Staffing a Mathematics Support Service

Jonathan Gillard, Vincent Knight, Julie Vile and Robert Wilson
GillardJW@Cardiff.ac.uk
Cardiff University, Cardiff, UK

July 22, 2014

Abstract

We study the problem of staffing university mathematics support services in which students drop in to the service (without appointment) for tutoring support. Our approach seeks to find the minimum sufficient number of tutors (with appropriate specialities) to present by hour and day to cover student demand with tolerable delays. We employ traditional operational research techniques to aid managers and administrators of mathematics support services to roster their services. The machine interference type queueing is adopted to model the number of student queries within a mathematics support session. We define and solve an appropriate integer program to roster the number of tutors needed to run the service efficiently.

Keywords: machine interference model, mathematics support service, rostering

1 Introduction

Over the last decade the 'mathematics problem' (students lacking basic mathematical skills on entry into higher education), and proposed solutions of this problem have been debated in much detail [10]. One approach that has been developed to help combat this issue has been the introduction of mathematics support services (MSSs) across higher education institutions. Such services in higher education can be traced back to 1990 and it is thought that they were launched even earlier in further education institutions [14]. The resources provided by MSS's often vary in nature but typically revolve around some form of drop-in service, whereby students call in to the service (without appointment) to discuss their mathematical query with a tutor.

The MSS at Cardiff University (CU) is open each weekday between 11:00 - 13:00. Many students go to the service immediately after the completion of their previous lecture, leading to large influxes of students arriving at 11:00 and 12:00, with fewer arrivals throughout the remaining period. Figure 2 shows the average number of students present at the MSS for every hour of the week between October 2010 to June 2011.

One goal of this paper is to obtain minimal staffing levels that ensure that no more than 10% of students wait longer than 15 minutes before being seen by a tutor. When employing queueing theoretical methods to estimate the optimum number of staff to ensure that a given performance metric is satisfied, we consider employing a member of staff for the two 1 hour periods throughout the two-hour session (i.e. 11:00-12:00 and 12:00-13:00). Over a given week, we thus require the staffing constraints for 10 time periods.

There is a vast quantity of literature that make use of queueing models to obtain staffing constraints for various types of service centres (for example: emergency departments [6] and police patrols [12]). One novelty proposed in this paper is the use of a particular queueing model (a 'finite source queue') that is more often applied to the service of machinery than the service of individuals, to predict the amount of assistance students will require within a particular mathematics support drop-in session. Finally an integer program is defined; the solution of

which rosters the available tutors. In this integer program we maximise a linear cost function which maximises the amount of expertise available in different subject areas (statistics, pure mathematics and applied mathematics). A heuristic is offered that gives an efficient approach to solving the rostering problem for larger problem instances.

To summarise the contribution of the work presented:

- A novel application of the 'finite source queue' is demonstrated.
- An MSS is modelled using a particular queueing model to ensure sufficient levels of staffing.
- An integer program is developed that allows for an efficient rostering of a MSS, a heuristic to approximate the solution of this integer program is also discussed.
- The value of operational research to aid in the provision of mathematics support services is demonstrated.

The structure of the paper is as follows. The presentation of the queueing model is given in Section 2 and the mathematical program and heuristic are presented in Section 3. Conclusions are given in Section 4.

2 MSS as a finite source queue

2.1 The mathematical model

Most of the literature dealing with setting staffing constraints assume the service under analysis to be an $M/M/c$ queue. A good review of approximation approaches that are relevant under this assumption is available in [5]. For reasons that become apparent in Section 2.2 we choose to use a 'finite source queue' (also known as a 'machine interference model') to model an MSS. This model is often used to represent situations where a finite number of machines run. Such machines run until breaking down, and consequently require repair, after which they are set to run again [8]. The M machines (this is not to be confused with the M used in the conventional queueing theory notation used earlier) break down each with a mean breakdown rate λ per unit time - a Poisson process. The inter-breakdown time is then negative exponentially distributed. Each machine is repaired at one of c repair centres with mean repair rate μ per unit time (again assumed to have a negative exponential distribution). A diagrammatic representation is given in Figure 1.

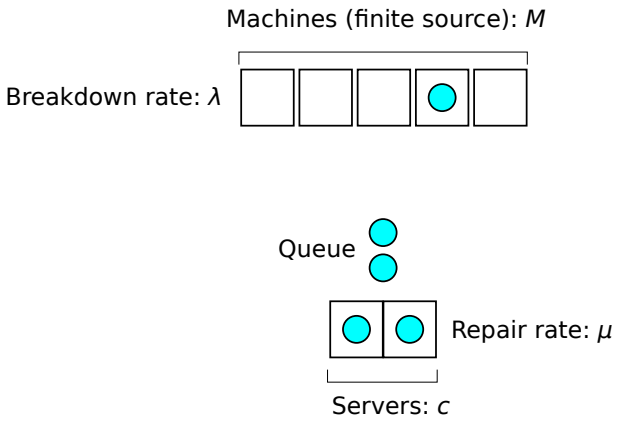

Figure 1: Diagrammatic representation of a finite source queue with $M = 5$ and $c = 2$.

Let p_n be the probability of having n of the M machines broken down at any point in time. The steady state probabilities are given by [7]:

$$p_n = \begin{cases} \binom{M}{n} \frac{\lambda}{\mu}^n p_0, & 1 \leq n < c \\ \binom{M}{n} \frac{n!}{c^{n-c} c!} p_0, & c \leq n \leq M \end{cases} \tag{1}$$

where p_0 can then be calculated by normalising the probabilities.

Of particular interest is the actual waiting time distribution $W_q(t)$: that is the proportion of individuals spending less than t time units waiting in the queue. This is given by:

$$W_q(t) = 1 - \sum_{n=c}^{M-1} q_n \sum_{i=0}^{n-c} \frac{c\mu t^i}{i!} e^{-c\mu t}, \tag{2}$$

where

$$q_n = \frac{(M-n)p_n}{L} \tag{3}$$

is the probability that an arrival finds the system in state n, and $L = \sum_{n=1}^{M} np_n$ is the average number of breakdowns in the system.

The next section describes the application of this finite source queueing model to CU's MSS.

2.2 Application to CU's MSS

The MSS at CU (and in many other institutions, see for example [14]) encourages students to work in small groups and seek assistance from a tutor when required. As such the service can be modelled as a finite source queueing model where the students working in groups represent machines. A breakdown corresponds to a student requiring assistance from a tutor.

We can use (2) to obtain staffing levels that ensure a particular level of service. In particular we obtain a value of c that ensures $1-W_q(0.25) \leq 0.1$; the number of tutors required that ensures that the percentage of students waiting more than 15 minutes is less than 10%. Data collected from the service indicates a breakdown rate of $\lambda = 2.19$ (i.e. students seek assistance about 2 times an hour). Furthermore a service rate of $\mu = 6$ is taken implying that a tutor will spend, on average, 10 minutes with a student. As arrivals at any support service will likely be affected by the timing of lectures (timetables naturally change year upon year) it is important to carefully consider the validity of historic data to provide information that is used for mathematical models. In this case we believe that the data over the academic year 2010/2011 is still representative of current demand and thus proceed.

Data collected at the MSS over the academic year 2010/2011 (data available in [9]) included the following variables:

1. The time when each student arrived
2. Number of tutor consultations the student received during the session
3. The time duration of each tutor consultation
4. The length of time each student spent waiting for their tutor consultation to begin, from the moment they requested assistance.

Figure 2 contains the mean number of students present at each session of the service, for the two semesters of the academic year, autumn and spring. The busiest semester was clearly the autumn semester, with a reduction in student attendees observed in the spring semester. In the autumn semester the busiest period was 11:00-12:00, with the following hour, 12:00-13:00 being much quieter. The busiest period was Thursday 11:00-12:00. In the spring semester, the demand is similar across all the time periods. There is some increase in attendance during the first hour of service 11:00-12:00, with the busiest period being Friday 11:00-12:00.

2.3 Results

The staffing algorithm is implemented in Microsoft Excel (the package which also implements the rostering algorithm of Section 3.2 is available at [11]) and allows for an immediate calculation (using (2)) of the number of tutors required. Initial results are given in Figure 3. The amount of staff scheduled reflects:

1. The increased demand during 11:00-12:00 in the autumn semester

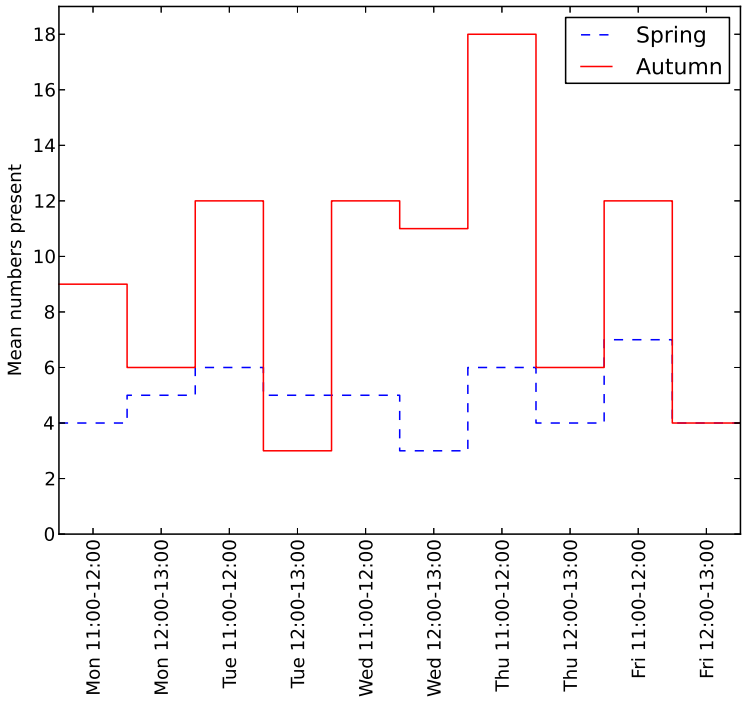

Figure 2: Mean number of students present by semester. Autumn semester in solid line, spring semester in dashed line.

2. More staff needed during the busiest session of the autumn semester (Thursday, 11:00-12:00)
3. A constant number of staff needed during the spring semester, reflecting the demand viewed in Figure 2, with an increase in the number of staff during the busiest session of the spring semester (Friday, 11:00-12:00).

3 Optimisation of staffing and skill mix

3.1 Mathematical program

The task of assigning tutors to sessions is often the responsibility of the MSS coordinator, who attempts to find a suitable number of tutors to cover all shifts, that are additionally able to cover as many specialities (statistics, pure mathematics and applied mathematics) as possible so they may deal with all query types efficiently. However, to reduce the burden placed on this individual, we provide an integer program (IP) formulation of the problem which, in smaller cases, can be solved exactly using off-the-shelf software: in our case Xpress-MP. We utilise this method to solve the IP exactly, and also suggest a heuristic-based algorithm that may be used to produce a desirable timetable for larger problems.

The model assigns the correct number of tutors required to work for each shift as determined by the machine interference model, and incorporates a constraint for the maximum number of shifts that each tutor is willing to work during the week. Treated as an optimisation problem, the objective is to maximise a linear cost function to ensure that as many fields of mathematics (statistics, pure mathematics and applied mathematics are used as examples in this paper) are covered as possible within each shift.

Variables:

- m: number of tutors.
- n: number of shifts.

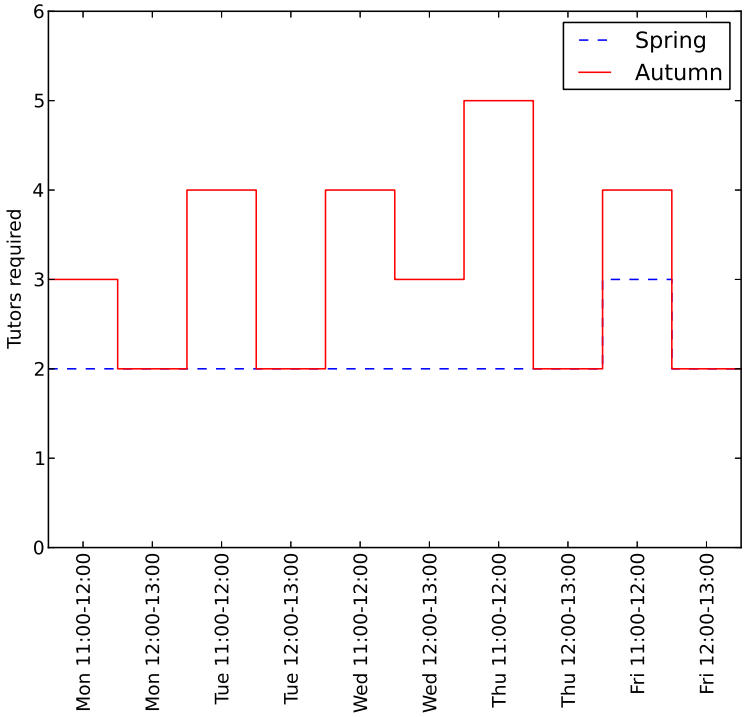

Figure 3: Number of tutors required by semester. Autumn semester in solid line, spring semester in dashed line.

- α_i: the maximum number of shifts that can be worked by tutor $i \in [m]$.
- β_j: the number of tutors required for shift $j \in [n]$.
- k: the number of specialities.
- $C_i^{(k)} \in [0,1]_{\mathbb{R}}$ the efficiency of tutor i in speciality k, denoted 'subject scores'.
- $D_i \in [0,1]_{\mathbb{R}}$ the desirability coefficient of tutor i.

We have the binary variable x_{ij}

$$x_{ij} = \begin{cases} 1, & \text{if tutor } i \text{ works in shift } j \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i \in [m];\ j \in [n]$$

The desirability coefficient is a scaling factor which is multiplied by the subject scores (the measure of expertise held by each of the tutors in statistics, applied and pure mathematics) to scale the values to account for unreliability, or any other factor, associated with each tutor. For example, if a particular tutor has asked that they are only scheduled to work if there is extreme need, we can give this tutor a low desirability score.

To achieve a roster that assigns tutors to sessions in a way that all fields are covered within each shift, the IP model representing our rostering model is given by the following:

Maximise

$$\sum_{j=1}^{n} \sum_{l=1}^{k} \min \left(1, \sum_{i=1}^{m} x_{ij} C_i^{(l)} D_i \right) \tag{4}$$

subject to constraints

$$\sum_{j=1}^{n} x_{ij} \leq \alpha_i \text{ for all } i \in [m] \tag{5}$$

$$\sum_{i=1}^{m} x_{ij} = \beta_j \text{ for all } j \in [n] \tag{6}$$

Here (5) is the worker constraint, representing the maximum number of shifts each tutor is allowed to work in a week, while (6) details the particular number of tutors that are required per session.

Because we wish to maximise the number of specialities covered in each session, we constrain each of the weighted scores within the objective function (4) to be at most 1. This ensures that high scores may not arise from sessions where the tutors all have expertise in the *same* area of mathematics. The objective is then to maximise a cost function composed of k expressions for each session (each representing the degree to which a speciality of mathematics is covered within each session).

Note that due to the formulation of the objective function, we will always have:

$$\sum_{j=1}^{n}\sum_{l=1}^{k} \min\left(1, \sum_{i=1}^{m} x_{ij} C_i^{(l)} D_i\right) \leqslant kn,$$

thus kn is an upper bound on the objective function.

In our case, the timetable is composed of 10 one hour sessions a week, and is concerned with ensuring that three separate specialities of mathematics are covered within each shift; thus if we have a sufficient number of skilled tutors available, the maximum achievable cost is 30.

3.2 Finding a roster

This section considers ways for solving the MSS rostering problem defined above. Similar problems have been well studied in the literature since the 1960s with numerous heuristics, simulation approaches and graph colouring techniques suggested to solve variants of the timetabling problem in an effective way [3, 2, 15, 13, 4]. IP formulations have also been developed to make scheduling decisions [15], but in real-life applications where these problems are often too large to solve using exact methods, heuristic approaches have been shown to be successful [13]. Heuristics are acknowledged to produce good quality solutions for such problems in a short amount of time; however they often lack the ability to find an optimal solution [1].

CU's MSS has 10 one hour sessions a week, with 8 tutors available, each with different specialities. The staff rostering problem with 10 one hour sessions and 8 tutors possesses a relatively small search space, with possibly several local optima; thus existing IP software such as Xpress-MP can be used to solve the problem exactly. However for larger instances (more sessions, and more tutors), we propose a simple heuristic (descent) algorithm for finding an approximate solution.

For the descent algorithm, the initial feasible schedule is produced using a greedy algorithm: taking the tutors in an arbitrary order, we assign as many time slots allowed by constraints (5) to the first tutor, and when this limit is reached, we consider the availability of the next tutor. Providing we have enough manpower to allow a feasible solution, we continue to allocate all remaining shifts in this way.

Local search is a general strategy for optimisation that involves iteratively applying small changes (moves) to a candidate solution, attempting to improve its quality. The random descent approach is the simplest type of local search; in each iteration a potential move is randomly selected and only accepted if it produces a solution which is at least as good as the current solution. In our algorithm, we employ a swap operator which randomly selects an element $x_{i_1 j_1}$ in the timetable such that $x_{i_1 j_1} = 1$ and a secondary element $x_{i_2 j_2}$ such that $x_{i_2 j_2} = 0$ $(1 \leqslant i_1, i_2 \leqslant m; 1 \leqslant j_1, j_2 \leqslant n)$ and swap the value of these elements. If $j_1 = j_2$ (i.e. both elements are in the same column) then no further action is needed, as we will maintain constraint (6) which ensures that the correct number of tutors are assigned to each session. Otherwise, to maintain integrity of the solution, we add a corrective procedure by randomly choosing two other elements $x_{i_3 j_2}$ and $x_{i_4 j_1}$, $(i_3 \neq i_1; i_4 \neq i_2; 1 \leqslant i_3, i_4 \leqslant m)$ and setting $x_{i_3 j_1} = 1$ and $x_{i_4 j_2} = 0$. In our descent method, the neighbourhood operator is repeatedly applied to produce new solutions for a set number of iterations, or until the upper bound kn is achieved for the objective function.

For each solution, the new assignment of tutors to shifts is accepted if the move improves on (or maintains) the current cost.

3.3 Results

Using the IP solver, we found that the best possible cost of 30 *was* achievable for the autumn semester data; and the best cost achievable for the spring semester data was 29.52. We investigated the potential of the descent method to find good quality solutions for each of the MSS instances and found that it produced solutions that often came close to optimal, but not quite. In particular, the algorithm made rapid improvements to the initial solution, and marginal improvements after around 2000 iterations. Figure 4 demonstrates that the solution came within 0.8% and 1.3% of the optimal solution after 2000 runs for the autumn and spring semester data respectively, and within 0.3% and 0.7% after 5000 runs. The algorithm provided better initial solutions for the autumn semester data (as more staff were required in each shift, it was an easier task to achieve the daily optimal score), and the capability of the algorithm to rapidly improve on the initial solution was indicated in particular for the spring semester data.

However, whereas heuristics are not necessary to roster CU's MSS, or for other instances that are small enough to optimally solve quickly and efficiently, they are appropriate for medium-sized instances.The descent algorithm would nevertheless take longer to find good quality solutions for large timetables; thus for such problems, we could consider using other heuristics such as simulated annealing.

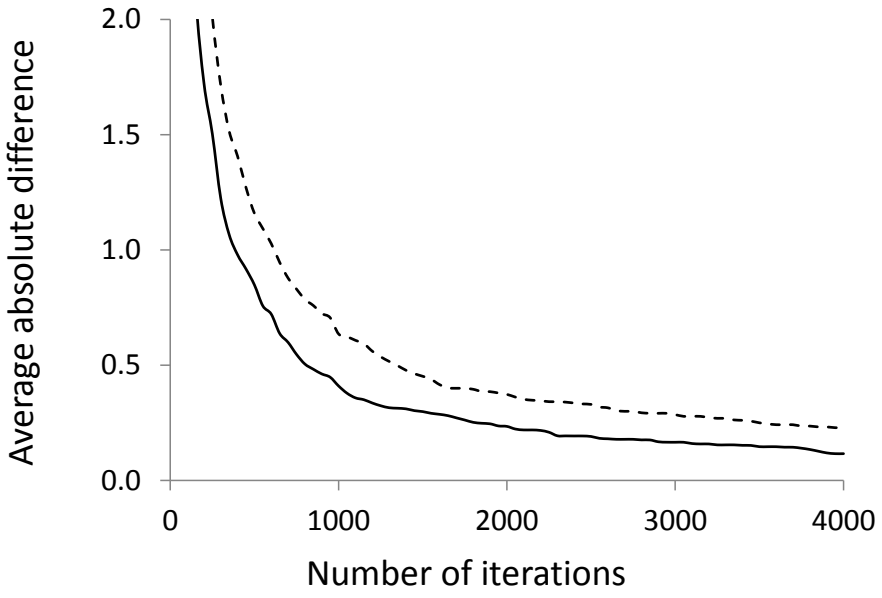

Figure 4: Plot of average absolute difference (averaged over 50 runs) of values of the objective function obtained from the solution of the descent method against iteration number, and the value of the objective function at the optimal solution (found using the IP solver). Autumn semester in solid line, spring semester in dashed line.

The optimal roster for the autumn semester data displayed in Figure 5 appears visually logical. For example, on days where only two tutors are required, tutors D and F are consistently chosen as they are able to cover all areas of mathematics to a sufficient level amongst themselves, and are both highly desirable.

Tutor	Mon 11-12	Mon 12-1	Tue 11-12	Tue 12-1	Wed 11-12	Wed 12-1	Thu 11-12	Thu 12-1	Fri 11-12	Fri 12-1	Max no. of shifts	Stats score	Pure score	Applied score	Desirability coefficient
A					1		1				5	1	0.5	0.1	0.2
B							1		1		5	0.2	1	0.2	1
C			1		1	1	1		1		5	0.1	1	1	0.5
D	1	1		1				1		1	5	0.1	0.6	1	0.8
E	1		1						1		5	0.1	1	0.5	0.2
F		1		1		1		1		1	5	1	0.8	0.4	1
G	1		1		1		1		1		5	1	0.5	0.2	1
H			1		1	1	1				5	0.1	1	0.7	0.7
Tutors required	3	2	4	2	4	3	5	2	4	2					

Figure 5: Optimal timetable for given tutor profiles, autumn semester

4 Conclusions

In this paper we have demonstrated that traditional operational research techniques are likely to be of much value to administrators of MSSs. Our approach in this paper was to model the amount of student queries received during a session of the MSS at CU requiring tutor attention as breakdowns in a machine interference model. The results generated from this model were consequently input into a specially designed IP to ensure sufficient numbers of staff were scheduled, with a sufficient coverage of mathematics specialities to allow for most queries to be readily dealt with by the tutors.

The IP has been constructed with sufficient generality so that other MSSs, with different numbers of support sessions and staff available, each with different mathematical specialities, may roster their own staff. We suspect that many problem instances will be small enough to allow for an exact solution of this integer program to be found using available IP solvers. Nevertheless, for larger problem instances where IP solvers may take too long, we have demonstrated the potential use of a heuristic (descent approach).

Acknowledgements

The authors would like to acknowledge the time and valuable feedback offered by Dr Rhydian Lewis and Dr Jonathan Thompson during the writing of this manuscript.

References

[1] D. Abramson. Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science*, 37(1):98–113.

[2] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper heuristic for timetabling problems. *European Journal of Operational Research*, 176:177–192, 2007.

[3] E.K. Burke and J.P. Newall. Enhancing Timetable Solutions with Local Search Methods. Practice and theory of automated timetabling: selected papers from the 4th International Conference. *Springer Lecture Notes in Computer Science*, 2740:195–206, 2002.

[4] H. Cambazard, E. Hebrard, B. O'Sullivan, and A. Papadopoulos. Local Search and Constraint Programming for the Post-Enrolment-based Course Timetabling Problem. *Annals of Operations Research*, 177:1–25, 2010.

[5] L. Green, P. Kolesar, and W. Whitt. Coping with time-varying demand when setting staffing requirements for a service system. *Production and Operations Management*, 16:13–39, 2007.

[6] L. Green, J. Soares, J. Giglio, and R. Green. Using queuing theory to increase effectiveness of emergency department provider staffing. *Academic Emergency Medicine*, 13:61–69, 2006.

[7] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, 1998.

[8] L. Haque and M.J. Armstrong. A survey of the machine interference problem. *European Journal of Operational Research*, 179(2):469–482, 2007.

[9] C. Harding. Measuring the effectiveness of a maths support service. *Cardiff University Undergraduate Project*, 2011.

[10] T. Hawkes and M. Savage. Measuring the mathematics problem. *London: Engineering Council*, 2000.

[11] V.A. Knight. Personal Webpage. *www.vincent-knight.com*, (Last Accessed 12/08/11).

[12] P. Kolesar, K. Rider, T. Crabill, and W. Walker. A queueing-linear approach to scheduling police patrol cars. *Operations Research*, 23:1045–1062, 1975.

[13] R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30(1):167–190, 2008.

[14] P. C. Samuels and C. Patel. Scholarship in mathematics support services. *Journal of Learning Development in Higher Education*, 2, 2010.

[15] K. Schimmelpfeng and S. Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, (4):783–803, 2007.