22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia, 3 to 8 December 2017
mssanz.org.au/modsim2017

Weighted random sampling for military aircrew timetabling

K.S. Talbot [a], D. Kirszenblat [b], W. Moran [c], V. Nguyen [d], A. Novak [d]

[a]*School of Mathematical Sciences, Monash University, Clayton, Victoria, Australia*
[b]*School of Mathematics and Statistics, University of Melbourne, Parkville, Victoria, Australia*
[c]*School of Engineering, RMIT University, Melbourne, Victoria, Australia*
[d]*Defence Science and Technology Group, Fishermans Bend, Victoria, Australia*
Email: bill.moran@rmit.edu.au

Abstract: The problem we consider is motivated by the timetabling of training schedules for Royal Australian Navy (RAN) aircrew, in particular helicopter pilots. The RAN aircrew training syllabus is a sequence of courses with a prerequisite structure that enforces a strict ordering on some segments of the training continuum. Courses have several sessions (repetitions) each year, and each session has a limited student capacity. Current ap-proaches to timetabling RAN aircrew training result in students spending significant periods (several months, say) waiting between courses. Since students are paid salaries throughout their training, such delays are costly. Furthermore, significant waiting periods can negatively affect student morale. An efficient allocation of stu-dents to schedules minimises this waiting time.

Our recent work shows that it is possible to apply Knuth's "Dancing Links" algorithm to rapidly generate all feasible schedules for RAN aircrew students. In this context, a feasible schedule is a sequence of course sessions that includes a session of each course required by the syllabus, satisfies the prerequisite constraints and avoids time clashes. Upon generating all such feasible schedules, one allocates each student to a schedule, ensuring that course session capacities are not exceeded. The objective is to minimise the total cost, in this case the total makespan of the selected schedules. The makespan of a schedule is the total time between the start of its first course session and the end of its last course session.

Realistic RAN aircrew syllabuses can generate tens of millions of feasible schedules. The number of students is, however, relatively small: on the order of fifty. Earlier work on simple test syllabuses uses a linear pro-gramming relaxation to arrive at an approximately optimal solution to the allocation problem. However, as the number of feasible schedules generated by DLX grows, standard "off-the-shelf" Integer Linear Programming (ILP) solvers may no longer be a computationally feasible choice.

This paper proposes the use of a Weighted Random Sampling (WRS) algorithm to obtain from the set of all feasible schedules a fixed-size sample that satisfies session capacity constraints. Schedules are weighted according to their makespan — shorter schedules have greater weights — and the probability that a schedule is selected for the sample is determined by its relative weight.

We compare the results of this sampling approach to the optimal solution obtained by a deterministic ILP solver for some simple test cases, and demonstrate that provided one can efficiently obtain the set of all feasible schedules, the use of a WRS algorithm is a possible alternative to an ILP formulation of the allocation problem.

Keywords: *Weighted random sampling, timetabling, military aircrew training, Dancing Links*

1 INTRODUCTION

1.1 Problem Description

Royal Australian Navy (RAN) aircrew students undertake long and complex training, in most cases taking several years to complete. The training syllabus for helicopter pilots is a sequence of major courses whose ordering is strict, together with a number of short courses that may be undertaken more flexibly between major courses, for a total on the order of twenty courses. Each course in the syllabus may have many sessions (repetitions) per year, but students undertake at most one course at a time. The session capacity (class size) is a hard constraint, and the dates and locations at which sessions occur is fixed, owing to resource requirements such as the availability of specialised equipment, facilities and instructors. Students may commence training at one of many entry points throughout the year.

The task is to allocate each student to a feasible schedule — a sequence of non-overlapping course sessions whose ordering satisfies the prerequisite structure of the training syllabus — in order to minimise the total time to graduate for all students.

Note that while our focus herein is on RAN helicopter pilots, much of the formulation and associated ideas are equally applicable to other student types, and indeed other branches of the Australian Defence Force.

1.2 A Two-Phase Approach To Timetabling

This problem has a convenient division into two phases: the determination of feasible schedules, and the allocation of each student to one of these schedules. Determining feasible schedules is a constraint satisfaction problem, in which one enumerates sequences of course sessions subject to curriculum requirements. The allocation of students to schedules must ensure that none of the course session capacities are exceeded.

Forthcoming work of Nguyen et al. (2017) shows that the enumeration of feasible schedules may be posed as an *exact cover problem*. In brief, for exact cover problems one has a finite set Ω and a collection $\mathcal{A} = \{A_n\}_{n=1}^{N}$ of subsets of Ω that cover Ω: $\bigcup_{n=1}^{N} A_n = \Omega$. One seeks a subcollection $\mathcal{A}^* = \{A_{n_k}\}_{k=1}^{K}$ of $\mathcal{A}$ of disjoint subsets that also cover Ω. Note that such a subcollection may not exist. Knuth (2000) presents *Algorithm X*, a recursive, nondeterministic, depth-first backtracking algorithm implemented using a circular doubly-linked list data structure — collectively referred to as *DLX* — to solve such problems, instances of which include Sudoku and the N-queens problem. Knuth uses a $0-1$ matrix to describe exact cover problems, in which columns correspond to elements of Ω and rows correspond to the subsets in $\mathcal{A}$. Entry (i, j) of this matrix is 1 if subset i contains element j of Ω, and is zero otherwise. In our context, one forms an analogous tableau in which the rows correspond to individual course sessions and constraints are represented by collections of columns. Using this approach and applying a modified form of Knuth's algorithm, one can efficiently enumerate all feasible schedules for a given syllabus. For an example syllabus consisting of 15 courses and between 12 and 18 sessions per course, DLX generates all schedules (~ 132 million) in approximately 40 seconds on modest desktop hardware (Nguyen et al., 2017).

The allocation of students (the number of which is relatively small; on the order of 50), subject to course session capacity constraints and with the objective of minimising the total time to graduate for all students, constititutes the second phase. One may formulate this allocation problem as an Integer Linear Program (ILP); indeed, this was our initial approach and is still the subject of ongoing work. However, the number of feasible schedules generated by DLX can result in an ILP with memory requirements that are prohibitively large for the hardware upon which it must be solved. This article can thus be viewed as an initial exploration of an alternative to formulating the second phase as an ILP.

1.3 Related Work

The military training-specific timetabling literature is sparse, and existing works present solutions to problems with objectives and constraints quite distinct from our own. Lee et al. (2009) consider the problem of allocating instructors to classes at the Korea Army Training Centre, in which an instructor gives a lecture to only one class at a time, and each class must be lectured by one instructor. Students are considered as a cohort, in that all recruits in the same batallion undertake the same subjects at the same time. Using a graph formulation — as is common in timetabling problems (de Werra, 1997) — the authors describe an edge-colouring based heuristic for its solution.

Wang et al. (2010) study the problem of scheduling term-end examinations for cadets at the United States Military Academy's West Point facility. The authors employ a hybrid heuristic–integer programming approach and propose scheduling multiple sessions of some exams in order to deal with their otherwise intractable problem. While the allocation of students to one of many repeated sessions is conceptually similar to our problem, their objective is to minimise the number of such repeats. Moreover, the number and dates of the repeat sessions in our problem are given, rather than variables to be determined.

Related timetabling and scheduling problems are well-studied in other contexts. We note two in particular: curriculum-based university timetabling (Bettinelli et al., 2015) and job-shop scheduling (Graham et al., 1979). In both cases, again the key difference between our problem and those typically studied in these domains appears to be that in our case, the time periods over which course sessions take place and their associated locations is fixed in advance.

The approach taken to other timetabling and scheduling problems does not follow the two-phase paradigm described in Section 1.2. Rather, students are allocated to courses and resources as the algorithm proceeds. Such is the case for meta-heuristic algorithms such as tabu search and simulated annealing (Schaerf, 1999) and also *column generation*, a technique often applied for large-scale integer or mixed-integer programs (Barnhart et al., 1998). We have nonetheless chosen to explore the two-phase paradigm as an alternative.

Before detailing our approach, we give a precise formulation of the problem.

2 PROBLEM FORMULATION

We consider a single training *syllabus* $\mathfrak{C}$, which consists of a (finite) set of *courses* C_i, $i \in I$. Each course C_i has a number of *course sessions* C_{ij}, $j \in J_i$, and a (possibly empty) set of *prerequisites* $\mathfrak{P}_i \subset \mathfrak{C}$ that must be completed by a student before they undertake course C_i. A course session C_{ij} has a *capacity* $M_{ij} \in \mathbb{N}$, the maximum number of students it can accommodate, and a *time extent* $\tau_{ij} = [t^s_{ij}, t^e_{ij}] \subset \mathbb{R}^+$ defined by a start and end time. A *feasible schedule* S for syllabus $\mathfrak{C}$ is then a sequence $(C_{ij})_{i \in I, j \in J_i}$ of course sessions such that

(i) For every course $C_i \in \mathfrak{C}$, there exists exactly one course session $C_{ij} \in S$;

(ii) If $C_{ij} \in S$, for every $C_{i'} \in \mathfrak{P}_i$ there is a course session $C_{i'j'} \in S$ such that $C_{i'j'}$ occurs before C_{ij} in S. That is, the prerequisite structure must be obeyed.

(iii) If $C_{ij}, C_{i'j'} \in S$, then $\tau_{ij} \cap \tau_{i'j'} = \emptyset$. That is, no two course sessions in the same schedule have overlapping time extents.

We denote by $\mathfrak{S}$ the set of all feasible schedules, which for us is obtained through the modified DLX algorithm alluded to in Section 1.2. For every $S \in \mathfrak{S}$ we define the *cost* $E(S)$ of S as the total length of time from the start of its first course session to the end of its last course session: if the first course session of S is C_{ij} and the last course session of S is C_{mn}, then $E(S) = t^e_{mn} - t^s_{ij}$. We write N for the number of students required to undertake syllabus $\mathfrak{C}$ and $n(S)$ for the number of students we assign to schedule $S \in \mathfrak{S}$. For $S \in \mathfrak{S}$, define

$$\delta_{ij}(S) = \begin{cases} 1 & \text{if course session } C_{ij} \in S \\ 0 & \text{otherwise.} \end{cases}$$

The course session capacity constraint then becomes

$$\sum_{S \in \mathfrak{S}} n(S)\delta_{ij}(S) \leq M_{ij} \quad \text{for every course session } C_{ij}. \tag{1}$$

We assume that each student is identical, so we may regard the assignment of N students to N (not necessarily distinct) schedules as simply the selection of N schedules (with *bounded* replacement; see Section 3) from $\mathfrak{S}$. Taking this view (which is more convenient for the sampling approach we propose herein), we consider $n(S)$ to be the number of times that schedule S is selected (i.e. the *multiplicity* of S in the sample $\mathcal{S}$), so that

$$\sum_{S \in \mathcal{S}} n(S) = N. \tag{2}$$

The sample $\mathcal{S}$ is thus a *multiset*. Our objective is to minimise the total training time over all students, which is simply the sum of the costs of all the schedules in the sample $\mathcal{S}$ counted according to multiplicity:

$$T(\mathcal{S}) = \sum_{S \in \mathcal{S}} n(S)E(S) \tag{3}$$

With this notation in place, the sampling problem is as follows:

Find from $\mathfrak{S}$ a sample $\mathcal{S}$ of size N that satisfies (1)–(2) and minimises (3). (P)

3 WEIGHTED RANDOM SAMPLING

Weighted Random Sampling (WRS) is the task of selecting at random a fixed number of items from a population, where the items are weighted and the probability that an item is selected is determined by its relative weight in the population (Efraimidis and Spirakis, 2006). Together with the modified DLX algorithm to generate all feasible schedules, we propose the use of a modified existing WRS algorithm to solve (P).

A WRS may be generated with replacement (Park et al., 2004) or without replacement (Wong and Easton, 1980). Using an ILP solver on test cases, we observed that the solutions corresponding to optimum values of the objective function (3) frequently feature some schedules having multiple students assigned them. In order to maintain the possibility that we attain this optimum with a WRS algorithm, clearly we must generate our sample with replacement. We must, however, satisfy (1). The maximum number of times that a particular schedule may appear in the sample is determined by the capacities of its course sessions, and those of the other schedules in the sample. We are therefore dealing with WRS with *bounded replacement*.

Many algorithms for WRS are developed for applications in which the size of the population is very large or perhaps even unknown, such as sampling from data streams in scientific simulations, web server logs or financial markets. Hardware and/or time constraints make it desirable that samples be obtained in a single pass over the population (Park et al., 2004). If the population size is unknown, an auxiliary storage known as a *reservoir* may be employed to maintain a fixed-size sample throughout the process (Vitter, 1985). Although our population $\mathfrak{S}$ is of known size thanks to its efficient enumeration via the DLX algorithm, $|\mathfrak{S}|$ may be large and so one-pass WRS with a reservoir is attractive. Moreover, one may sample schedules *as DLX generates them*, in which case one-pass WRS with bounded replacement using a reservoir is an obvious choice.

3.1 The Efraimidis-Spirakis Algorithm

Efraimidis and Spirakis (2006) present an algorithm for one-pass WRS with a reservoir, for sampling without replacement. One-pass WRS without replacement over large or unknown populations presents a challenge in the determination of the selection probabilities from the item weights: the probability that an item is selected is given by the ratio of its weight to the sum of the weights of all the items not in the sample. Such a sum may be computationally expensive or — when the population is of unknown size — impossible.

The key insight of Efraimidis and Spirakis is that probabilities can be determined by the exponentiation of uniform random variables using the item weights. More precisely, let U_1 and U_2 be independent random variables with uniform distributions on $[0, 1]$ and let w_1, $w_2 > 0$ be weights. Defining $X_1 := (U_1)^{1/w_1}$ and $X_2 := (U_2)^{1/w_2}$, they show that $P[X_1 \leq X_2] = \frac{w_2}{w_1+w_2}$, with a straightforward extension to $k > 2$ weights. This avoids the onerous computation of the aforementioned denominator. Using this fact, for each item in the population they generate a *key*: a random number in the unit interval raised to a power given by the reciprocal of that item's weight. Using our notation from Section 2, the WRS is then obtained by selecting the N (say) items with the largest keys:

Algorithm 1: Efraimidis–Spirakis

Input : A population $\mathfrak{S}$ of M weighted items
Output: A reservoir $\mathcal{S}$ with a WRS of size N
Insert first N items of $\mathfrak{S}$ into $\mathcal{S}$
for *each item* $S_i \in \mathcal{S}$ **do**
  Calculate key $k_i = u_i^{1/w_i}$, where $u_i = \text{random}(0, 1)$
end
for $i \leftarrow N + 1$ **to** M **do**
  Set theshold $T = \min\{k_j \mid S_j \in \mathcal{S}\}$
  Calculate key $k_i = u_i^{1/w_i}$, where $u_i = \text{random}(0, 1)$
  if $k_i > T$ **then**
    Replace item with minimum key in $\mathcal{S}$ by item S_i
  end
end

In a subsequent paper, Efraimidis (2015) describes an extension of Algorithm 1 to sampling with bounded replacement, in which each item S_i may have its own multiplicity k_i. The idea is to deploy N instances of Algorithm 1, each of which generates a WRS of size 1. At any point, each instance is one item behind the previous instance, so that a given item S is processed by each instance in turn, thus ensuring that the algorithm only makes one pass. Multiplicities are accounted for by excluding from remaining instances those items whose multiplicity in the sample has been reached. If such an item is replaced in the sample by another item and its multiplicity therefore reduced, it is submitted to the next instance of the algorithm.

3.2 Adaptation to Timetabling

Adapting Efraimidis' extension of Algorithm 1 for WRS with bounded replacement to the timetabling problem (P) amounts to keeping track of the course session capacities, and ensuring that when the algorithm terminates we satisfy constraint (1). This adaptation is reasonably straightforward, moreso if one assumes that the set of all feasible schedules $\mathfrak{S}$ is known. Indeed, complete knowledge of $\mathfrak{S}$ affords significant simplifications. We can determine at each step those feasible schedules that, if one were added to our sample $\mathcal{S}$, would preserve the inequality (1). To this end, define $\mathcal{S}_i := \{S \in \mathfrak{S} \mid \mathcal{S} \cup \{S\} \text{ satisfies constraint (1) at iteration } i\}$, where schedules in $\mathcal{S}$ are counted according to multiplicity, so that if S appears $n(S)$ times in $\mathcal{S}$ (recall that $\mathcal{S}$ is a multiset), S appears $(n(S) + 1)$ times in $\mathcal{S} \cup \{S\}$. At iteration i, we need only generate keys for schedules in $\mathcal{S}_i$. This removes the need for the use of a threshold key value; we may simply extract the largest key from those generated. The schedule corresponding to this largest key then constitutes a reservoir with a WRS of size 1. We add this schedule to our sample $\mathcal{S}$ and adjust the (global) course session capacities accordingly, repeating this process until we have obtained N schedules:

Algorithm 2: Efraimidis–Spirakis modified for (P)

Input : A population $\mathfrak{S}$ of M weighted items
Output: A WRS $\mathcal{S}$ of size N
Initialise $\mathcal{S} = \emptyset$
for $i \leftarrow 1$ **to** N **do**
  Determine $\mathcal{S}_i$
  for *each schedule* $S_j \in \mathcal{S}_i$ **do**
    Calculate a key $k_j = u_j^{1/w_j}$, where $u_j = \text{random}(0, 1)$
  end
  Determine $k_{j^*} = \max\{k_j \mid S_j \in \mathcal{S}_i\}$
  Add to $\mathcal{S}$ the schedule S_{j^*} corresponding to k_{j^*}
  for *each course session* $C_{l^*m^*}$ *in schedule* S_{j^*} **do**
    $M_{l^*m^*} \leftarrow M_{l^*m^*} - 1$
  end
end

4 RESULTS AND FUTURE WORK

We compare the results of using Algorithm 2 to solve (P) to the optimal solution obtained by a deterministic ILP solver for three test cases that model the sequence of major courses undertaken by RAN aircrew. We used our modified DLX algorithm to generate all feasible schedules $\mathfrak{S}$, and for schedule weights we use an exponential distribution of makespans, so that schedule S has weight $w(S) = \lambda \exp(-\lambda E(S))$. In each case we use $\lambda = 0.1$. Table 1 summarises our results, with objective function values given in days. We see that

Table 1. Summary of results for the 3 test cases

Case	N	# courses	# sessions/course	# schedules	ILP optimum	WRS $T(\mathcal{S})$
1	20	3	$3-4$	42	15604	15604
2	24	6	$9-17$	737	16152	16152
3	24	5	12	5882	24440	28010

Algorithm 2 recovered the optimum value of the objective function $T(\mathcal{S})$ for Cases 1 and 2, but exceeded the optimum (minimum) by over 14% in Case 3. We believe that this may be due to a poor choice of schedule weights. The parameter λ controls the steepness and decay of the weight distribution, so that finding the optimal λ for a given problem is an optimisation problem in itself. This suggests the potential use of an annealing procedure on λ in order to improve the value of $T(\mathcal{S})$.

We tested Algorithm 2 on several larger syllabuses, for which DLX generated schedules numbering from under $15,000$ to over 3 million. However, in each of these cases the algorithm reached an iteration $i < N$ at which $\mathcal{S}_i$ was determined to be empty, i.e. there does not exist any schedules that could be included in the sample without violating (1). Upon closer examination of the partial samples obtained in these cases, we observed that a small number (either one or two) of schedules were selected their maximum number of times (equivalently, no additional students could be allocated these schedules). These schedules appeared to feature a 'critical' sequence of course sessions, in that by allocating as many students as possible to them, they reduced to zero the available capacities of all other feasible schedules. This 'rigidity' displayed by the algorithm again seems a symptom of a poor choice of weight function, and despite testing numerous values of λ, we were unable to obtain full samples in these cases. A potential solution to this issue would be to incorporate the remaining capacity of a schedule into the weight distribution, in such a way that given two schedules of equal length, the schedule with the greater remaining capacity is assigned a higher weight. This would, however, entail updating schedule weights with each iteration, which may be undesirable when the number of schedules is large.

Indeed, taking the aforementioned issues into account, future work should include a strong focus on how to determine the weights of the schedules. Rather than specifying a specific functional form for their distribution, the performance of Algorithm 2 might be better served by a nonparametric approach, in which the schedule data determine the weights intrinsically.

5 CONCLUSION

We introduced a two-phase paradigm for timetabling military aircrew training, in which the generation of feasible schedules is separated from the allocation of students to these schedules. Schedules are generated using a modified version of Knuth's "Dancing Links" algorithm. Owing to computational constraints, as an alternative to integer linear programming formulations of the allocation phase we proposed the use of an existing Weighted Random Sampling algorithm, modified to account for course session capacities. This approach has the advantage of simplicity, ease of understanding and implementation. Preliminary results on simple syllabuses show some promise, however further research is needed to ensure its reliability on larger, more complex syllabuses, and a formal complexity analysis and proof of consistency is necessary to obtain theoretical confirmation of its strengths and weaknesses.

ACKNOWLEDGEMENT

This work was performed while KST and DK were undertaking internships with the Defence Science & Technology Group at Fisherman's Bend, enabled by the Australian Mathematical Sciences Insititute's AMSI Intern program. We thank AMSI Intern for facilitating this collaboration.

REFERENCES

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research 46*(3), 316–329.

Bettinelli, A., V. Cacchiani, R. Roberti, and P. Toth (2015). An overview of curriculum-based course timetabling. *TOP 23*(2), 313–349.

de Werra, D. (1997). The combinatorics of timetabling. *European Journal of Operational Research 96*(3), 504–513.

Efraimidis, P. S. (2015). Weighted random sampling over data streams. In *Algorithms, Probability, Networks, and Games*, pp. 183–195. Springer.

Efraimidis, P. S. and P. G. Spirakis (2006). Weighted random sampling with a reservoir. *Information Processing Letters 97*(5), 181–185.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. R. Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 5*, 287–326.

Knuth, D. E. (2000). Dancing links. *Millenial Perspectives in Computer Science 18*, 187–214.

Lee, S.-Y., Y.-D. Kim, and H.-J. Lee (2009). Heuristic algorithm for a military training timetabling problem. *Asia Pacific Management Review 14*(3), 289–299.

Nguyen, V., B. Moran, A. Novak, T. Caelli, M. Shokr, and K. Pash (2017). A new paradigm for optimal timetabling. In preparation.

Park, B.-H., G. Ostrouchov, N. F. Samatova, and A. Geist (2004). Reservoir-based random sampling with replacement from data stream. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 492–496. SIAM.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review 13*(2), 87–127.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS) 11*(1), 37–57.

Wang, S., M. Bussieck, M. Guignard, A. Meeraus, and F. O'Brien (2010). Term-end exam scheduling at United States Military Academy/West Point. *Journal of Scheduling 13*(4), 375–391.

Wong, C.-K. and M. C. Easton (1980). An efficient method for weighted sampling without replacement. *SIAM Journal on Computing 9*(1), 111–113.