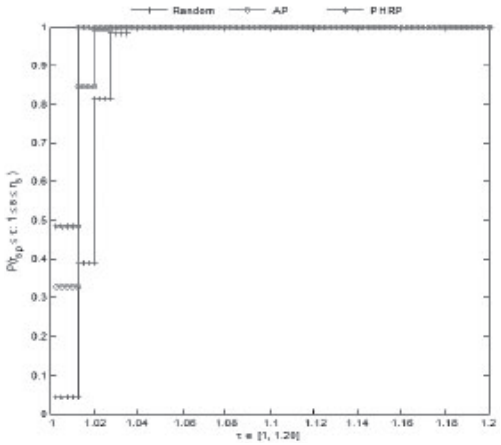

Figure 8. Schedule course rate

We found that, in terms of utilization, approximately 50%, 30% and less than 10% of PHRP, AP and Random, respectively, generate solutions equal to the best generated solution (Figure 9). Within a factor of 0.02, AP and Random increase to 85% and 45% respectively. PHRP had the best dispersion, see degradation analysis. However, the amount of solutions with equal professor time dispersion is near cero (Figure 10). Approximately 40% of Random generated orderings are equal to or up to 1.05 times worse than the best generated solution.

Almost 100% of the solutions generated by PHRP are within a factor of 1.06 from the best generated solution.

From the previous analysis, we conclude that the advantage of PHRP over AP and Random is marginal, when considering valid solutions.

In general AP and Random can produce numerous solutions close to the best generated solution within a factor of 0.03. These results cannot be considered conclusive, if problem restrictions are modified or amounts of resources vary, then response variables can be affected. This case will be studied in the future work.

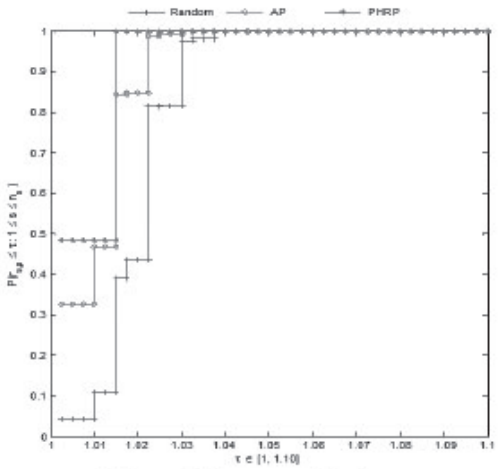

Figure 9. Resource utilization

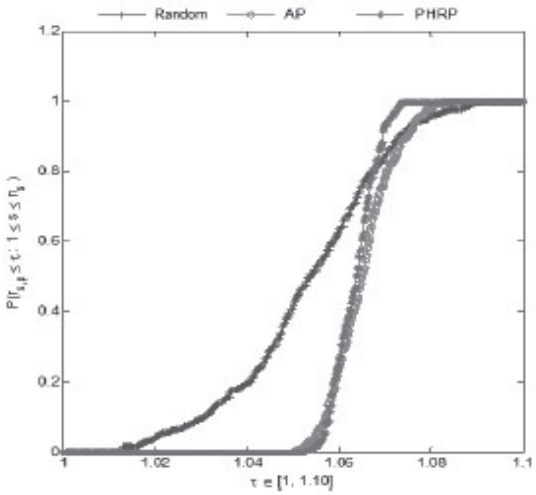

Figure 10. Dispersion

7 Conclusions

University timetabling requires an effective allocation of courses to resources, given a set of hard constraints, different professor availabilities, and frequently a reduced number of resources. It is a combinatory optimization problem and it is classified as NP-Complete, therefore, there is no knowledge of efficient algorithms to find an optimal solution.

In this work, we propose a deterministic heuristic called Deterministic Timetabling Strategy (DTTS) to generate university timetables. We analyze its performance under three different input list ordering conditions (Random, AP and PHRP) to determine which input produce best solutions. We evaluate the quality of the solutions in terms of three metrics: scheduled course rate, utilization, and professor time dispersion. For the selection of the best strategy, we conduct a joint analysis of the metrics in terms of the mean and degradation. In multi-criteria analysis, we assume equal importance of each metric. To address the possibility of biased results, where a small number of instances of the problem have large deviations that affect the average results, we analyze the performance profile of the strategies.
Our study results in these contributions:

- We developed and analyzed the performance of DTTS using different input criteria
- *We proposed and evaluated a mechanism for resource selection, specifically selection in terms of the Least Number of Reservations (LNR)
- We demonstrated that PHRP + DTTS + MNR strategy finds the largest number of feasible solutions via proof of concept. We conclude that it results beneficial to allocate courses of professors with more academic load relative to their availability first.
- We demonstrated that AP + DTTS + MNR and Random + DTTS + MNR possess the capacity to produce high percentage of valid solutions, although not feasible solutions.

When analyzing DTTS performance using real data, we found that an appropriate distribution of courses is achieved by prioritizing the allocation of courses in terms of the course load of professors. At the same time, we useknowledge

of the number of reservations held by each resource; that is, the state of the resource. This is opposed to the idea of giving higher priority to the allocation of full time professors first.

By varying the amount of resources PHRP + DTTS produces the best results in terms of the three metrics and generates the largest number of feasible solutions (194). Using Random and AP orderings, 2 and 3 feasible solutions are generated when utilizing 20 and 24 resources, respectively. PHRP accomplishes this by finding a larger number of independent congested availability patterns. On average, the solutions generated by AP and Random are approximately 4.5% and 5% worse than those generated by PHRP, as both are able to find a greater number of valid solutions.

When analyzing the performance of DTTS evaluating their performance profile, we found that approximately 50% of the solutions found by PHRP + DTTS are equal to the best solution found in terms of utilization and scheduled courses rate. However, being only a factor of 0.03 from the best found solution, about 80% of the solutions generated by Random and AP are valid. From the above perspective, the advantage of PHRP over AP and Random is marginal when valid solutions are considered.

Regarding the amount of knowledge required by the scheduling strategy, we conclude the following: Since DTTS uses MNR, it is necessary to have knowledge of the total number of resources and their status. Having state information is favored to PHRP that allows to find 194 feasible solutions. Knowledge of the professor course load is also useful, as it allows prioritization for allocating courses giving better results by issuing higher importance to professors with bigger academic load with respect to their availability. Computationally, prioritization of resources (MNR) and professors results in a higher computational cost, which increases the execution time of the heuristics but it allows us to find a larger number of feasible solutions.

7.1 Future work

From this study a number of questions arise. We underline the following observations and unknowns.

- DTTS does not consider the concept of group, in future implementations we will incorporate it.
- During the scheduling process, resources are selected in ascending order of the number of reservations they have, giving higher priority to the resource with fewer reservations. Such a mechanism is intrusive since it requires knowledge of the characteristics of the resources. From this, the following questions arise: What is the impact in the quality of generated solutions, having more or less informations about the resources?
- So far, we can only say that we found valid and feasible solutions, and that a valid solution meets all constraints of the problem. However, what are the differences between valid and feasible solutions? Why a valid solution did not schedule all courses?
- The input list is prioritized in terms of each professor academic load. It is interesting to study the possibility of prioritizing the workload together. For example, all professors teaching courses ini-th semester receive the same priority. During the scheduling process, the heuristic tries to assign professors of such a groupfirst.

8 References

1. University of Oregon classroom space utilization study fall term 2012.
2. University of Victoria classroom utilization report 2004/05, November 2005.
3. University of Wisconsin-Madison administrative excellence classroom space utilization, June 2012.
4. J. Aubin and J. A. Ferland (1989).A large scale timetabling problem. Comput. Oper. Res., 16(1):67–77.
5. Edmund Kieran Burke and SanjaPetrovic (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140(2):266 – 280.
6. Michael W. Carter and Gilbert Laporte (1998). Recent developments in practical course timetabling. In Practice and Theory of Automated Timetabling II, volume 1408 of Lecture Notes in Computer Science, pages 3–19. Springer Berlin Heidelberg.
7. Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf (2012).Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Comput.Oper. Res., 39(7):1615–1624.
8. Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo (1998). Metaheuristics for high school

timetabling. Comput.Optim. Appl., 9(3):275–298, March.
9. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2009). Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.
10. Daniel Costa (1994). A tabu search algorithm for computing an operational timetable. European Journal of Operational Research, 76(1):98–110.
11. Onno B. de Gans (1981).A computer timetabling system for secondary schools in the netherlands. European Journal of OperationalResearch, 7(2):175–182.
12. D. de Werra (1985). An introduction to timetabling. European Journal of Operational Research, 19(2):151–162.
13. S. Even, A. Itai, and A. Shamir (1976).On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing, 5(4):691–703.
14. Fred Glover and Claude McMillan (1986). The general employee scheduling problem: an integration of ms and ai. Comput.Oper. Res., 13(5):563–573.
15. A Hertz (1991). Tabu search for large scale timetabling problems. European Journal of Operational Research, 54:39–47.
16. Tomáš Müller, Hana Rudová, and Roman Barták (2005).Minimal perturbation problem in course timetabling. In Practice and Theory of Automated Timetabling V, volume 3616 of Lecture Notes in Computer Science, pages 126–146. Springer Berlin Heidelberg.
17. Clemens Nothegger, Alfred Mayer, Andreas M. Chwatal, and Günther R. Raidl (2012).Solving the post enrolment course timetabling problem by ant colony optimization.Annals of Operations Research, 194(1):325–339.
18. Ender Ozcan, Andrew J. Parkes, and AlpayAlkan (2012).The interleaved constructive memetic algorithm and its application to timetabling. Comput.Oper. Res., 39(10):2310–2322.
19. A. Schaerf (1999).A survey of automated timetabling.Artif.Intell. Rev., 13(2):87–127.
20. Kate Smith-Miles and Leo Lopes (2012). Measuring instance difficulty for combinatorial optimization problems.Computers& Operations Research, 39(5):875 – 889.
21. Ioannis X. Tassopoulos and Grigorios N. Beligiannis (2012).Solving effectively the school timetabling problem using particle swarm optimization.Expert Syst. Appl., 39(5):6029–6040.

Marco A. Peña Luna is a professor in the college of engineering at CETYS University. He received the MS degree in Computer Science and Engineering from the University of Texas at Arlington in 1995 and a Doctoral degree in Engineering with specialization in computer networks and communication systems in 2013 from CETYS University. His interest lies in the area of scheduling, distributed systems, parallel computing and computer networks.

Adan Hirales Carbajal graduated from the Faculty of Sciences of the Autonomous University of Baja California, Ensenada B.C. Mexico, with specialization in Computer Sciences in 1999. He received the MS degree in Computer Science from CICESE Research Center in 2001 and the Ph. D. also in Computer Science from CICESE in 2012. He is a current member of the National System of Researchers of Mexico (SNI), Candidate.His research deals with different aspects of parallel, Grid computing, and cloud computing. His main interests include autonomic computing, scheduling, distributed systems and algorithms, multi-objective optimization, Grid and Cloud computing, parallel computing, and performance evaluation.

Andrei Tchernykh is a researcher in the Computer ScienceDepartment, CICESE Research Center, Ensenada, Baja California, Mexico. From 1975 to 1990 he was with the Institute of Precise Mechanics and Computer Technology of the Russian Academy of Sciences (Moscow, Russia). He received his Ph.D. in Computer Science in 1986. In CICESE, he is a coordinator of the Parallel Computing Laboratory. He is a current member of the National System of Researchers of Mexico (SNI), Level II. He leads a number of national and international research projects. He is active in grid-cloud research with a focus on resource and energy optimization. He served as a program committee member of several professional conferences and a general co-chair for International Conferences on Parallel Computing Systems. His main interests include scheduling, load balancing, adaptive resource allocation, scalable energy-aware algorithms, green grid and cloud computing, eco-friendly P2P scheduling, multi-objective optimization, scheduling in real time systems, computational intelligence, heuristics and meta-heuristic, and incomplete information processing.

Miguel Alberto Salinas Yáñez is the director of the Engineering College at CETYS University. He holds a Ph. D in Engineering with specialization in Computer Networks and Communication Systems. His main interests include autonomic computing, pervasive networking, combinatorial problems, and different aspects of distributed systems.