Course Scheduling Under Uncertainty for Defence Workforce Training

Trudy Lam[1][a], Vicky Mak-Hau[1][b] and Kristan Pash[2]
[1]*School of Information Technology, Deakin University, Waurn Ponds, Geelong, Australia*
[2]*Defence Science and Technology Group, Department of Defence, Fishermans Bend, Australia*

Keywords: Scheduling, Uncertainty, Simulation, Optimisation.

Abstract: In this paper, we revisit a previously studied Defence course scheduling problem where trainees are allocated to course sessions, the Simultaneous Sequencing and Allocation Problem. The courses have a non-linear prerequisite structure, each course has a number of sessions, and each session has a class size limit. The uncertainty in the planning is due to a historical pass-rate associated with each course, and a trainee will not be able to proceed once s/he fails a course. We develop a simulation-based three-stage solution algorithm that optimises the scheduling for each scenario using integer programming. The way pass-rates are handled is different from previous approaches, and because of this, we are able to provide decision-makers with better insights on best-case, average-case, and worst-case outcomes.

1 INTRODUCTION

In this paper, we study a Defence trainee course scheduling problem with multiple business requirements.

Defence personnel maintain and develop their professional skills by attending applicable courses throughout each phase of their careers, with some courses forming part of the requirements for promotion to higher ranks.

The efficient scheduling of course sessions and allocating personnel to sessions can reduce the number of times each course must be sessioned, reduce waiting times, give more time for individuals to practice their learned skills, reduce time to promotion, and increase morale.

1.1 Problem description

The optimisation problem concerns the allocation of Defence trainees to course sessions in a planning horizon.

The courses have a pre-requisite structure. See Figure 1 for an example, where the course structure of a 25-course instance is provided. Since the courses are not linearly structured, there are multiple components of the courses that can be taken in a different order. Take the 25-course system as an example. Once a trainee has successfully completed Course 2, they can proceed to either Courses 3, 4, 6, 7, 8, 9, 10 or 17 without violating any pre-requisite requirement. Course 13, on the other hand, can only be taken if a trainee has successfully completed Courses 1, 2, and 4–12, but it can be taken either before or after Courses 3 and 17. Course 13 is also a direct pre-requisite for Courses 14 and 15 and an implied pre-requisite for Courses 16–25. Each trainee has only one chance to complete a course. If a trainee fails a course, s/he is most likely offered a transfer to a less complex trade or be asked to leave the service. Each course has a number of sessions. A trainee will only take one session for each course. All sessions for the same course have the same duration; however, the span of different sessions of a course may overlap. A trainee can only take one (course) session at a time. Each course session has a lower limit and an upper limit on the size of the class (i.e., the number of trainees that can be admitted to the session).

There is a historic pass-rate associated with each course (as shown in Figure 1). We were informed that the pass-rates are independent of the sequence the courses taken so far by a trainee. This means that, for example, the probability of a trainee successfully completing Courses 1, 2, 6, 11, and 8 will be 0.3249. However, these five courses can be taken in one of the following orders (1, 2, 6, 11, 8), (1, 2, 6, 8, 11), and (1, 2, 8, 6, 11). This leads to an interesting question - how far can each trainee reach in the system (in other words, how many courses can a trainee complete be-

[a] https://orcid.org/0000-0003-1412-6598
[b] https://orcid.org/0000-0002-9306-5780

Lam, T., Mak-Hau, V. and Pash, K.
Course Scheduling Under Uncertainty for Defence Workforce Training.
DOI: 10.5220/0011924200003396
In *Proceedings of the 12th International Conference on Operations Research and Enterprise Systems (ICORES 2023)*, pages 269-278
ISBN: 978-989-758-627-9; ISSN: 2184-4372

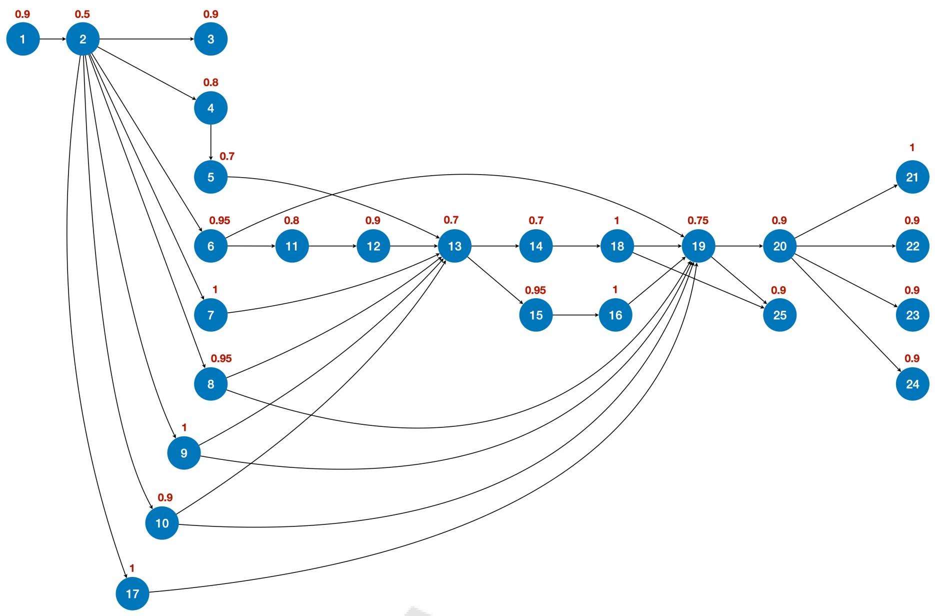


Figure 1: This figure illustrates the course structure of the problem instances with 25 courses. The nodes are the courses. An arc between two courses indicates the pre-requisite relation (e.g., Course 1 is a pre-requisite of Course 2). The numbers in red on top of each node represent the historic pass-rate of the course the node represents.

fore s/he is out of the system) and in what sequence are these courses completed?

A mathematical description of the Simultaneous Sequencing and Allocation Problem (SSAP) is given below. Consider a directed graph $D = (V, A)$, where V represents all the course sessions and is partitioned to $|C|$ clusters. Each cluster represents a course, and the vertices in each cluster represent available sessions for the course. Each vertex has a capacity (class size) and a time-window associated with it. The time-windows for all vertices in the same cluster have the same span; however, the time windows themselves are different but may overlap. Precedence relations exist between some clusters. The optimisation problem is to find a "route" for each trainee such that no more than one vertex from each cluster is visited. In this aspect, the SSAP problem is similar to a generalized Vehicle Routing Problem with Time-Windows (GVRP-TW). However, the major difference is that vehicles do not disappear mid-tour in a Vehicle Routing Problem, but in the SSAP problem, trainees will not be able to proceed if they fail a course.

1.2 Literature Review

Optimal scheduling and timetabling problems can be found in many industry sectors, e.g., hospital trainees scheduling (Beliën and Demeulemeester, 2006), military aircraft fleet management, (Looker et al., 2017), and army training (Lee et al., 2009), among others. Many are based on graphs and networks, for example, timetabling problems as graph colouring problems, (de Werra, 1985). Integer programming models that lead to further research in solution methodologies for real-world timetabling problems can be found in, e.g., (Carter, 1989; Costa, 1994; Burke et al., 2005). Many heuristic approaches for timetabling- as well as scheduling-family problems can be found in the literature. For example, (Glover and Laguna, 1997; Battiti and Tecchiolli, 1994; Dell'Amico and Trubian, 1993; Hertz and Widmer, 1996; Nowicki and Smutnicki, 1996). Regarding meta-heuristic algorithms, their hybrids and hyper-heuristics have some success in scheduling problems in various domains (Teoh et al., 2015; Pillay, 2016). In vehicle routing-family problems, much recent research is on the "rich" optimisation version of the problems, such as (Lam and Hentenryck, 2016).

In the literature of scheduling work in the education domain, the majority of work are on examination timetabling and course timetabling (Rudová et al., 2011; Bettinelli et al., 2015; Müller and Rudová, 2016). The university course timetabling problems are mainly about scheduling lectures, tutorials, and so on, that recurs on a weekly basis, be it scheduling classes after students have enrolled, whereas in post-enrolment course timetabling, or the schedule of classes are defined before enrolment in curriculum-based course timetabling. For this reason, they are very different from the SSAP problem.

In military training, policy effects, bottlenecks, and resource utilisation has been studied in, e.g., (Davenport et al., 2007; Séguin and Hunter, 2013; Novak et al., 2015; Nguyen et al., 2016). System dynamics simulations in workforce planning is reviewed in (Wang, 2005). Heuristic approaches can be found in (Yang and Ignizio, 1987) for peacetime scheduling of training activities of battalions where decisions must be made in regards to the kind of training tasks to be performed, by which battalion, at what time, with the objective of minimizing the make-span. In military and other aircrew training, heuristic methods can be found in, e.g., (Lee et al., 2009). A hybrid heuristic-integer programming approach was used to schedule examination timetables for cadets in (Wang et al., 2010). A two-phase heuristic method is proposed for scheduling army battalion training exercises in (Yang and Ignizio, 1987). Integer programming has been applied in (Scott, 2005) to produce multi-year schedules at the Defence Language Institute. In (Qi et al., 2004), a hybrid Branch and bound method and heuristics are applied to the scheduling of the retraining of pilots for Continental Airlines. Dynamic programming and column generation approaches were used in (Raffensperger and Schrage, 2008) for training in a tank battalion. A resource scheduling problem is presented in (McGinnis and Fernandez-Gaucherand, 1994) for the US Army's basic combat training program. An intra-theatre military airlift problem was presented in (Brown et al., 2013). In (Squires and Hoffman, 2015), a military maintenance planning and scheduling problem is modelled as ILPs and solved using Benders decomposition. A logic-based Benders decomposition approach for multi-phase course timetabling is proposed in (Esmaeilbeigi et al., 2022).

The SSAP problem has been studied in (Nguyen et al., 2018), (Mak-Hau et al., 2021), and (Nguyen et al., 2022). The common feature of these methods is that schedules are formed by sequencing the course sessions either before the optimal allocation of trainees to these schedules, or the two problems solved simultaneously. The course pass-rates are handled deterministically in the sense that the number of trainees to be taking a particular session of a course is obtained by multiplying the number of trainees allocated to the session by the accumulative pass-rate from the first course to the course immediately before the current course. Further details will be provided in Section 3.

In (Nguyen et al., 2018), the timetabling problem is solved as a generalised exact cover problem. The set of all feasible schedules is enumerated using Donald Knuth's Dancing Links algorithms DLX.

On the other hand, multiple mixed-integer linear programming (MILP)-based approaches are proposed and tested in (Mak-Hau et al., 2021). The first MILP model considers each trainee individually as a schedule. A continuous decision variable is used to represent the probability that a trainee has not failed when s/he reaches a course, and the class size constraint is based on the expected number of trainees attending a course session. The second method is a two-stage approach with the Stage-one MILP using an integer variable to determine the expected number of trainees taking a schedule. Given that the Stage-one MILP did not show significant improvement, Stage-two MILP was not implemented. The third method is a column generation-based heuristic, where an integer programming (IP) model is solved, with each column as a schedule. Column generation is used for solving the Linear Programming (LP) relaxation, and the columns obtained are ultimately used as the set of columns in the IP. The LP solution to all but two instances tested are naturally integral, and for the ones that are not, they are solved to optimality at the root node in under 0.05 seconds due to the pre-solve, reduction, and cutting planes implemented automatically by CPLEX.

Another search algorithm is proposed in (Nguyen et al., 2022), where schedules with minimum cost-function values are generated by Knuth's Dancing Links indexing scheme with A^* search, and trainees are greedily allocated until the maximum class size is reached by one or more of sessions. This approach is most computationally efficient when compared to (Nguyen et al., 2018) and (Mak-Hau et al., 2021).

These methods essentially provide an "expected number" of trainees attending each course session, and it must be within the class size's lower and upper limits. Essentially this gives users insights into an average-case outcome. If resources (such as instructors and equipment) are planned based on the average-case scenario, in real-life practice, there may be an under-resourcing issue. However, resourcing for worst-case or somewhere between average-case and worst-case requires the knowledge of what the worst-case scenarios are, which previous approaches cannot tell us unless we change the pass-rate. For this reason, we propose an alternative approach that will allow us to obtain insights on best-case, average-case, and worst-case solutions, and to take them into consideration in evidence-based decision-making.

1.3 Contributions and Outline of Paper

This paper proposes a new paradigm to integrate historical pass-rates into our optimisation procedure, and because of that, we obtained a new integer programming formulation for the SSAP problem. We implemented and extensively tested the new method on the same set of test instances used in (Nguyen et al., 2018; Nguyen et al., 2022; Mak-Hau et al., 2021), and presented best-case, average-case, and worst-case outcomes in the number of distinct sessions required as well as total make-span. We provide the Integer Programming formulation in Section 2 and a detailed explanation of pass-rate based progression simulation in Section 3. In Section 4, we present the computational experiments. Finally, we conclude our findings and present future research directions in 5.

2 INTEGER PROGRAMMING FORMULATION

First we introduce the mathematical notation used in the rest of the paper. Let:

- $\mathcal{S} = \{1,\dots,S\}$ be the set of all trainees;
- $\mathcal{C} = \{1,\dots,C\}$ be the set of all courses;
- p_c, the pass-rate of Course c;
- $\mathcal{K} = \{1,\dots,K\} = \{1,\dots,n_1,n_1+1,\dots,n_2,n_2+1,\dots,n_{c-1}+1,\dots,n_c\}$ be the set of all sessions;
- E_s the time trainee s enters the training program;
- $k_{\min}$ and $k_{\max}$ the minimum and maximum number of trainees Session k can take;
- $n_c - n_{c-1}$ the number of sessions for Course c; with $n_0 = 0$;
- σ_k and ε_k the start and end time of Session k; and
- $\mathcal{E} = \{(c,c') \mid c,c' \in \mathcal{C}, c \text{ is a pre-requisite of } c'\}$ the set of all pairs of courses with a pre-requisite relation.

We define the following decision variables.

- $z_k \in \{0,1\}$ with $z_k = 1$ indicating Session k is scheduled and $z_k = 0$ otherwise; and
- $x_{sk} \in \{0,1\}$ with $x_{sk} = 1$ indicating trainee s is allocated to Session k, and $x_{sk} = 0$ otherwise.

We now discuss the constraints used to capture the business rules. Each trainee will take no more than 1 session for a course.

$$\sum_{k=n_{c-1}+1}^{n_c} x_{sk} \leq 1, \quad \forall s \in \mathcal{S}, c \in \mathcal{C} \qquad (1)$$

The number of trainees enrolled in a session must be within its lower and upper class size limit.

$$k_{\min} z_k \leq \sum_{s \in \mathcal{S}} x_{sk} \leq k_{\max} z_k, \quad \forall k \in \mathcal{K} \qquad (2)$$

The constraint below will ensure that if Course c is a pre-requisite of Course c', then a trainee cannot take Course c' unless they have taken Course c.

$$\sum_{k=n_{c'-1}+1}^{n'_c} x_{sk} \leq \sum_{k=n_{c-1}+1}^{n_c} x_{sk}, \quad \forall s \in \mathcal{S},\ (c,c') \in \mathcal{E} \qquad (3)$$

Now, if a trainee will take both of Courses c and c', then the start date of Course c' for this trainee cannot be earlier than the end date of Course c for this trainee. To model this requirement, let α_{sc} be a binary decision variable with $\alpha_{sc} = 1$ if trainee s will complete Course c and $\alpha_{sc} = 0$ otherwise. We have that

$$\alpha_{sc} = \sum_{k=n_{c-1}+1}^{n_c} x_{sk}, \quad \forall s \in \mathcal{S},\ c \in \mathcal{C} \qquad (4)$$

We introduce another binary decision variable $\beta_{s,c,c'}$ with $\beta_{s,c,c'} = 1$ representing trainee s will take both Courses c and c', and $\beta_{s,c,c'} = 0$ otherwise (which means that they trainee would have failed some courses before reaching Course c'). These can be captured by the following constraints.

$$\alpha_{s,c} + \alpha_{s,c'} \leq \beta_{s,c,c'} + 1, \quad \forall s \in \mathcal{S},\ (c,c') \in \mathcal{E} \qquad (5)$$

$$\beta_{s,c,c'} \leq \alpha_{s,c}, \quad \beta_{s,c,c'} \leq \alpha_{s,c'}, \forall s \in \mathcal{S},\ (c,c') \in \mathcal{E} \qquad (6)$$

We have that

$$\sum_{k=n_{c-1}+1}^{n_c} \varepsilon_k x_{sk} + 1 \leq \sum_{k=n_{c'-1}+1}^{n_{c'}} \sigma_k x_{sk} + M_{c'}(1-\beta_{s,c,c'}), \quad \forall s \in \mathcal{S},\ (c,c') \in \mathcal{E} \qquad (7)$$

where $M_{c'}$ is a large number and we used the latest session finish time of Course c'.

Each trainee can only be taking one session at a time. We enumerate the set of all cliques Ξ, where each clique contains a maximal set of sessions that cannot be taking simultaneously.

$$\sum_{k \in \xi} x_{sk} \leq 1, \quad \forall s \in \mathcal{S},\ \xi \in \Xi \qquad (8)$$

See Figure 2 for an illustration of all cliques for Problem Instance 2repsPrYr_5courses.

3 PROGRESSION SIMULATION BASED ON PASS-RATE

Previously, the course pass-rates are handled in the following manner. Take the 5-course problem instances as an example. The historic pass-rates of Courses C1, C2, C3, C4, and C5 are 0.9, 0.5, 0.9, 0.8, and 0.7, respectively. Suppose that a trainee takes the course schedule: C1, C2, C4, C3, followed by C5, in

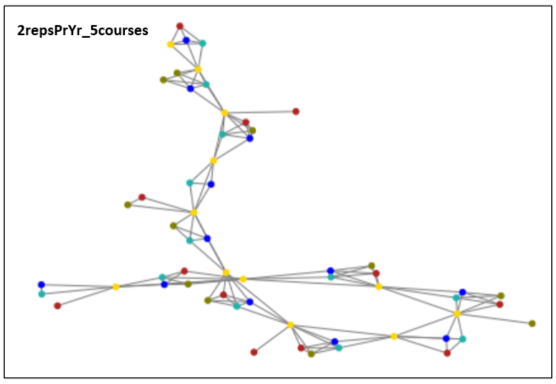


Figure 2: Illustration of the set of all cliques Ξ for the 2repsPrYr_5courses problem instance. The colours represent the course to which the nodes (i.e., sessions) belong. That is, blue denotes Course 1, green denotes Course 2, red denotes Course 3, olive denotes Course 4, and yellow denotes Course 5.

this precise order. The cumulative pass-rates will be 0.9, 0.45, 0.36, 0.324, and 0.2268. In other words, there is a 36% chance, for example, the trainee will pass C1, C2, and C4. Once s/he makes it to Course C3; however, the probability of passing the course is still 80%. Time-feasible schedules (a session for each course, and all courses must be included exactly once) are either exhaustively enumerated using DLX (Knuth's Dancing Links) or generated within a mixed-integer linear program. For each course session, all the schedules generated by DLX that contain the course session are considered in the class-size constraint.

As an example, suppose that Session 2 of Course C3 has a lower class size limit and an upper limit of 2 and 3, respectively. Now, in a solution, Session 2 of Course C3 are only taken by Schedule 1 and Schedule 2. Schedule 1, with the following order: (C1, C2, C4, C3, C5), has two trainees allocated to it. The probability that a trainee can pass courses C1, C2, and C4 and reach C3 is 36%. Schedule 2, with the following order: C1, C2, C3, C4, C5, has three trainees allocated to it, and also takes Session 2 of Course C3. The probability a trainee can pass courses C1 and C2 and reach C3 is 45%. The expected number of trainees taking Session 2 of Course C3 will be $0.36 \times 2 + 0.45 \times 3 = 2.07$. Since $2 \leq 2.07 \leq 3$, the class size constraint for Session 2 of Course C3 is satisfied. Pass-rates handled in this manner, in a way, provide users with an average-case outcome.

In reality, however, if we have both trainees allocated to Schedule 1 and two out of three trainees allocated to Schedule 2 passing the relevant prior courses and reach Course C3, the upper class size limit of 3 would have been violated, and one of them would have to be re-allocated to another session. This has an impact on resourcing for the planning horizon. Resource planning based on an average-case scenario is risky, and therefore being able to obtain insights into the risk boundaries is critical.

In what follows, we propose a new approach for handling pass-rates. The key idea of the proposed method is the course failure simulation, which determines the course at which a trainee may fail based on the historic pass-rates. A trainee who fails a course will not be able to repeat the course. As a result, failing a course suggests that the trainee will not take all other courses for which the failed course is the direct or implied prerequisite course. Additionally, the proposed method incorporates a post-processing procedure in order to determine the courses to be done prior to the failed course. This post-processing procedure can help find the set of to-do courses more easily for some large problem instances. It is noted that the existence of a large number of courses that can be done before the failed course renders the enumeration of all possible permutations inefficient. Take the 25-course problem instance (see Figure 1) as an example. If the trainee fails Course 3, then there are 22 other courses (i.e., 4, 5, ..., 24, 25) that Course 3 can be among in any position. This gives a total of 23! possible permutations.

The new approach is summarised in Algorithm 1. In this pseudocode, $pred(c)$ denotes the set of direct and implied prerequisite courses of $c \in \mathcal{C}$, $succ(c)$ denotes the set of all courses for which c is a direct or implied prerequisite course.

The algorithm starts with Phase 1 (WHILE loop lines 5 - 12), which is an iterative procedure that finds the failed course for the considered trainee. At each iteration, a course c is assigned a random number r generated from uniform distribution $U(0,1)$. If r is not greater than the pass-rate, then the procedure moves to the next course. Otherwise, this course c will be considered as a failed course. In the pseudocode below, the failed course of trainee s is denoted by c_s^*. This procedure terminates when a failed course is found or when all courses have been considered. Note that if c_s^* is 0 when the While loop completes, this indicates that the trainee passes all courses. If this happens, Phase 2 will not be invoked, and the algorithm simply returns the set of all courses (lines 13 - 15).

Given the failed course c_s^* of trainee s, Phase 2 of the algorithm includes a post-processing procedure, which attempts to determine what number of courses and exactly which courses are done before c_s^* (lines 17 - 60). When determining how many courses from the set $Parallel(c_s^*)$ that should be done before the failed

Algorithm 1: Simulating Trainee's Progression.

```
1: **Input**: Trainee $s \in \mathcal{S}$; pass-rate $p_c, \forall c \in \mathcal{C}$;
2: **Output**: Failed course $c_s^*$; set of courses that trainee $s$ may complete before the failed course $\mathcal{R}_s$
3: **Phase 1: Determine at which course the trainee may fail**
4: Set $c \leftarrow 1$, $c_s^* \leftarrow 0$
5: **while** $c \leq C$ **do**
6:     Generate $r \sim U(0,1)$
7:     **if** $r \leq P_c$ **then**
8:         Set $c \leftarrow c+1$
9:     **else**
10:        $c_s^* \leftarrow c$; break
11:    **end if**
12: **end while**
13: **if** $c_s^* = 0$ **then**
14:     **return** $\mathcal{C}$
15: **end if**
16:
17: **Phase 2: Determine which courses are done before the failed course $c_s^*$**
18: Initialise $parallel(c_s^*) \leftarrow \mathcal{C} \setminus (pred(c_s^*) \cup succ(c_s^*) \cup \{c_s^*\})$
19: Set $n \leftarrow |parallel(c_s^*)|$
20: Calculate $\pi_k, k = 0, ..., n$ according to (9)
21: Using $\pi_k$, randomly select a number $k \in [0, ..., n]$
22: $ImmediateNext \leftarrow \emptyset$, $status \leftarrow True$
23: **for all** $c \in parallel(c_s^*)$ **do**
24:     **for all** $j \in pred(c)$ **do**
25:         **if** $j \in parallel(c_s^*)$ **then**
26:             $status \leftarrow False$; break
27:         **end if**
28:     **end for**
29:     **if** $status = True$ **then**
30:         $ImmediateNext \leftarrow ImmediateNext \cup \{c\}$
31:     **end if**
32: **end for**
33: $\mathcal{R}_s \leftarrow \emptyset$
34: **if** $k \leq |ImmediateNext|$ **then**
35:     **while** $|\mathcal{R}_s| \leq k$ **do**
36:         $c' \leftarrow$ randomly select a course from the set $ImmediateNext$.
37:         $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{c'\}$
38:         $ImmediateNext \leftarrow ImmediateNext \setminus \{c'\}$
39:     **end while**
40: **else**
41:     **while** $|\mathcal{R}_s| \leq k$ **do**
42:         $c' \leftarrow$ randomly select a course from the set $ImmediateNext$.
43:         $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{c'\}$
44:         $\mathcal{F} \leftarrow parallel(c_s^*) \setminus (\mathcal{R}_s \cup ImmediateNext)$
45:         $ImmediateNext \leftarrow ImmediateNext \setminus \{c'\}$
46:         **for all** $i \in \mathcal{F}$ **do**
47:             $status \leftarrow True$
48:             **for all** $j \in pred(i)$ **do**
49:                 **if** $j \notin \mathcal{R}_s$ **then**
50:                     $status \leftarrow False$; break
51:                 **end if**
52:             **end for**
53:             **if** $status = True$ **then**
54:                 $ImmediateNext \leftarrow ImmediateNext \cup \{i\}$
55:             **end if**
56:         **end for**
57:     **end while**
58: **end if**
59: Add $(c, c_s^*)$ to $\mathcal{E}$, for all $c \in \mathcal{R}_s$
60: $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup pred(c_s^*)$
61: **return** $c_s^*$, $\mathcal{R}_s$
```

course (lines 18 - 21), the number is chosen randomly with non-uniform probability, i.e., larger number has lower probability of being chosen. Specifically, the probability of having to do exactly k courses before the failed course is given by

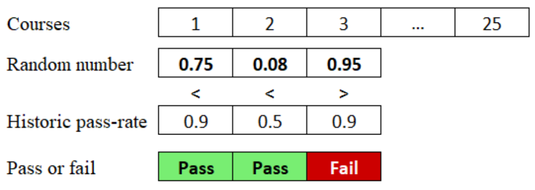
$$\pi_k = \frac{n-k+1}{(n+1)(n+2)/2}, \quad (9)$$

where n is the total number of courses in the set $Parallel(c_s^*)$. The rationale behind (9) is that as there needs to be more courses done before the failed course, hence the higher the chance a trainee would have failed before they reach this course. A more sophisticated formula can be considered and we leave this for future research. Let k be the number chosen at random based on (9). The next step of the post-processing procedure is as follows. First, a set of courses that can be immediately taken after the last course in $pred(c_s^*)$, denoted by $ImmediateNext$, is constructed (lines 22 - 32). Then, we initialise a set of courses that trainee s may complete before the failed course, denoted by $\mathcal{R}_s$, which is empty at the beginning. Next, we randomly select a course $c' \in \mathcal{C}$ from the set $ImmediateNext$ and add it into the set $\mathcal{R}_s$. If the size of the set $ImmediateNext$ is not greater than k, this step repeats by adding courses from $ImmediateNext$ to $\mathcal{R}_s$ until the size of $ImmediateNext$ reaches k. On the other hand, if the size of the set $ImmediateNext$ is greater than k, an additional step is performed to insert courses that can be immediately done after course c', that is not already in $ImmediateNext$. We present an example of the trainee's progression simulation procedure below using a 25-course problem instance.

Example. Suppose that the considered trainee, i.e. $s = 1$, fails course 3 (see Figure 3 for an illustration of the failed course). According to the course structure in Figure 1, we have $pred(3) = \{1,2\}$ and $succ(3) = \emptyset$. Thus, the set of courses that can be completed before reaching Course 3 is $Parallel(3) = \{1,...,25\} \setminus \{1,2\} \setminus \{3\} = \{4,5,...,25\}$ and the set of courses that can be immediately taken after the last course in $pred(3)$, i.e., Course 2, is $ImmediateNext = \{4,6,7,8,9,10,17\}$. Suppose that the random number chosen is 5, which means that the trainee may complete five courses before reaching Course 3. Since the size of $ImmediateNext$ is greater than 5, the algorithm randomly selects five courses from $ImmediateNext$ and adds them to $\mathcal{R}_1$, e.g., $\mathcal{R}_1 = \{4,6,7,10,17\}$. Otherwise, if the random number chosen is 8, which is greater than the size of $ImmediateNext$. In this case, $\mathcal{R}_1$ is constructed as follows:

- Iteration 1: Suppose that Course 6 was chosen at random from $ImmediateNext$. Then, we have $\mathcal{R}_1 = \{6\}$ and $ImmediateNext = \{4,7,8,9,10,17,11\}$

- Iteration 2: Suppose that Course 4 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4\}$ and $ImmediateNext = \{7,8,9,10,17,11,5\}$
- Iteration 3: Suppose that Course 5 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5\}$ and $ImmediateNext = \{7,8,9,10,17,11\}$
- Iteration 4: Suppose that Course 7 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5,7\}$ and $ImmediateNext = \{8,9,10,17,11\}$
- Iteration 5: Suppose that Course 11 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5,7,11\}$ and $ImmediateNext = \{8,9,10,17,12\}$
- Iteration 6: Suppose that Course 12 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5,7,11,12\}$ and $ImmediateNext = \{8,9,10,17\}$
- Iteration 7: Suppose that Course 17 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5,7,11,12,17\}$ and $ImmediateNext = \{8,9,10\}$
- Iteration 8: Suppose that Course 10 was chosen at random from *ImmediateNext*. Then, we have $\mathcal{R}_1 = \{6,4,5,7,11,12,17,10\}$ and $ImmediateNext = \{8,9\}$
- Finally, $\mathcal{R}_1 = \{6,4,5,7,11,12,17,10\} \cup \{1,2\}$, which gives $\mathcal{R}_1 = \{1,2,4,5,6,7,10,11,12,17\}$



Courses	1	2	3	...	25
Random number	**0.75**	**0.08**	**0.95**		
	<	<	>		
Historic pass-rate	0.9	0.5	0.9		
Pass or fail	🟩 **Pass**	🟩 **Pass**	🟥 **Fail**		

Figure 3: Finding a course that the trainee may fail.

4 COMPUTATIONAL EXPERIMENTS

This section presents the results of computational experiments aimed at the evaluation of the new approach to deal with course pass-rates. All computational experiments were conducted on a computer with Intel Core i5-7300U 2.60GHz and 8GB RAM. We coded our models and algorithms in Python. IBM ILOG CPLEX 12.10 was used to solve the mathematical programming models.

The computational experiments use the synthetic instances comprised of 5, 6, 15, 20, and 25 courses. These instances are the same as in (Mak-Hau et al., 2021). For each course, the number of sessions range from 7 to 24 sessions, the durations range from 1 to 30 weeks, and the pass-rates range from 0.5 to 1. There are 24 trainees in each instance. Table 1 summarizes the main characteristics of the 14 synthetic instances.

In what follows, Section 4.1 explains a three-stage approach to optimally allocate course-sessions to trainees taking into account the historic pass-rate. Section 4.2 evaluates the performance of the proposed trainee's progression simulation procedure.

4.1 Optimisation Procedure

In order to produce an optimal course-sessions-trainees allocations, while considering uncertainty, specifically, the course pass-rate, we propose a simulation-based three-stage solution approach (see Figure 4). The first stage determines, for each trainee, the course s/he fails and the set of courses that s/he may complete before the failed course, which can be obtained using Algorithm 1 described in Section 3. Each trainee $s \in \mathcal{S}$ thus has the associated set of to-do courses $\mathcal{A}_s = \mathcal{R}_s \cup \{c_s^*\}$. Given $\mathcal{A}_s, \forall s \in \mathcal{S}$, the second stage tackles the SSAP problem, where the objective is that the total number of course-sessions to trainees allocations is maximised. The Integer Programming model used in the second stage is as follows.

$$\max : Z^{stage2} = \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}} x_{sk} \tag{10}$$

$$\text{s.t. } (1) - (8)$$

$$x_{sk} = 0, \quad \forall s \in \mathcal{S},\ c \in \mathcal{C} \setminus \mathcal{A}_s, \quad k = n_{c-1}+1, \ldots, n_c \tag{11}$$

where (11) fixes the x-variables to 0 for the course sessions that the trainees cannot take. Note that the allocation of course-sessions to trainees obtained from the second stage is unlikely to be optimal, since we seek for a maximum number of allocations in this stage. The third stage attempts to improve the allocations by solving the following Integer Programming

Table 1: Characteristics of the problem instances.

Problem instances	Total # courses	Total # sessions	Total pairs of overlapping sessions	Total # cliques
2repsPrYr_5courses	5	60	260	31
3repsPrYr_5courses	5	72	324	36
4repsPrYr_5courses	5	84	392	42
INS_6	6	75	806	37
2repsPrYr_15courses	15	180	2048	47
3repsPrYr_15courses	15	240	2982	47
4repsPrYr_15courses	15	300	4132	47
2repsPrYr_25courses	25	300	4246	47
3repsPrYr_25courses	25	420	7246	47
4repsPrYr_25courses	25	540	11004	48
INS_994224	20	164	1666	81
INS_1743540	20	162	1676	81
INS_2233256	20	163	1676	81
INS_3353124	20	165	1696	81

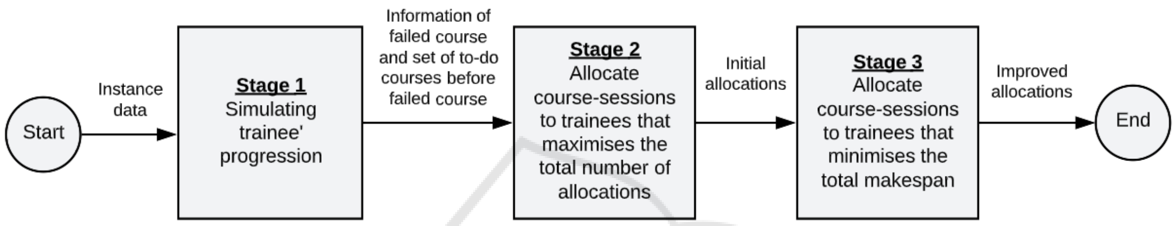

Start
Instance data
Stage 1
Simulating trainee' progression
Information of failed course and set of to-do courses before failed course
Stage 2
Allocate course-sessions to trainees that maximises the total number of allocations
Initial allocations
Stage 3
Allocate course-sessions to trainees that minimises the total makespan
Improved allocations
End

Figure 4: A high-level view of the simulation-based three-stage solution approach.

model:

$$\min : Z^{stage3} = \sum_{s\in\mathcal{S}} (T_e^s - E_s) \quad (12)$$

$$\text{s.t. } (1)-(8),\ (11)$$

$$\sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}} x_{sk} \geq Z^{stage2} \quad (13)$$

$$T_e^s \geq \sum_{k=n_{c-1}+1}^{n_c} \varepsilon_k x_{sk}, \quad \forall s \in \mathcal{S},\ c \in \mathcal{A}_s \quad (14)$$

where (13) imposes the requirement that the total number of course sessions to trainees allocations must be greater than or equal to Z^{stage2} and (14) captures the time the trainees finish their last course.

4.2 Computational Results

We apply the proposed simulation-based three-stage solution approach on the 14 synthetic instances in Table 1. For the parameter $M_{c'}, \forall(c,c') \in \mathcal{E}$, a choice of very large M can lead to slow progress in solving the MILP due to weak relaxation. In our implementation, we choose the value of $M_{c'}$ to be the latest end time among all the sessions of Course c', i.e., $M_{c'} = \max_{k=n_{c'-1}+1}^{n_{c'}} \varepsilon_k$.

The time CPLEX spent to solve the IP models in Stage 2 (construction of feasible course sessions to trainees allocations) and Stage 3 (improvement), as well as the total time are shown in Figure 5. We report the average time (in seconds) over 100 runs. In these two figures, the total time captures the running time of the three-stage approach from start to end. For Stage 2, the largest instance, i.e., 4repsPrYr_25courses required only 0.8 seconds to produce an optimal solution to the SSAP problem. For Stage 3, CPLEX can solve all the instances with $C \leq 20$ in less than 20 seconds. When $C = 25$, CPLEX required, on average, 60 seconds. Overall, the time required by CPLEX increased significantly as the number of courses increased.

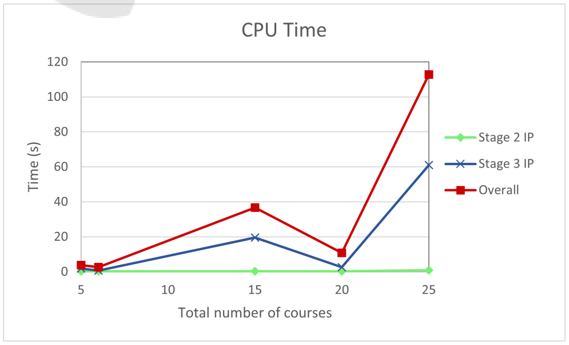


Figure 5: The average time, in seconds, CPLEX spent to solve the IP models in Stage 2 (green line) and Stage 3 (blue line) as well as the total running time of the three-stage approach (red line).

The solution quality for Stages 2 and 3 are given in Tables 2. In this table, we report the average value

Table 2: Summary of results for the SSAP problem. The best, average and worst values across 100 runs are reported.

Problem Instances	Stage 2							Stage 3					
	Avg.	Distinct sessions			make-span			Distinct sessions			make-span		
	SSA	Best	Avg.	Worst	Best	Avg.	Worst	Best	Avg.	Worst	Best	Avg.	Worst
2repsPrYr_5courses	71	30	35	41	30222	37927	44600	29	31	32	14878	17363	20681
3repsPrYr_5courses	72	36	43	48	33190	38212	46791	37	40	43	15508	17556	19428
4repsPrYr_5courses	71	39	45	50	31965	39214	49808	38	41	43	14325	16820	19456
INS_6	78	30	36	42	11287	15171	18273	29	29	31	6649	9142	11304
2repsPrYr_15courses	114	56	81	93	30817	40441	47869	52	75	84	16908	21698	27464
3repsPrYr_15courses	114	81	107	132	33498	41775	49220	84	105	121	17727	20495	26008
4repsPrYr_15courses	116	98	126	148	35808	41429	50970	97	127	144	17839	21108	27058
2repsPrYr_25courses	124	72	123	148	30299	39627	49052	78	120	144	18028	23090	29578
3repsPrYr_25courses	130	93	170	204	30124	38886	47750	88	165	197	15781	21520	27520
4repsPrYr_25courses	133	101	180	228	33127	40086	47484	100	176	219	16656	21945	29060
INS_994224	108	36	76	89	9460	14384	22816	30	71	83	5932	9310	15928
INS_1743540	109	26	76	92	8081	14595	21388	27	71	84	4259	9352	14738
INS_2233256	110	48	78	90	9922	14393	23390	42	73	84	5337	9439	15067
INS_3353124	111	20	76	92	8655	14266	20422	20	71	84	4791	9514	15361

of Z^{stage2} over 100 runs under the column titled 'Avg. SSA'. We also report the best, average, and worst values of the number of distinct sessions and the total make-span over 100 runs. For all the instances, Stage 3 was able to improve the make-span objective by about 44% on average. While minimising the total make-span in Stage 3, it is noted that the total number of distinct sessions was also optimised. Stage 3 was able to reduce the number of distinct sessions by about four sessions on average.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this paper, we studied a Defence trainee course scheduling problem where every course has a historic pass-rate and a trainee will not be allowed to progress further if they fail a course. We proposed a new approach for handling pass-rate, in which the failed course and the courses a trainee can complete before the failed course are determined prior to the optimisation procedure. A simulation-based three-stage solution approach was presented, which consists of a trainee's progression simulation stage, an initial course-sessions to trainees allocation stage, and an allocation improvement stage. The computational results show that the proposed approach yields optimal solutions to the 25-course instances in less than two minutes, on average. The proposed approach also offers information on the best-case, average-case, and worst-case solutions that can benefit managerial decision-making.

Future research can apply the introduced solution approach to an extended problem where geography and language introduce additional business rules regarding the location of courses, and language match between instructors and trainees.

REFERENCES

Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search. *ORSA journal on computing*, 6(2):126–140.

Beliën, J. and Demeulemeester, E. (2006). Scheduling trainees at a hospital department using a branch-and-price approach. *European Journal of Operational Research*, 175(1):258 – 278.

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

Brown, Gerald, G., Carlyle, W., M., Dell, Robert, F., and Brau, John, W. (2013). Optimizing intratheater military airlift in iraq and afghanistan.

Burke, E., Dror, M., Petrovic, S., and Qu, R. (2005). Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In *The next wave in computing, optimization, and decision technologies*, pages 79–91. Springer US.

Carter, M. W. (1989). A lagrangian relaxation approach to the classroom assignment problem. *INFOR: Information Systems and Operational Research*, 27(2):230–246.

Costa, D. (1994). A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76(1):98 – 110.

Davenport, J., Neu, C., Smith, W., and Heath, S. (2007). Using discrete event simulation to examine marine training at the marine corps communication-electronics school. In *2007 Winter Simulation Conference*, pages 1387–1394.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2):151 – 162.

Dell'Amico, M. and Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Ann. Oper. Res.*, 41(1-4):231–252.

Esmaeilbeigi, R., Mak-Hau, V., Yearwood, J., and Nguyen, V. (2022). The multiphase course timetabling problem. *European Journal of Operational Research*, 300(3):1098–1119.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA.

Hertz, A. and Widmer, M. (1996). An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics*, 65(1):319 – 345.

Lam, E. and Hentenryck, P. V. (2016). A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints*, 21(3):394–412.

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

Looker, J., Mak-Hau, V., and Marlow, D. (2017). Optimal policies for aircraft fleet management in the presence of unscheduled maintenance. In Syme, G., Hatton MacDonald, D., Fulton, B., and Piantadosi, J., editors, *Proceedings of MODSIM2017, 22nd International Congress on Modelling and Simulation*.

Mak-Hau, V., Hill, B., Kirszenblat, D., Moran, B., Nguyen, V., and Novak, A. (2021). A simultaneous sequencing and allocation problem for military pilot training: Integer programming approaches. *Computers & Industrial Engineering*, 154:107161.

McGinnis, M. L. and Fernandez-Gaucherand, E. (1994). Resource scheduling for the united states army's basic combat training program. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 553–558 vol.1.

Müller, T. and Rudová, H. (2016). Real-life curriculum-based timetabling with elective courses and course sections. *Annals of Operations Research*, 239(1):153–170.

Nguyen, V., Mak-Hau, V., Moran, B., and Novak, A. (2022). An efficient and exact algorithm for military timetabling and trainee assignment problems. *Computers & Industrial Engineering*, 169:108192.

Nguyen, V., Moran, B., Novak, A., Mak-Hau, V., Caelli, T., Hill, B., and Kirszenblat, D. (2018). Dancing links for optimal timetabling. *Military Operations Research*, 23(2):61–78.

Nguyen, V., Shokr, M., Novak, A., and Caelli, T. (2016). A reconfigurable agent-based discrete event simulator for helicopter aircrew training. In *International Symposium on Military Operations Research (ISMOR)*.

Novak, A., Tracey, L., Nguyen, V., Johnstone, M., Le, V. T., and Creighton, D. C. (2015). Evaluation of tender solutions for aviation training using discrete event simulation and best performance criteria. In *Winter Simulation Conference*.

Nowicki, E. and Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Manage. Sci.*, 42(6):797–813.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, 239(1):3–38.

Qi, X., Bard, J. F., and Yu, G. (2004). Class scheduling for pilot training. *Operations Research*, 52(1):148–162.

Raffensperger, J. F. and Schrage, L. E. (2008). Scheduling training for a tank battalion: How to measure readiness. *Computers & Operations Research*, 35(6):1844 – 1864. Part Special Issue: OR Applications in the Military and in Counter-Terrorism.

Rudová, H., Müller, T., and Murray, K. (2011). Complex university course timetabling. *Journal of Scheduling*, 14(2):187–207.

Scott, J. (2005). Optimally scheduling basic courses at the defense language institute using integer programming. Master's thesis, Naval Postgraduate School, Monterey, CA.

Séguin, R. and Hunter, C. (2013). 2 canadian forces flying training school (2cffts) resource allocation simulation tool. In *Winter Simulations Conference (WSC)*, pages 2866–2877.

Squires, R. R. and Hoffman, K. L. (2015). A military maintenance planning and scheduling problem. *Optimization Letters*, 9(8):1675–1688.

Teoh, C. K., Wibowo, A., and Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, 44(1):1–21.

Wang, J. (2005). A review of operations research applications in workforce planning and potential modeling of military training. Technical report, DSTO, Salisbury, South Australia.

Wang, S., Bussieck, M., Guignard, M., Meeraus, A., and O'Brien, F. (2010). Term-end exam scheduling at united states military academy/west point. *Journal of Scheduling*, 13(4):375–391.

Yang, T. and Ignizio, J. P. (1987). An algorithm for the scheduling of army battalion training exercises. *Computers & Operations Research*, 14(6):479–491.