Towards Improving the Utilisation of University Teaching Space

Camille Beyrouthy[1], Edmund K. Burke[1], J. Dario Landa-Silva[1], Barry McCollum[2,3], Paul McMullan[2,3], and Andrew J. Parkes[1⋆]

[1] Computer Science Dept., University of Nottingham, Nottingham NG8 1BB, UK.
{cbb,ekb,jds,ajp}@cs.nott.ac.uk
[2] Queen's University of Belfast, Belfast, BT7 1NN, UK
{b.mccollum,p.p.mcmullan}@qub.ac.uk
[3] Realtime Solutions Ltd, 21 Stranmillis Road, Belfast, BT9 5AF
b.mccollum@realtimesolutions-uk.com

March 2006

Abstract. There is a perception that teaching space in universities is a rather scarce resource. However, some studies have revealed that in many institutions it is actually chronically under-used. Often, rooms are occupied only half the time, and even when in use they are often only half full. This is usually measured by the "utilisation" which is basically the percentage of available 'seat-hours' that are employed. In real institutions, this utilisation can often takes values as low as 20-40%.

One consequence of such low utilisation is that space managers are under pressure to make a more efficient use of the available teaching space. However, better management is hampered because there does not appear to be a good understanding within space management (near-term planning) of why this happens. Nor, a good basis within space planning (long-term planning) of how best to accommodate the expected low utilisations. This motivates our two main goals: (i) To understand the factors that drive down utilisations, (ii) To set up methods to provide better space planning.

Here, we provide quantitative evidence that constraints arising from timetabling and location requirements easily have the potential to explain the low utilisations seen in reality. Furthermore, on considering the decision question "Can this given set of courses *all* be allocated in the available teaching space?" we find that the answer depends on the associated utilisation in a way that exhibits threshold behaviour: There is a sharp division between regions in which the answer is "almost always yes" and those of "almost always no".

Our work suggests that progress in space management and planning will arise from an integrated approach; combining purely space issues with restrictions representing an aggregated or abstracted version of key constraints such as timetabling or location, and also performing statistical studies to reveal underlying threshold phenomena.

1 Introduction

In this paper we are concerned with understanding the efficient planning and management of teaching space allocation within academic (or similar) institutions. Teaching space includes the usual lecture rooms, but also includes rooms for tutorials, seminars, workshops, etc. Generally, the efficiency of teaching space management is measured by the "Utilisation" U. Exact definitions will be given later, but basically U is a simple measure of the fraction of the available space that is actually used. A utilisation of

⋆ Contact Author (ajp). Authors listed alphabetically.

100% corresponds to every seat being occupied at all available times. Unfortunately, and perhaps surprisingly, utilisation figures are often very low; often around 20-30% in practice. The 'Higher Education Funding Council for England' (HEFCE) has reported low utilisations [20], and two of the authors have commercial experience of such low utilisations from their work with Realtime Solutions Ltd [24, 25]. As another example, in work at the University of Puget Sound in the USA, Fizzano and Swanson [17] report that the registrar asked them space-related questions such as "How many classrooms does the University need to hold the classes it currently offers?". They include as one of their conclusions that "the university is not using all of its classroom space as efficiently as it might". Naturally, many institutions would like to improve this situation in order to reduce costs, improve services, or to identify teaching space that might be converted to other uses (e.g. office space might be in higher demand).

The overall area of management of space can be divided into two broad areas: "Space Management" for the near-term usage of existing resources, and "Space Planning" for long-term decisions relating to the provision of space resources. For example, space management handles the assignment of people to existing rooms, whereas space planning is concerned with decisions as to which rooms ought to be built or re-allocated to different tasks. In particular, the long-range nature of space planning implies that decisions need to be made before the exact details of current timetables, student numbers, etc, become available. One approach to cope with this incomplete information is to rely upon some "tried-and-tested" standard practice. In the case of space allocation problems, this corresponds to relying upon what are called "space norms". An example of a norm might be a physical objective such as "$5m^2$ per Ph.D. student", in which case it can form the basis for space management. Norms of this form can provide the basis for space management in Office Space Allocation (OSA) [34]. Such norms provide a basis for space planning: use the norms to calculate the overall demand for office space, and then design the supply of rooms to closely match the demand. However, this works so well for OSA only because, generally, most offices are all used. Attempting to do this for Teaching Space Allocation is more difficult because the expected low utilisation implies that planners must build in a corresponding excess capacity. Furthermore, expected utilisations are such that this inbuilt "safety margin" has to be as much as a factor of two, or more. This has obvious and large impacts on costs.

However, attempts to remedy this situation, and so to carry out better space planning are hampered because there does not seem to be an agreed or qualitative understanding of why utilisation is so low in the first place. Furthermore, the safety margins incorporated into space norms are obtained from standard sources, the origin of which is generally unclear, and might well be inappropriate for modern module systems, as the sizes of classes might well have changed significantly. Hence, we have two primary goals:

1. to develop an understanding of the factors that lead to low utilisations.
2. to develop methods to chose safety-margins that are more cost-efficient: aiming to reduce the teaching space that needs to be provided, whilst not increasing the risk of it turning out to be inadequate

To these ends, we first consider a simple "pure" event allocation problem in which we optimise utilisation by taking events from a pool of courses and assigning them to the available timeslots. On the datasets we have available, this immediately gave utilisations of 85-90%. This is far too high to match reality, and so indicates that a model based purely on space issues, and given free choice of courses, is inadequate to model the problem of managing teaching space allocation in real-world universities.

To extend the model we move in two independent directions:

Extra Constraints: Event-allocation usually takes place within the context of many constraints on locations and timings of events. Accordingly we include within our model

objectives that are intended to provide a simplified approximation/abstraction of real timetabling issues.

Threshold Phenomena: We study notions arising from the threshold phenomena (also called phase transitions) common in many large systems. Such phenomena arise when typical properties of a system tend to be reliably predictable, based merely on overall properties of an instance; see for example [4]. (Phase transitions in the course timetabling arena have been studied from a different perspective in [31], and we will discuss the differences later).

We find that the location and timetabling-based objectives do indeed have the potential to drive down utilisations, when performing trade-offs in the multi-objective sense. Also, the achievable utilisation measures are statistically predictable, and this supports the case for reliable space planning. We also find that if courses are selected in advance then the reliably achievable utilisation can be much lower than when course selection is by the optimizer.

We remark that the problem classes we use are not new within the general area of course timetabling (for general surveys of the area see [13, 2, 33, 9, 6, 29]). The underlying problem that we will consider is the event allocation problem. This is a simple existing problem and indeed often occurs as the classroom assignment problem [8] and within timetabling problems in order to select feasible room assignments for events. However, in our work it is not automatically a hard constraint that all the events must be allocated, but rather the decision as to which events are used becomes a part of the problem. To the best of our knowledge, this differs from all of the course timetabling literature in which it is a hard constraint that every event must be allocated a place. This also meant we implemented a new solver rather than attempt to use an existing one.

Concerning our choice of potential search algorithm, we remark that recent work in [1] on exact (provable optimal) solutions was limited to relatively small instances of course timetabling; up to 69 courses and 15 rooms. For larger instances it is necessary to use (meta)-heuristics. In this case, a general pattern of the most successful studies is that firstly a constructive algorithm is used in order to produce initial feasible solutions, followed by improvement of the feasible solution uses some form of heuristic local search. For example, in the International Timetabling Competition 2002[4] (organised by the Metaheuristics network[5] and PATAT 2002[6]) the top four solvers used: 1) simulated annealing [23] 2) tabu search [11] 3) Dueck's Great Deluge [15] in [7, 16], 4) Tabu search with shakes [19]. Also, in a comparison of performances of different metaheuristics [32] it is noteworthy that the authors restricted themselves to local search "in an evolutionary algorithm, and ant colony optimization algorithm, and an iterated local search. A simulated annealing, and a tabu search were restricted to the same neighbourhood structure". Hence, local search is the favoured method for improving feasible solutions. In our case, because we do not have the hard constraint that all events must be allocated a time and place, constructing an initial feasible solution is trivial. Hence, our algorithm "only" needs to do improvement, and hence will do so using local search (with simulated annealing).

Another difference of our work from existing course timetabling work lies in the focus and methodology for using the problem instances and optimization algorithms. Typically, timetabling research focuses on a small number of instances, and attempts to obtain excellent solutions with the intention of using the entire solution. In contrast, we take a large number of problem instances, derive reasonable solutions, and then take

[4] http://www.idsia.ch/Files/ttcomp2002/
[5] http://www.metaheuristics.net/
[6] http://www.asap.cs.nott.ac.uk/patat/patat02/patat02.shtml

only "aggregate" properties such as utilisation and frequency and discard the rest. We then look at how these aggregate properties change as we manipulate the overall resources, sizes, and other aggregate properties of the problem instances. Generally we only need a reasonable solver as improving the solver will have minimal difference on the patterns we are studying. We expect it is more important for our solver to be robust, in the sense of consistency between instances, than for it to be particularly well-performing on some.

It should be emphasised that we use problem instances based on real data for courses and rooms obtained from a University in Sydney Australia, and so we expect our methods, and the broad picture of our results, are likely to be applicable to other institutions.

Outline of the Paper Section 2 covers the basics of the problem: the terminology and algorithms used. For example, we will see that maximisation of utilisation alone is a straightforward optimization problem often reducing to maximum weight bipartite matching. Section 3 covers the algorithms, and the data instances. Section 4 displays the threshold phenomena, and introduces the question of when a request for a specific amount of utilisation of frequency is likely to be satisfied; which we will call safe vs. unsafe requests. Section 5 introduces specific location and timetable penalties. Section 6 presents the Pareto fronts, or multi-objective trade-off surfaces, for utilisation, location and timetabling objectives. Section 7 returns to the issue of safe or unsafe requests, but this time in the presence of timetabling constraints. Section 8 covers safety in the presence of location constraints.

2 Background and Basics

In this section we cover the basic background needed for the main results. We describe the terminology of the domain, the constraints and the objective functions that measure the space usage.

2.1 Basic Terminology and the Hard Constraints

For each teaching room, assume that we are given:

1. capacity : the maximum number of students that the room can accommodate
2. timeslots : the number of timeslots for which the room is available during the week (or other relevant scheduling time period)
3. department : the department that "owns," or is most closely associated with, the room

An "event" requires the following information:

1. students : the number of students that must be accommodated
2. department : the owning/associated department

The primary task is to assign events to rooms so as to satisfy the following hard constraints:

1. room capacity: the size of an event (students) must not exceed the room capacity
2. the number of events allocated to a room must not exceed the number of timeslots, as events cannot share room timeslots.

Notice that there is no constraint saying that an event/course must be allocated to some room (in contrast, such a constraint is usual within timetabling), instead it can be part of the problem to find a set of events that are to be allocated. Also, at this

level, all the timeslots are indistinguishable; it does not matter exactly which timeslot an event actually gets (again, unlike timetabling).

We believe that this model captures enough of the real world for the purposes of this research study (our belief is based on our dealing with universities through Realtime Solutions Ltd). A more complete model would include other effects such as spacetypes and splitting. Rooms often have a "spacetype" that gives their intended usage: lecture, seminar, workshop, etc, and a fuller model would allow the mixing of spacetypes. Courses are typically not single "atomic" events, but instead might need multiple timeslots. Also, courses can need splitting into smaller events, called sections, because they are too large for the rooms or there is a recommended section size. We study this "splitting problem" in [3].

2.2 Quantifying Space Usage

The simplest and most direct measure of the space usage is to take the sum over all timeslots and rooms of the number of students allocated to that room-slot, which we will refer to as 'seat-hours' (though of course there is no implication that the timeslots really need to be an hour long). The intent is that "Utilisation" measures the fraction (typically expressed as percentages) of the total available (or maximum) seat-hours that are actually used:

$$\text{Utilisation} = \frac{\text{achieved seat-hours}}{\text{available seat-hours}} \tag{1}$$

Let C_i be the capacity of room i, and $S_{i,t}$ the number of students allocated to room i at timeslot t. Then the total number of seat-hours (denoted by B) is

$$B = \sum_{i,t} S_{i,t} \tag{2}$$

Since we enforce

$$S_{i,t} \leq C_i \quad \text{for all } i,\, t \tag{3}$$

we then have $B \leq B_M$, where B_M is the maximum number of seat-hours and is simply defined as follows

$$B_M := \sum_{i,t} C_i \tag{4}$$

Generally, the utilisation is defined by means of "occupancies" and "frequencies" [24, 25]. The occupancy $O_{i,t}$ of room i at time t is the fractional usage at that time

$$O_{i,t} = \frac{S_{i,t}}{C_i} \tag{5}$$

The occupancy, O_i, of a room, i, is defined as the mean of its occupancies over all *occupied* timeslots. Suppose that for room i the number of timeslots is t_i and the number of *occupied* timeslots is t_i^{occ}. Then the occupancy for room i is defined as

$$O_i := \frac{1}{t_i^{occ}} \sum_t \; O_{i,t} \tag{6}$$

The frequency usage, F_i, for a given room, i, is defined as the fraction of its timeslots to which some event is assigned:

$$F_i := \frac{t_i^{occ}}{t_i} \tag{7}$$

The utilisation, U_i, of room i, is the product of its occupancy and frequency:

$$U_i := O_i F_i \tag{8}$$

and so

$$U_i = \frac{\sum_t S_{i,t}}{\sum_t C_i} \tag{9}$$

that is, U_i is simply the fraction of the room's seat-hours potential that is actually used. However, to obtain an overall utilisation we will need to combine the utilisations from different rooms. We will take a weighted mean over the rooms

$$U^W := \frac{\sum_i W_i U_i}{\sum_i W_i} \tag{10}$$

where W_i is the weight assigned to room i. Usually, one just finds an unweighted mean U^{UW} corresponding to the special case that $W_i = 1$. However, a natural and simple choice is that larger rooms have a larger weight; and so we take the weight to be the room capacity, $W_i = C_i$. In this case, straightforward manipulation yields

$$U = \frac{\sum_{i,t} S_{i,t}}{\sum_{i,t} C_i} = \frac{B}{B_M} \tag{11}$$

which is just the promised overall fractional usage of the seat-hours. In our view, U as defined in 11 is preferable, as it is conceptually simpler than U^{UW}, at least as good a measure, and that the practical differences will generally only be a secondary effect. (In some experiments not presented here, we looked at both U and U^{UW} and found them to be tightly correlated anyway). We will also measure the overall frequency F of a solution,

$$\mathrm{F} = \frac{\text{timeslots used}}{\text{timeslots available}} \tag{12}$$

We do not weight frequency by the size of rooms, because we want a measure that is direct and simple to understand, and also because F is a "counting measure" that ought not in itself take account of room sizes. Again, in any case, we would expect other frequency measures to give similar results.

3 Optimization Algorithms and Data

This paper focuses on the nature of the space of solutions rather than on the algorithms that we employ. However, for completeness, we briefly describe them: firstly, we use mathematical programming to exploit cases that reduce to a max-weight matching problem, and secondly we employ a local search algorithm.

3.1 Mathematical Programming Methods

Suppose we call each (room,timeslot) pair a "room-slot", then the event allocation problem is to assign events to room-slots, and to maximise the allocated seat-hours. In the absence of other constraints or objective functions, it is well-known that this is just a standard assignment problem, and reduces to a maximum weight matching problem in a bipartite graph (see for example [12]). The events are taken to correspond to one set of nodes in the graph, and room-slots to the other set. The edges are the set of possible assignments of events to room-slots for which the capacity is sufficient. The weight, or value, of an edge is the contribution of the assignment to the total seat-hours, hence, simply, the number of students in the event. We are forced to have a bipartite matching because events can be assigned to at most one room-slot, and each room-slot can have at most one event allocated to it. Max weight matching has polynomial time complexity using the standard network flow algorithms. The simple optimisation of utilisation for event-based assignment is not a hard problem. For simplicity, we instead

exploit this by converting the assignment problem to a mathematical programming formulation. We encode it as an (binary) integer programming problem (see [27], or see [5] for a recent brief introduction to mathematical programming), but then relax to a linear programming (LP) problem, and will still expect to obtain integer solutions, which are hence optimal for the integer program as well. We use use this to derive optimal solutions when appropriate. This is used for checking that the local search is working well.

In some cases, the problem reduces to the assignment problem but with just an extra constraint which means that the solutions from the LP are not necessarily integer. We exploit a "rounding" method as follows. We solve the problem as an LP and extract the integer parts of the solution. The integer parts are then added as constraints to the original problem, which is then generally small enough to be solved as an IP. That is, we take the variables that are set to 0/1 in the LP, but leave the fractionally valued ones to be determined by integer program.

3.2 The Local Search Algorithm

Local search is performed on solutions in which some events are allocated to room-slots and others are unallocated. Operators are used that maintain feasibility (do not break the hard constraints such as capacity), and are as follows

1-OPT-swap-rand: Select 2 different rooms at random, and from each room randomly select an allocated event. If it maintains feasibility, then swap the two events between the room-slots.

2-OPT-swap-rand: Similar to *1-OPT-swap-rand* except select 4 rather than 2 events and swap them while maintaining feasibility of the given solution.

Move-exterior-rand: Randomly selects an allocated and an unallocated event. If it maintains feasibility, then the allocated event is deallocated, and the previously unallocated one given its room-slot.

Push-rand: Randomly select one unallocated event and one room. Try to allocate the event to the room; selecting the timeslot at random from those (if any) that would maintain feasibility.

Pop-rand: Randomly select one event from a randomly selected room and deallocate it.

Move-inner: Swap the timeslots of two randomly selected events in a single randomly selected room.

The operators use random sampling because the underlying neighborhoods tend to be quadratic in the number of events and too large to be searched completely.

The search itself performed with either standard Hill-Climbing (HC) or simulated annealing (SA) [22]. Each move operator is assigned a static probability for selection. On each iteration, we first select an operator according to their probabilities. Multiple, but limited (we use 10), attempts are then made to apply the operator in order to generate an improving candidate move. An iteration ends when a move is accepted or a pre-defined number of failures is reached. In simulated annealing worsening moves can be accepted depending in standard fashion on the temperature. In experiments we use standard geometric cooling with reheats.

When possible, the local search was compared against optimal solutions derived from the LP solver. When this was not possible we would compare standard runs of the simulated annealing against runs using much slower cooling, and many more reheats. Our standard run using simulated annealing is 4 coolings, 4m iterations each, cooling by a factor of 0.998 each 650 iterations, taking 20-60 minutes, and is chosen so that the search seems to become static become the end of each cooling. We take the best result obtained at the end of each cooling, and did find that the multiple reheats do help. We have checked that even with much longer runs (10 coolings, 15m iterations

each, cooling of .9995 each 650 iterations, taking up to 3 hours) the graphs presented do not change significantly. This gives us confidence that the results presented here are a good reflection of the underlying properties of the solution spaces, and have not been biased by the search methods.

3.3 Problem Instances

The real data set that we use arises from the "Appleby" building of a university in Sydney, Australia. The data contains many different space types; lectures, workshops, seminars, etc. However, for the purposes of this paper we are not covering the issues of splitting and also seek clarity, and so we select only the lectures (and also eliminate one lecture that is so large that it would need splitting). We have 20 rooms, and each has 50 timeslots. This gives a total of 1000 timeslots, whereas the lecture courses only have 608 events. Also, the total seat-hours demand from the lecture courses is 69983 whereas the total supply from the rooms is 202650. Hence, in the initial data set, the lectures are substantially under-subscribed, in the sense that the total demand for seat-hours and timeslots from the courses is much smaller than the supply of seat-hours and timeslots from the rooms.

In order to explore a wider range of these supply-to-demand ratio we need to do one or more of (i) add more courses, (ii) reduce the number of rooms, or (iii) have fewer timeslots per room. We opt against creating more courses, as it would make the problems unnecessarily large. The options of reducing rooms or reducing timeslots are similar in that they reduce the available seat-hours. Eliminating rooms requires a decision of which ones to remove, and it is hard to know what counts as a fair reduction, especially as we suspect that it is the distribution of room and course sizes that is the most important, and so do not want to change it accidentally (and this is also why we do not attempt to use a random generator for instances). So instead we uniformly reduce timeslots for all rooms. Hence, we create "Lecture Room" problem instances, LR(T), with the timeslots per room reduced to T. In the original data T=50, but we will also study T=10,18, and 30. The case T=18 is the smallest T in which the seat-hours demand could potentially be still be met by the rooms.

We have now covered the basic terminology, algorithms and data sets used; and so can commence the main results.

4 The Safety of Utilisation and Frequency Requests

Suppose that, we are carrying out space planning, and have a proposal for a set of rooms and a reliable forecast for the expected demand for total seat-hours from courses. We would like to know whether we can be confident that it will be possible to satisfy the demand, but we do not yet know exactly how the demand for seat hours will break down into actual courses. Instead we just expect that the demand will arise from a subset of some much larger set of expected courses. Given the set of rooms and so the supply of seat-hours, then the expected seat-hours demand can be converted to a "requested" utilization, U_R. In the absence of low utilisations, then we could be confident that as long as $U_R \leq 100\%$ then we would be able to satisfy all the demand; that is, the "achieved" utilisation, U_A would equal U_R; but maybe this is no longer true when U is expected to be low? Hence, in this section we build towards answering the question

"Under what conditions is a request for utilisation fully satisfiable?"

We will work in terms of "Achievement Curves". These will represent the quality (either U or F) of valid solutions in terms of the quality requested, U_R or F_R, that is the

quality that would have been achieved if the entire request could have been satisfied, and compare to the quality, U_A or F_A, of the Achievable solutions

We find achievement curves using the following procedure:

> for each value of probability $p \in [0.01, 0.02, \ldots, 1.0]$
>
> S := a random subset of the courses; taking each course independently with probability p.
>
> 1. sum the sizes of events in S to find the total requested seat-hours, B_R.
> 2. optimise the utilisation for S (using an appropriate solver) to determine the achieved seat hours, B_A

We repeat this many times to generate more points.

The requested and achieved values for seat-hours are converted into achieved and requested Utilisations U_R and U_A. We also measure the total number of requested and achieved events to give the requested frequency (F_R) and achieved frequency (F_A). Thus, each fixed, but randomly generated, subset S, generates data points (U_A, U_R) and (F_A, F_R). Note that we can request a U (as total seat-hours), or an F (as total timeslots), but it does not seem to be useful to talk about a "Requested occupancy". Although we have three measures, U, F and O, only two of them are independent, as they are related by "$U = FO$". It seems simplest and clearest to select the two independent measures to be U and F.

Figure 1(a) presents the results of following the above procedure for the room data LR(10). We find that as well as plotting U_A vs. U_R it is also helpful to plot the "Fractional Achievement Ratio"; that is the fractions of the requested U or F that turn out to be achievable. The results in terms of fractional achievement are given in Figure 1(b).

The first, and crucial result, is that the values of achieved U and F, for given corresponding requests, tend to be "grouped around the mean". That is, the variation of U_A between points near to some value of U_R is small compared to the value of U_A itself, and similarly for F_A. This implies that properties of the system are statistically predictable. In our case, take for example, $U_R = 80\%$ then the mean value for the achieved $U_A \approx 70\%$ but the variation between instances in that region of the curve is relatively small. This is crucially important, because if the variation were very large we would not be able to make reliable and fairly tight predictions of the achievable utilisation or frequency.

The second, and also crucial result from figure 1, is that we see a threshold phenomenon on U. There is a "critical value", U_C, for the requested utilisation, U_R. In this case, $U_C \approx 60\%$, and this value divides the results into two distinct regions:

> **SAFE**: $U_R < U_C$. Requests for seat-hours are almost always totally satisfied.
> **UNSAFE**: $U_R > U_C$. Requests for seat-hours are almost never totally satisfied. Even in the cases when there are enough seat-hours available, it turns out that the oversupply is actually unusable.

(Though with a narrow region close around U_C in which the situation is less predictable.)

This has important implications. When planning course offerings we cannot assume that we can simply count seat-hours, but must account that we are unlikely to be able to rely upon using more than 55% of the seats available (in this case). But for $U_R > U_C$ we will (almost) inevitably find that some of the events will need to be dropped.

We refer to a request for U as "safe" when statistically there is high probability (for example, with better than 95% chance) that it will be possible to satisfy all of the request. We use the terms "safe" to convey that it means low-risk but not an absolute guarantee.

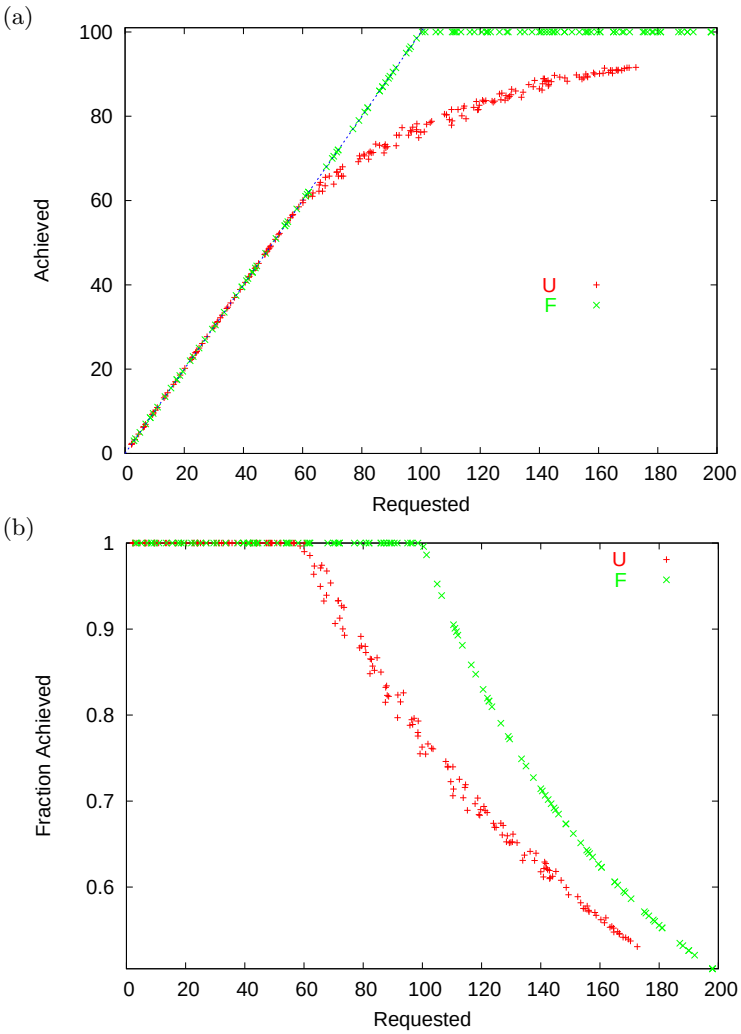

Fig. 1. (a) Requested vs. Achieved percentages for U and F, for random subsets of the courses, and with the rooms LR(10). The diagonal line, 'Achieved=Requested', is given for reference purposes. (b) Same data but for the "fractional achievement", that is, the y-axis is U_A/U_R or F_A/F_R.

Note that, in this case, the frequency is always maximally satisfiable. Obviously requests $F_R > 100\%$ are unachievable, but here all requests $F_R \leq 100\%$ are safe. In this case, it turns out that the frequency seems to be the limiting factor. Even if $F_R = 100\%$ and all of the events are allocated, $F_A = 100\%$, and all the timeslots are occupied then the overall utilisation is only around 50-60% because most of the rooms are not fully filled. However, in general, maybe there are different regimes according to the most important limiting resource, whether it be seats, or timeslots, or something else.

The other interesting points on Figure 1(a) are the endpoints at the largest values of requested U and F. These correspond to taking all the courses, but allowing the solver to make a free choice of which events are going to be allocated an room-slot. This motivates us to introduce the terminology:

Fixed-Choice Mode. Given a fixed set of courses, the solver is not given freedom to select those that should be allocated or not, but instead we want to know if we can allocate them *all*.
Free-Choice Mode. The solver is allowed a free choice of which courses to allocate when it is optimising U

The free-choice mode allows the solver to pick events that are better suited to the room sizes, and so to increase U. From Figure 1(a), for this data set and no extra constraints, with free choice we can reach $U \approx 92\%$ but with the "fixed choice" we are only safe up to $U \approx 57\%$. The real situation would probably be somewhere in between these two extremes. If courses were selected from the pool with no regard to overall utilisation, then the safe choice would be limited to $U < 57\%$. However, in practice, there probably is an effect (that accrues from term-to-term) that the sizes of the courses will evolve towards being a better fit to the rooms. So, arguably, a natural evolution might push us a little above the safe point. However, it seems unlikely that such natural evolution would be so strong as to achieve the highest ends of the utilisation values.

The "grouping about the mean" and thresholds observed here are fairly common properties of problem classes in which instances are selected from a large set of possibilities (It is important to remember that the number of subsets is exponentially large: with n courses there are 2^n possible subsets.) The phenomena is analogous to that of phase transitions in physical systems (such as water into ice), and in computer science is best known in the context of random graphs [4]. For example, a standard distribution for random graphs is to take n nodes and add every potential edge independently with probability p. In this case, many properties of the graphs become statistically predictable from the values of n and p, and boolean properties, such as "the graph is connected", will exhibit a threshold at some critical values of p. Another example is that the chromatic number for random graphs is statistically predictable for such random graphs.

Practical Usage of the Critical Value The intention is that such results can be used in order to generate a set of rooms with the appropriate safety margins for space planning. However, one might be tempted to adopt the following method: pick a "just-fit" set of rooms for which the requested utilisation would be 100%, find its critical value of U_C, and use $1/U_C$ as a safety margin. Although probably fine in practice, this would not be strictly correct as the scaling of U_C with the problem size is not necessarily linear. Instead, one should pick a "safe-fit" set of rooms for which the requested U would be just U_C.

5 Introducing Location and Timetable Objectives

In the previous section we obtained the safety requirements $F \leq 100\%$ and $U < 57\%$. However, even this value for U is still unrealistically high. We suspect that,

in practice, the need to take account of other objectives and constraints will drive down the achievable U and F. Hence, in this section we introduce specific "location" and "timetabling" penalties in order to investigate the effects of physical location and timetabling requirements.

Location Penalty (L) The intention of the location penalty is to model how well the rooms match the events allocated to them in terms of physical location. Here, this means that the allocation of an event E to a room R should receive a penalty depending on the perceived distance between E and R; the distance between the department owning the room, and the department owning the event allocated to it. (We will treat minimisation of the L penalty as an objective; or specifically will be maximising $-L$.) In the absence of any data for physical distances we have arbitrarily selected a set of penalties based purely on the departments involved. An event E is owned by a department, and this is assumed to be its natural home. The L penalty is non-zero only if the event is owned by a department different from that which owns the room. Basically, this just encourages events to be placed within their owning department. (Such location penalties are also likely to be fairly natural even if the allocation decisions are made be a central administration: lecturers and students will generally prefer to remain close to their "home" department.)

Note such a location matching is a common desire within course timetabling. For example, after their conclusion that "the university is not using all of its classroom space as efficiently as it might", Fizzano and Swanson [17] continue: "Our results do not guarantee that there are practical schedules that use the number of classrooms we determined because our process does not consider things like teachers' room preferences or class location requirements (English classes might not end up near the English department)."

Notice that the penalty depends only on the room and event, and so if we take a linear combination with utilisation it can also be modelled within the framework of maximum weight bipartite matching; the L penalty and seat-hours score together generate a new set of edge weights. However, if L and U are treated as independent objectives we end up with a bi-criteria matching problem which is harder due to the presence of unsupported solutions [35, 36].

Timetable Penalty (TT) In order to take some account of the effects of timetabling we introduce a conflict graph between events. Again, in the absence of data we have used various simple randomised generators for the timetable conflict matrix. Our model is again based on the owning department for the each event. Specifically, we generate conflict matrices using recipes denoted by "TT(p, q)", and according to

1. conflicts between events from the same department are generated randomly with probability p
2. conflicts between events from different department are generated randomly with probability q

The case $p = q$ corresponds to ignoring the department, that is, a standard random graph [4], and will refer to this as simply TT(p). Another simple case is TT$(100, 0)$: the conflict graph that has edges between any two events in the same department but none otherwise. This corresponds to expecting that events from the same department are more likely to have students in common, or simply that departments strongly prefer that their own events do not clash. We expect that the timetabling constraints of this form will capture some of the broad effects of real problems. A similar structure was used in [31] where they had a "probability p_w, for within-clump constraints, or p_b for between-clump constraints". Their motivation was that "Real timetabling problems are typically clumped rather more clumped than homogeneous. For example, exams

within an arts faculty may typically form a distinct clump, largely separate from those within a science faculty."

Note that the departmental owner of an event is fixed, and is not necessarily the same as its allocated location, though in cases of a mismatch the assignment would generate a location penalty as explained earlier.

6 Multi-Objective Optimisation in the Free-Choice Mode

Adding L or TT penalties to a problem cannot increase utilisation; instead it is likely that they will drive it down. However, the issue is the magnitude of such an effect, and in particular whether the effects can be large enough to be responsible for the low values seen in practice.

In this section, we are interested in reducing the upper estimates on utilisation, and will use the free choice mode: take all the events, and allow the solver to select those that will be allocated. We will treat the system as a multi-objective problem using the utilisations, and (the negatives of) the location penalty L, and timetable penalty, TT, and determine the appropriate (approximate) Pareto fronts (see [35, 14] for descriptions of the concepts of Pareto optimisation). Let's first consider the simpler two-objective sub-cases.

6.1 Bi-objective Problems: U vs. L and U vs. TT

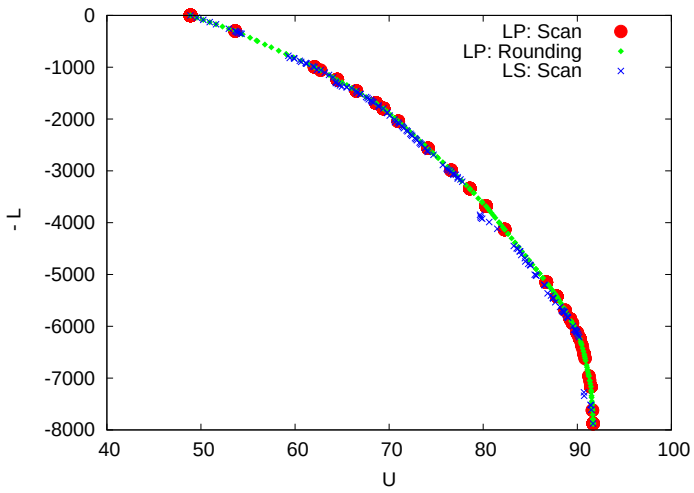

Fig. 2. L vs. U for LR(10). LP-Scan and LP-Rounding use the LP solver. LS uses the local search method.

To find trade-off surfaces, or (approximate) Pareto fronts, we follow the standard procedure of generating points by taking many possible linearisations of the problem; data points are generated by running the solver(s) for for each of many different choices for the relative weights of objectives. Figure 2 shows the results of such an exploration. Note that in fact we plot $-L$ rather than the location penalty L itself, merely so that we meet the convention that all axes correspond to maximisation problems, i.e., "better" means towards to the top right. Also, for clarity, in all such plots we remove all Pareto dominated points.

The first set of points are obtained by taking a linear combination of the weights, solving using linear programming, as discussed in subsection 3.1. Each such solution is hence Pareto Optimal. However, some Pareto optimal solutions might not be reachable in this way (that is, "unsupported" solutions [35, 36]), and this leaves some gaps in the experimental Pareto Front. To remedy this we also generate a second set of integer solutions using the rounding method of subsection 3.1. In practice, the number of non-integer assignments is very small and so the IP problem is then very small (5-10 variables, instead of thousands). As is evident from the figure, the difference between the LP and the rounding is very small. This indicates that the underlying problem is rather easy in this case. It was also observed in very early days of the office space allocation problem [30] that the IP formulations can result in very few non-integer variables, and so the problems are relatively tractable for their size. The final set of points in Figure 2 are obtained using our local search method (simulated annealing). It gives points that are also very close to optimal – the difference being small enough so as to not significantly change the shape of the curve. This gives us confidence that our local search method will not be distorting later results.

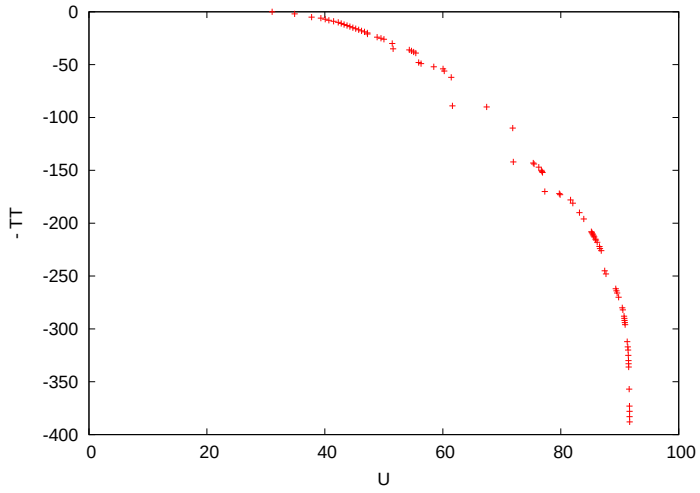

Fig. 3. The trade-off between Utilisation, U, and Timetable Objectives, -TT, for the rooms LR(10).

The primary result from Figure 2 is that incorporating the location objective can sharply reduce the utilisation: driving it down to about 50% from 93%.

We also consider utilisation and timetabling alone. This time we do use an exact method, but just the local search. Figure 3 shows the resulting trade-off curve when using TT(100,0). The introduction of the timetabling objective can drastically reduce utilisation, this time to about 32%.

6.2 Three-objective Problems: U vs. L vs. TT

For the combination of three objectives, we fix the weight for the U objective and produce points by scanning many different weights for L and TT (again meaning TT(100,0)), and optimising each one separately. The results of this for the rooms LR(10) are given in Figure 4. As explained in the caption, the 3 dimensional data is converted to 2-dimensional by giving U and -L to the x and y axes. The timetable objective is represented by grouping points into different sets according to their TT

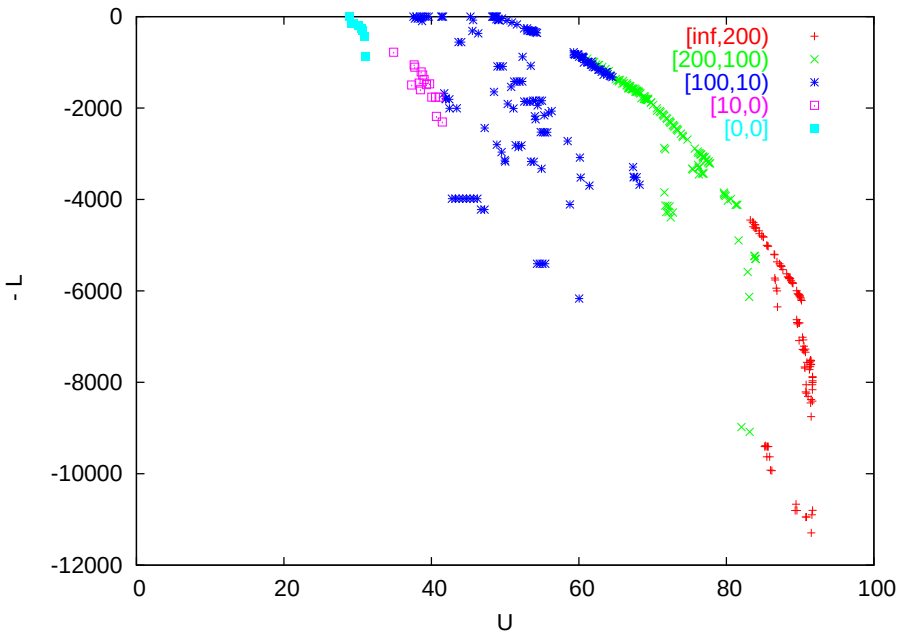

Fig. 4. U vs. L vs. TT for LR(10). The x and y axes are utilisation, U, and the location objective, -L, respectively. The third dimension of TT is represented by selecting the colour (or type) of the point based on the TT penalty as given in the key, e.g. $[200, 100)$ means $200 \geq TT > 100$, and `inf` means infinity. Only Pareto non-dominated points are included.

penalty, and plotting using different point types (and colours). Sets further down the key correspond to being better for the TT. We see that the maximal U is 92% but this is reduced to 29% if the L and TT penalties are forced down to zero.

Figure 5 gives results for LR(18) and LR(50). Notice that as we increase the numbers of timeslots, then utilisations decrease, but this is merely because the seat hours needed by the courses is less than the available seat-hours as soon as timeslots is greater than 18.

Notice that that L and TT are having a significant effect even for the case of LR(50). There are sufficient seats available that in the absence any L and TT constraints all events can be allocated, and this leads to the value of U=44% on the righthand edge. However, such complete allocations have large L and TT penalties. If the L and TT penalties are forced down then U is also forced down significantly. That is, even when the available seat-hours is twice that requested, L and TT can mean some events must remain unallocated.

These results support our hypothesis that the location and timetable penalties have the potential dramatically drive down utilisations, and so are a reasonable candidate to explain low utilisations in the real world.

7 Fixed Choice with Various Timetabling Models

In the previous section we found that the location and timetable requirements can significantly reduce utilisations within the free-choice mode. In this section we investigate the effect of timetabling within the fixed-choice mode. In particular, we look at the effects of timetabling on the safety of requests for utilisation and frequency. The basic procedure is the same as in Section 4, except that for each selected subset of the courses

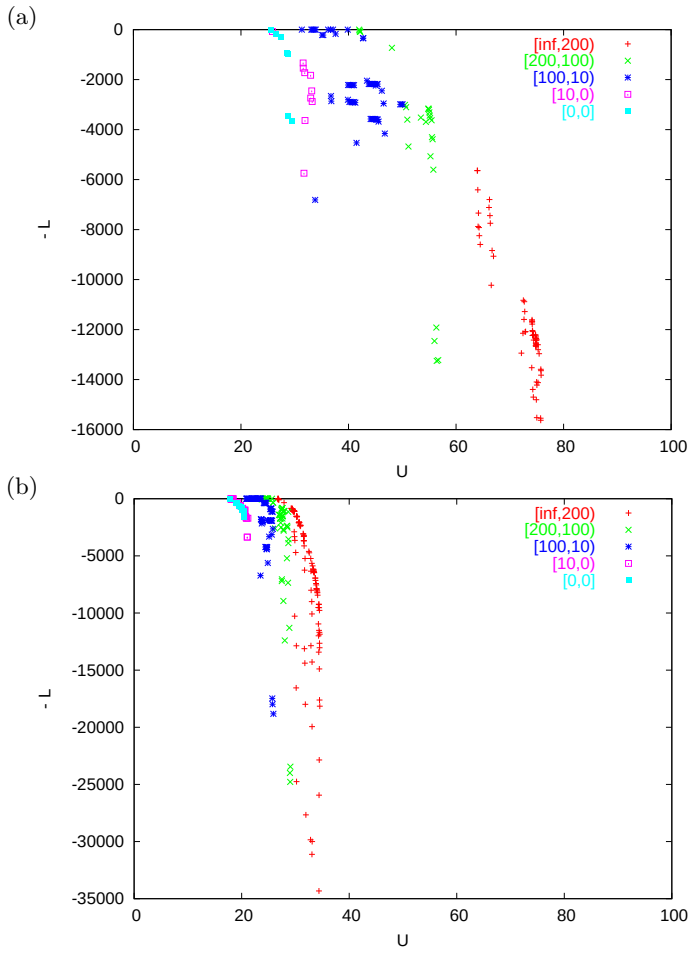

Fig. 5. The same as Figure 4 except for the room sets (a) LR(18), and (b) LR(50).

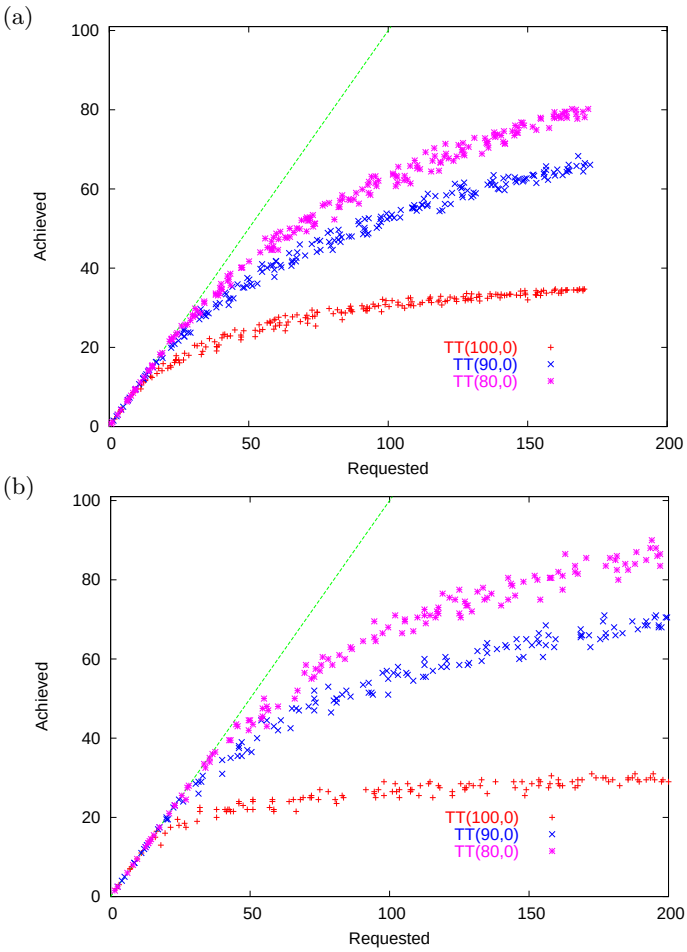

Fig. 6. Requested vs. Achieved: (a) U and (b) F. In presence of various hard timetabling constraints from TT(p,0) with p=100%,90% and 80% (and no L).

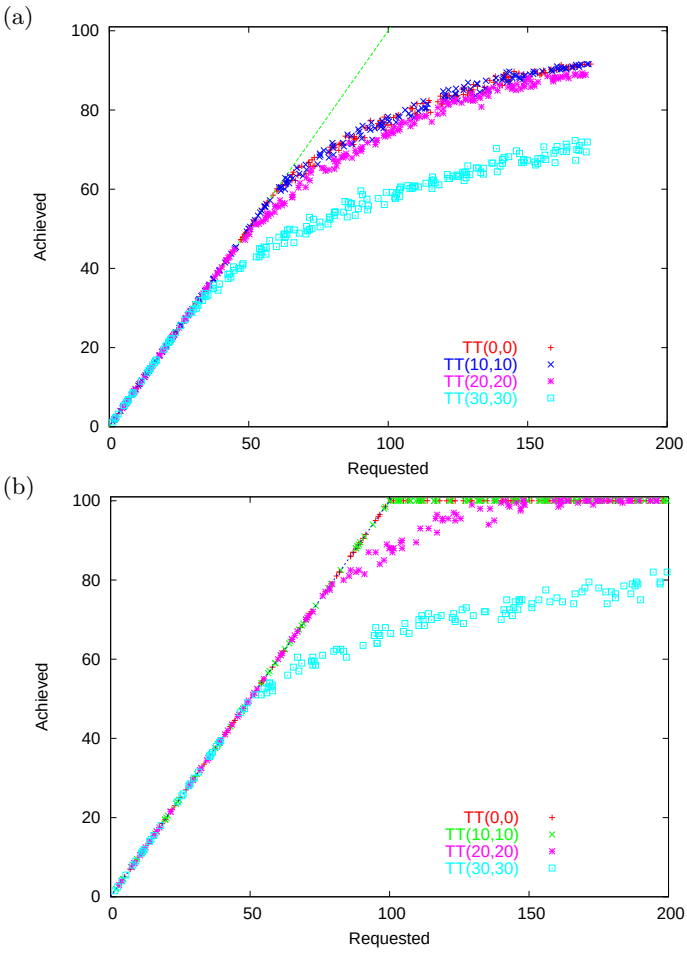

Fig. 7. Requested vs. Achieved: (a) U and (b) F. In presence of various TT with TT(p,p) conflicts with densities p=0%,10%,20% and 30% (and no L).

we also generate a corresponding random graph to be used as the conflict graph, and then solve treating the TT conflicts as hard constraints.

Figure 6 shows the effect on the achievement curves of enforcing hard TT(p,0) constraints.

On comparing with Figure 1, we see that the safe region for U is greatly reduced. More noteworthy, is that previously all $F \leq 100\%$ were safe, but this is no longer true. The hard timetabling penalty means that some timeslots remain unfilled. Also note that the difference between TT(100,0) and TT(90,0) is quite large, indicating that safety regions can be fairly sensitive to the details of the model.

Figure 7 shows the effects on the achievement curves of enforcing hard TT(p)=TT(p,p) constraints; the conflicts are independent of the owning departments of the events. Perhaps most notable from Figure 7(b) is that conflict densities of 10% do not lower the safe region. It appears that the safe region for F only starts to become reduced when the conflict density reaches about 15%. Apparently, in some cases the timetable conflicts have no effect on the safe regions until the conflict density exceeds some critical level. Presumably, the details will depend on the particular problem instance; but it does seem that safe regions may well be insensitive to the imposition of "small" amounts of other objectives.

8 Fixed-Choice Together With Both Hard L and TT

In this final section of results, we briefly study the effects of demanding that the assignments totally match the location; that is, we treat location as a hard constraint ($L = 0$). This might correspond to an institution with very localised control of rooms or no sharing between departments.

Figure 8 shows the achievement curves with a hard L constraint, and a hard timetabling, TT(90,0), constraint as well. Again the achieved U and F are statistically predictable in that they are "clustered about the mean". At first sight, it seems that the hard L constraint has merely reduced the safe regions. However, closer inspection of the achievement ratio, Figure 8(b), reveals that the effect is more extensive. There is no safe region in which we are almost always sure of satisfying all of the request. Instead we get a "weakly-safe" region, a region in which we are "only" almost always sure of satisfying a large fraction of the request. In particular, if we request any F less than about 40% then we are very likely to be able to satisfy about 95-98% of the request, but not the 100% we would expect in a safe region. We suspect that this indicates that there is a mismatch between rooms and events, and will explore such cases in future work.

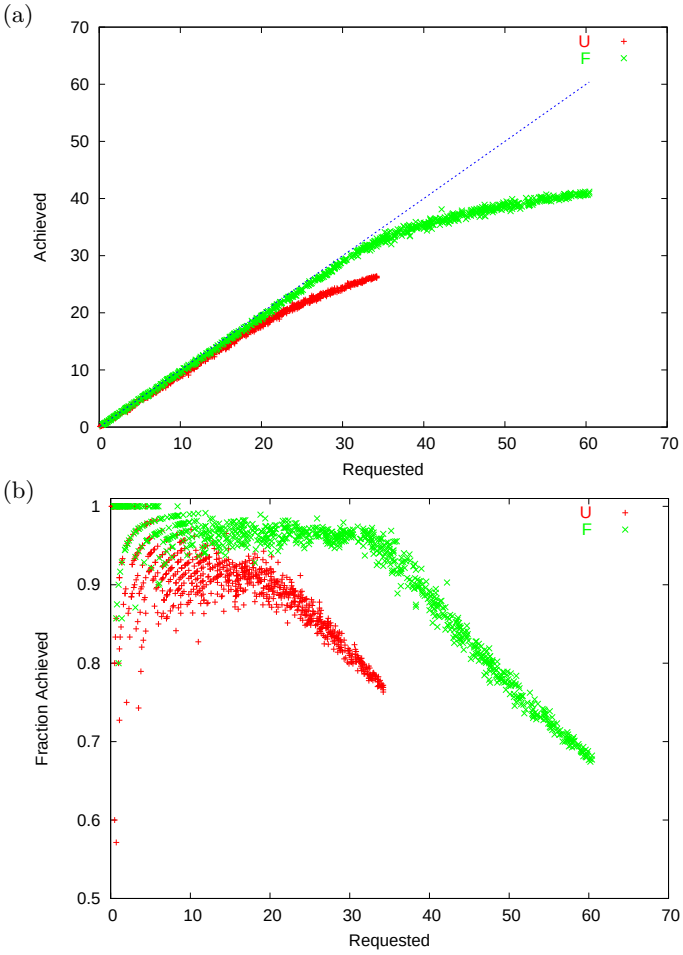

Fig. 8. Requested vs. Achieved F and U. With Hard L constraint. LR(50). TT(90,0). (a) Achieved (b) Fraction Achieved

9 Related Work

Room minimisation As mentioned in the introduction, Fizzano and Swanson [17] also studied space usage within a university. They do room minimisation by the simple procedure of removing rooms whenever doing so would not results in the problem no longer being solved by their algorithm. After removing unneeded rooms the overall frequency increases correspondingly. For example, in one case the number of rooms could be reduced from 51 to 38, and then the frequency increases from about 60% to about 83%, but note that it still does not become 100%. (Unfortunately, they do not give information on sizes of rooms and classes and so we were unable to infer the corresponding occupancies and utilisations.) However, they do not perform our multi-objective or phase transition studies. Also, the constraints used within are substantially different from the majority of the course timetabling literature. Their time restrictions arise because courses consisting of multiple events must take place on some specified subset of the days of the week and with the restriction that the course events must take place as the same time of day. That is, they have links between events that say they must be at the same time of day, but they do not have the usual "conflict matrix" of pairwise constraints that events must take place at different times. This changes their underlying methods from be variants of graph colouring to being variants of graph matching. Also, as mentioned in Section 5, they do not include the effects of location objectives. The difference in their constraints means that direct comparison is not meaningful.

Computational Hardness and Phase Transitions Phase transitions in the area of course timetabling were studied in [31]. However, this was from the point of view of the computational hardness rather than the positions of the phase transition (as a function of the controlling parameters such as conflict density). Many systems have a threshold, and it is well-known that the computationally difficult decision problems, "hard problems", typically occur at the threshold; that is, at the phase transition between "almost always yes" and "almost always no" regions [10]. Possibly one of the most well-studied thresholds, and associated hard problems, is the satisfiability transition in the "Random 3SAT" domain [26]. (In a study, [28], of the scaling properties of a local search near the satisfiability threshold it was found that scaling of average run-times was polynomial until getting very close to the phase transition itself.) However, in [31] the instances themselves are generated in such a fashion as to guarantee that they solvable; by construction, there cannot be a satisfiability threshold. Their focus is instead on the hardness of the instances for a solver, and they do indeed observe transitions in hardness. We differ in that we are not investigating the hardness but rather the actual satisfiability of a request for a particular overall utilisation.

In our case, although we have not studied it in detail as yet, it is reasonable to expect that there will be a peak in hardness for the satisfiability decision problem near the transition from safe to unsafe utilisation requests. Exact location of the transition point could be difficult. However, conversely, since we can expect the hardness to drop rapidly as move away from the threshold, it makes it likely that a good approximation of the threshold location is computationally relatively easy. In the random 3-SAT work, the solvers have improved by many orders of magnitude over the decade it has been studied, but the estimate of the threshold location has changed very little. This gives us confidence that any inadequacy of our solver is unlikely to have a significant effect on our results.

10 Summary and Conclusions

The issues of space allocation, space management and space planning, are crucially important to universities in general, and to Realtime Solutions Ltd in particular. Ac-

cordingly, we have studied teaching space allocation with two goals in mind. Firstly, we aimed to understand the factors that have the potential to explain the low utilisations and frequencies observed in real world institutions. Secondly, a requirement was to start to devise methods to determine the safety margins that must be included within space planning methods in order to compensate for the expected low utilisation figures when space is actually put into use.

In preliminary studies, it was clear that considering utilisation alone gives unrealistic results, in the sense that the realized utilization was too close to the maximum possible utilisation. However, if we also include objectives to mimic the effects of timetabling and physical location, and plot the resulting multi-objective trade-off surfaces, then in some regions the utilisation falls to much more realistic and observed levels, in the range of 20-40%.

Equally importantly, we find that when selecting courses at random from a pool then whether or not the selections are fully achievable ("safe") becomes statistically predictable. This means that the typical behaviour of different instances can be predicted. Also, the behaviour displays threshold phenomena: There is a critical value of requested utilisation below which there is a high probability of satisfying it all, but above which the probability drops sharply.

The intended usage of the results are (i) to build a better understanding of the factors that affect utilisation and frequency, a necessary first step to being able to improve them in practice, (ii) to use the statistical predictions of safe regions of U and F in order to give better, more cost-effective, safety margins to be used in space planning

Perhaps we should emphasize that, after removal of irrelevant event IDs and so forth, the input data needed for the results in this paper consists only of: a set of department IDs; for each a multi-set (a set that allows duplicate entries) of event sizes and a multi-set of room sizes; a location penalty matrix for [dept(event),dept(room)]; and a method to generate a conflict matrix over [event,event] pairs.

This work has provided an important foundation for a range of research issues that need to be explored. We emphasise that we are developing a methodology, and it is not the details of the algorithm or the exact detailed numbers in the results that are important. For example, we believe that the universality of threshold phenomena in large systems [4, 21, and others] will lead to wide applicability.

Perhaps, the most important follow-up is to find a better stand-in for timetabling, that is, (statistical) ways to characterize the timetabling problem in an institution, and to do this without relying on details that will not be available at the space planning stage. That is, possibly, the greatest challenge is to see whether useful statistical information can be produced without having to resort to full simulations. Simulations are conceptually straightforward, but often difficult in practice because of the lack of relevant data.

Other important aspects of future work will be to carry out a comprehensive series of comparisons against other real problems. Also, an important aspect of space planning is to determine how "room size profiles" – the distribution of room sizes – affect these results. Finally, we note that our current implementations are rather inefficient – the graphs here needed many thousands of hours of CPU time – and so we will be implementing more efficient methods to produce the trade-off surfaces and achievement vs. request curves, for example, to use the methods of [18] for finding Pareto fronts.

References

1. Pasquale Avella and Igor Vasil'ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8(6):497 – 514, 2005.

2. Victor A. Bardadym. Computer-aided school and university timetabling. In Edmund K. Burke and Michael A. Trick, editors, *Selected Papers from PATAT 95.*, volume 1153 of *Lecture Notes in Computer Science*, 1995.
3. Camille Beyrouthy, Edmund K. Burke, Barry McCollum, Paul McMullan, J. Dario Landa-Silva, and Andrew J. Parkes. The teaching space allocation problem with splitting. Submitted to PATAT 96.
4. B. Bollobas. *Random Graphs.* Academic Press, London, England, 1985.
5. Robert Bosch and Michael Trick. Integer programming. In E K Burke and G Kendall, editors, *Search Methodologies: Introductory Tutorial in Optimization and Decision Support Techniques*, chapter 3, pages 69–96. Springer, 2005.
6. E K Burke and S Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140:266–180, 2002.
7. Yuri Bykov. The description of the algorithm for international timetabling competition. http://www.idsia.ch/Files/ttcomp2002/bykov.pdf.
8. Michael W. Carter and Craig A. Tovey. When is the classroom assignment problem hard? *Operations Research*, 1992.
9. M.W. Carter and G. Laporte. Recent developments in practical course timetabling. In Edmund Burke and Peter Ross, editors, *Selected Papers from the Practice and Theory of Automated Timetabling IV.*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 1998.
10. Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia*, pages 331–337, 1991.
11. Jean-Francois Cordeau, Brigitte Jaumard, and Rodrigo Morales. Efficient timetabling solution with tabu search. http://www.idsia.ch/Files/ttcomp2002/jaumard.pdf.
12. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.
13. D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19:151–162, 1985.
14. Kalyanmoy Deb. Multi-objective optimization. In E K Burke and G Kendall, editors, *Search Methodologies: Introductory Tutorial in Optimization and Decision Support Techniques*, chapter 10, pages 273–316. Springer, 2005.
15. G Dueck. New optimisation heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:89–92, 1993.
16. E.K.Burke, Y.Bykov, J.P.Newall, and S.Petrovic. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, 13(2):139–151, 2003.
17. Perry Fizzano and Steven Swanson. Scheduling classes on a college campus. *Comput. Optim. Appl.*, 16(3):279–294, 2000.
18. X. Gandibleux, H. Morita, and N. Katoh. The supported solutions used as a genetic information in a population heuristics. In *Proceedings of the 1st international conference on evolutionary multi-criterion optimization (EMO 2001)*, volume 1993 of *Lecture notes in computer science.*, pages 429–442, 2001.
19. Luca Di Gaspero and Andrea Schaerf. Timetabling competiation ttcomp "002: Solver description. http://www.idsia.ch/Files/ttcomp2002/schaerf.pdf.
20. HEFCE. Estates management statistics project. Technical report, Higher Education Funding Council for England, March 1999. Report 99/18. http://www.hefce.ac.uk/pubs/hefce/1999/99_18.htm.
21. Bernardo A. Huberman and Tad Hogg. Phase transitions in artificial intelligence systems. *Artif. Intell.*, 33(2):155–171, 1987.
22. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
23. P A Kostuch. Timetabling competition - sa-based heuristic. http://www.idsia.ch/Files/ttcomp2002/kostuch.pdf.
24. B. McCollum and P. McMullan. The cornerstone of effective management and planning of space. Technical report, Realtime solutions, Jan 2004.
25. B. McCollum and T. Roche. Scenarios for allocation of space. Technical report, Realtime solutions, 2004.

26. David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions for SAT problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, Menlo Park, California, 1992. AAAI Press.
27. George L. Nemhauser and Laurence A. Wolsey. *Integer Programming and Combinatorial Optimization*. Wiley, New York, 1988.
28. Andrew J. Parkes. Scaling properties of pure random walk on random 3-sat. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes In Computer Science*, pages 708–713. Springer-Verlag, 2002.
29. S Petrovic and E K Burke. University timetabling. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 45. Chapman Hall/CRC Press, 2004.
30. Larry Ritzman, John Bradford, and Robert Jacobs. A multiple objective approach to space planning for academic facilities. *Management Science*, 25(9):895–906, Sep 1979.
31. Peter Ross and Dave Corne. The phase transition niche for evolutionary algorithms in timetabling. In *Proceedings of the First International Conference on the Theory and Practice of Automated Timetabling, Napier University, Edinburgh, 1995. Selected Papers.*, volume 1153 of *Lecture Notes in Computer Science*, pages 309–324, 1996.
32. O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, M. Dorigo, L. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, and T. Stützle. A comparison of the performance of different metaheuristics on the timetabling problem. In E. Burke and P. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2002*, volume 2740 of *Lecture Notes in Computer Science*, pages 329–351. Springer Verlag, 2003. Extended abstract available on the Proceedings of PATAT 2002, pages 115-119.
33. Andreas Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):18–27, 1999.
34. J.D. Landa Silva. *Metaheuristics and Multiobjective Approaches for Space Allocation*. PhD thesis, School of Computer Science and Information technology, University of Nottingham, Nov 2003.
35. Ralph E. Steuer. *Multiple criteria optimization: theory, computation and application.* Wiley, 1986.
36. D. Tuyttens, J. Teghem, Ph. Fortemps, and K. Van Nieuwenhuyze. Performance of the MOSA method for the bicriteria assignment problem. *Journal of Heuristics*, 6(3):295–310, 2000.