Chapter 8
Air Traffic Controllers Planning: A Rostering Problem

Richard Conniss

The air traffic controllers (ATC) rostering problem shares some features with standard rostering problems reported in the literature, and at the same time has some unique features that required special attention. All controllers start their careers by attending a specialised training college to learn the basic skills of ATC. Once a new controller arrives at their unit, they must undertake a period of on-the-job training. At any given unit, there will be a set of controlling tasks, or positions, for which the new controller must become proficient.

Most ATC units have multiple control positions, each with unique demands and training requirements. Ideally, all controllers will eventually become endorsed (qualified) in all positions and this is where the first main difference with other scheduling problems occur. If a controller holds an endorsement in a position, they are expected to be able to staff that task as required.

This is quite different to the use of qualifications in other rostering problems. As an example, in many nurse rostering problems qualifications denote the seniority of an employee. If a senior or more qualified nurse is assigned to a task that would more normally be undertaken by a more junior colleague, this assignment is penalised in some fashion by the solution method. The senior nurse is qualified to undertake the task, but it is seen as an inefficient use of resources as salary is linked to seniority.

For controllers, the need to maintain familiarity with all tasks for which they are qualified is safety critical. Skill fade is a significant problem and can induce potentially catastrophic effects on the safe movement of aircraft. Controller's salaries are excluded from rostering decisions as their remuneration has no effect on their ability to execute a task. As such, controllers should be regularly assigned to each of the positions for which they hold endorsements. One way of measuring this

R. Conniss (✉)
University of Derby, Derby, UK
e-mail: r.conniss@derby.ac.uk

S. Berry et al. (eds.), *Guide to Computational Modelling for Decision Processes*,
Simulation Foundations, Methods and Applications,
DOI 10.1007/978-3-319-55417-4_8

familiarity is known as currency, which is used as a measure of an individual's competence for a task.

Currency is defined as the number of days since a controller worked productively in a given position. What is meant by productive work is that a controller has completed a reasonable amount of work in a position as opposite to just awaiting traffic. Merely scheduling someone to a task is not sufficient to maintain their skill set, and as such the values for currency are difficult to predict. They are updated on a daily basis. The current limit is set at 30 days, and if a controller were to violate this restriction they would have to undergo a period of retraining and re-qualify for that position. Clearly, this situation will add to the training burden of a unit and should be avoided wherever possible; achieving this can be a difficult task for most units.

Like any employee, controllers require rest breaks throughout the working day. In the civilian ATC world, there are very strict and legally binding working rules and conditions to ensure the safety of aviation operations. The rules include maximum durations for a controller to work in a position (usually 2 h) and frequency of rest breaks and meal breaks. For UK ATC operations these rules are defined in the Scheme for Regulation of Air Traffic Controllers Hours (SRATCOH) which is published by the Civil Aviation Authority.

To give a controller a break in a particular position, another qualified controller must replace them. This transfer of responsibility requires a formal handover procedure to ensure that the incoming controller is aware of the location and intentions of all aircraft receiving a service, the local weather conditions, unusual variations to normal procedures, temporary airspace restrictions and any other information deemed necessary for safe operations. This requirement prevents controllers from switching tasks instantaneously, as this hand-over process will always require at least a few minutes to complete. Usually, this is accomplished by separating controller assignments with a break. The problem is exacerbated when multiple controllers require breaks over several time periods. In this scenario, some chain of moves must be found that simultaneously gives all controllers a suitable set of breaks and maintains the required staffing for tasks throughout the day. The goal of rostering is to produce a single day roster that ensures that qualified controllers are appropriately assigned to positions, given breaks and whilst maintaining currency.

One of the most difficult aspects of the process for the watch supervisors, whom are the senior controllers in charge of daily operations, is the initial creation of the daily schedule. With so many permutations of controllers and qualifications, it can be extremely difficult to construct a roster that is feasible. An inordinate amount of a supervisor's time is spent managing the roster to meet the goals of the day. This distracts from their core responsibilities to monitor staff and maintain safe ATC operations, therefore any automated approach that could achieve this part of their daily responsibility would not only simplify their working day, but could also have positive effects on flight safety in general. Breaks are monitored continuously, which places additional pressure onto the supervisor's workload, and occasionally controllers can be left in position for an unsuitable length of time. Often late notice changes to staffing can cause problems, last minute medical appointments, meetings

off site and even the rare controlling incident can all cause disruption. Planning for these events is almost impossible and as such supervisors are constantly dealing with new inputs of information, throughout a shift.

An effective algorithm to produce valid rosters has to consider all of the above restrictions placed on controlling staff. During conversations with senior ATC staff at RAF Cranwell, a number of key requirements for a daily roster have been identified.

These are as follows:

- All operational demand for the flying program must be met by qualified controllers.
- Controllers must have suitable rest breaks.
- The system must be able to adapt to sudden changes in staffing or the operational flying task.

8.1 Mathematical Model

An appropriate model for a single day roster is required to understand the complexities involved. Given a set of controllers C with $c \in C$, a set of positions P with $p \in P$ and a set of qualifications Q containing tuples c, p, f where f denotes a currency (familiarity) value in the range $\{0, \ldots, 30\}$ for controller c on position p. The shift is divided into a set of fixed interval time slots T with $t \in T$, and the task is to find a set of assignments A, containing tuples c, p, t which represent a roster for an entire shift.

In the model defined, the day is divided into T time slots of 30 min duration for reasons of simplification. With n controllers $i = 1, \ldots, n$, m positions $j = 1 \ldots m$ and $t \in \{1, \ldots T\}$, the following matrices are defined.

$R_{i,j,t} = 1$	controller i is in position j at time t, 0 otherwise
$Q_{i,j} = 1$	controller i is qualified in position j, 0 otherwise
$D_{j,t} = 1$	position j is to be staffed at time t, 0 otherwise
$A_{i,t} = 1$	controller i is available to work at time t, 0 otherwise
$C_{i,j} = \{0, \ldots, 30\}$	currency of controller i in position j, measured in days

This leads to the following set of hard constraints:

A controller must be qualified to work in a position:

$$R_{i,j,t} \leq Q_{i,j}, \quad \forall i, j, t$$

A controller must be available to work in a position:

$$R_{i,j,t} \leq A_{i,t}, \quad \forall i, j, t$$

If there is a demand for a position, that position must be staffed:

$$\sum_{j} R_{i,j,t} = D_{j,t}, \quad \forall j, t$$

Each controller can only be assigned to a single position at a given time slot:

$$\sum_{j} R_{i,j,t} \leq 1, \quad \forall i, t$$

Controllers cannot instantaneously switch tasks, and must have a break before being assigned to a position. This allows for a formal handover of responsibility as new controllers take on a task.

$$R_{i,j,t} - \sum_{k=1}^{j} R_{i,k,t} + 1 \geq R_{i,j,(t+1)}, \quad \forall i, j, t$$

A controller must be current in a position to work:

$$R_{i,j,t} \leq C_{i,j}, \quad \forall i, j, t$$

Additionally, controllers will require sufficient rest breaks during their shift. As it stands, the RAF has no formal system for defining controller worker hours and as such rules from civilian ATC have been incorporated into the model. Essentially, the main rule to consider is that a controller cannot work for longer than 2 h in a position without a break.

$$\sum_{t=s}^{s+4} \sum_{j=1}^{m} R_{i,j,t} \leq 4, \quad \forall i \in \{1, \ldots, n\}, s \in \{1, \ldots, T-4\}$$

This problem is formulated as constraint satisfaction problem and therefore no formal objective function is required. However, an evaluation function is defined to compare solutions quality. Once a valid roster is produced, the currency value for each (controller, position, time) assignment is retrieved, and the sum of all these assignments is used as a measure of roster quality. Larger values for this function are better, implying that controllers with high currency values in a position at the beginning of the rostering period are assigned to that task in the roster.

$$\sum_{i} \sum_{j} \sum_{t} \left(R_{i,j,t} \cdot C_{i,j} \right)$$

8.2 Methodology

Given the above restrictions and requirements for successfully scheduling controllers, our proposed algorithm constructs a roster and satisfies the requirements of a particular day. It ensures controllers are able to take reasonable breaks. It is also capable of re-rostering due to short notice events.

The rostering problem can be treated as a tree of fixed size. Therefore, the structure of the developed algorithm is similar to that of the depth first search (DFS) algorithm. DFS is a procedure for traversing every node in a given graph or tree, via their connecting vertices. It does this by first selecting a starting node and then travelling along vertices, always trying to get as far from the start as possible.

Each level of the tree represents a combination of a position and time slot value. The first layer is the first position to staff in the first time slot of the shift, the second is the second position and first time slot etc. Each node is the assignment of a controller at a particular position and time. Figure 8.1 shows the structure of this approach. Only those controllers who are qualified for a position will appear as nodes at each level, and the depth of the tree is equal to the product of the number of positions and the total number of time slots determined by the required length of planning horizon. A valid roster is any path that stretches from the root node to the lowest level.

The process to select each new node begins with the creation of a list of all available controllers and is then divided into three distinct phases. Each phase filters this list, based on the particular requirements at each stage.

The first phase focusses on ensuring that only suitably qualified controllers are assigned to a given position. The node under consideration has a position parameter attached to it. The list of controllers is filtered such that all unqualified controllers are removed and the resultant list is passed to the next phase.

The second phase considers the temporal restrictions on an assignment and considers the following three rules as follows:

1. No controller can be assigned to more than a single position per time slot.
2. No controller can change position in consecutive time slots.
3. Maximum work time limits must be enforced, to allow for controller rest breaks.

Any controller which will violate any of the above restrictions will be excluded from the list.

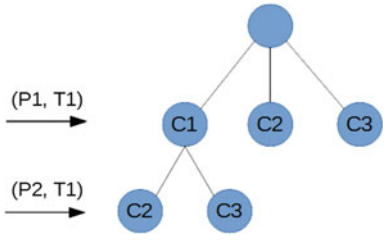

Fig. 8.1 Example of a roster tree

The final phase sorts the remaining controllers in the list into a specified order. In this instance, controllers are ranked according to their currency values, with the least current controller being assigned to the first element of the list, with the remaining controllers assigned to subsequent elements in decreasing order of number of day's currency. The resultant list is then stored with the node for use by the search algorithm.

The search is controlled by the state of the list of controllers. If after the filtering process a controller remains in the list, then this controller will be assigned to the current node and the search will move on to the next node in the tree. If at any point the list is found to be empty, this signals to the search that no feasible solution can be obtained with the current set of assignments. The search then backtracks to the previous node, removes the assigned controller and inserts the next controller in the list. The search then continues forward until another empty list of controllers is found. If after the removal of a controller from a node the search finds that the list is empty, the search immediately backtracks once more, removes the controller from the earlier node and continues this process until another controller can be assigned to a node. The pseudocode representation of the algorithm is shown in Fig. 8.2.

```
Input:
timeslots = A list of Timeslots [0,…,T]
positions = A list of Positions [0,…,m]
controllers = A list of all controllers to be considered
solution = A list of type assignment(c,p,t), where each assignment is initialised
as a position and a timeslot only. The values for the controller are left empty as
this is what will be found in the search.
solutionPointer = 0, an integer which tracks the current          assignment for
consideration in the solution. The value must be between 0 and
length(positions)*length(timeslots) i.e. the total number of assignments in the
roster.
stack = Empty stack of type assignment.
Output:
// Set up the stack for first use
Stack.push(Null)
// Null is used to show when the assignment under consideration // changes and
triggers a backtrack.
Stack.push(controllers)
While 0 ≤ solutionPointer ≤ total number of assignments
       If the first element in the stack is Null
              Then decrement the solutionPointer by 1
              And discard the Null value

       If the first element in the stack is a valid assignment
              Add the controller to the assignment in the solution
              Increment the solutionPointer by 1
              Push a Null and the controller list to the stack

       Else
              Discard the top element in the stack

End
Return solution
```

Fig. 8.2 Pseudocode for depth first search algorithm

8.3 Results

The algorithm as proposed has been implemented in C# and suit of experiments designed to validate the algorithm.

The aim of the experiments is as follows:

1. To test if the algorithm can produce feasible rosters which satisfy all the problem constraints.
2. To determine if a heuristic ordering on the list of controllers can improve performance.
3. To assess the effect of ordering the nodes in the search by the qualification requirements for each position.

To examine the effect of different heuristic sorting methods on performance, four variants of the search process were created. They are defined as follows:

- dfs: The basic version of the search. No ordering is applied to controller's currency or to the order of assignment of controllers to positions. This variant is useful for finding rosters that are at minimum feasible.
- dfsQ: The set of assignments required are ordered by the number of controllers qualified for each task. The algorithm attempts to first assign tasks with the fewest number of qualified controllers, for each time slot. The goal here is to try and force the search to backtrack as early as possible and leave the most flexibility and choice for assignment to positions with the most number of qualified controllers.
- dfsC: The set of controllers that are qualified for each position are ordered by descending currency value and presented for assignment in turn. The intention is for controllers with the most need to become current. In a particular position to be the first selected for assignment to any given task. Using currency to order the newly generated nodes is equivalent to expressing a preference to assign controllers to positions that they have not worked in for some time. It does not guarantee that the least current controller in a position is always assigned.
- dfsQC: The final variant is a mix of dfsQ and dfsC. The set of assignments are first ordered as in dfsQ and then controllers are presented in order of currency as in dfsC.

To compare the behaviour of each variant, a shared set of controller and task data was produced for each experiment. Initially, 20 controllers each with their own set of qualifications and 10 positions were considered. The planning horizon was a single 9 h day shift consisting of 18 time slots of 30 min duration. Currency values were randomised at the start of each of the four search process and each of the variants produced a roster subject to its respective heuristic ordering method. A total of 30 comparisons were produced, with each comparison comprising a single solution attempt using each of the 4 algorithms. For each new comparison, a common set of randomised currency values was used as input values for each algorithm.

Figure 8.3 shows an example roster produced by the dfs variant of the algorithm, which shows the assignment of controllers (A–Z) to positions for a particular time slot (Controller H's work schedule highlighted as an example).

In the real world, it is unlikely that all staff will be available for every shift, so the algorithm needs to be able to successfully produce rosters with reduced controller numbers. Fewer controllers imply a different distribution of qualifications and an increase in the difficulty of creating a valid schedule. Figure 8.4a shows the distribution of qualifications for the set of controllers and indicates some of the difficulties associated with finding feasible solutions, Fig. 8.4b shows the qualifications of the controllers used to construct the roster shown in Figs. 8.3 and 8.4c, d re-formulate the earlier tables to highlight the controllers needing retraining. Figure 8.4f drawn from Fig. 8.3 indicates where the controllers have gained or refreshed endorsements.

The experiment was then extended to include a reduced numbers of controllers. The aim here is to constrain the search as much as possible to check for changes in solution time and to validate the algorithms ability to deal with more realistic staffing levels. As before, each set of comparisons attempts to roster a number of controllers to 10 tasks and ensures that each controller in the roster receives an adequate number of rest breaks.

The number of controllers considered in each experiment was gradually reduced, starting with the most qualified controllers, the full titles for the positions are defined in Table 8.3.

Table 8.1 shows the average time required to generate a solution given in milliseconds for each set of comparisons, for different numbers of controllers. The basic dfs and dfsC variants have the largest average time to find a solution, although they are both relatively quick to find a feasible solution.

The dfsQC and dfsQ variants produce rosters in the shortest time. One interesting feature to note is the trend in solution time decreases from 20 to 17 controllers and then spikes at 16.

Position/Time Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
APP	S	S	S	X	X	X	K	K	K	K	Y	Y	Y	Y	S	S	S	S
CWL DIR	T	T	T	O	O	O	O	X	X	X	X	K	K	K	K	Y	Y	Y
BKN DIR	O	O	Y	Y	Y	**H**	**H**	**H**	**H**	S	S	S	S	O	O	O	O	X
CWL DEPS	**H**	**H**	**H**	**H**	S	S	S	S	O	O	O	O	X	X	X	X	Z	Z
BKH DEPS	X	X	Z	Z	Z	Z	Y	Y	Y	T	T	Z	Z	Z	Z	**H**	**H**	**H**
ADC	M	M	B	B	B	B	E	E	E	E	**H**	**H**	**H**	**H**	E	E	E	E
GND	Y	E	E	E	E	W	W	W	W	Q	Q	Q	Q	M	M	M	M	T
PAR	W	W	W	W	Q	Q	Q	Q	M	M	M	M	B	B	B	B	K	K
CWL SRA	Q	Q	Q	M	M	M	M	B	B	B	B	E	E	V	V	W	W	W
BKN SRA	B	K	K	K	K	V	V	Z	Z	Z	W	W	W	W	Q	Q	Q	Q

Fig. 8.3 Example roster produced by depth first search

(a)

Position/Controller	B	C	D	E	F	G	H	J	K	L	M	N	O	P	Q	R	S	T	V	W	X	Y	Z
APP		q	q		q	q		q	q	q		q		q		q	q	q	q		q	q	
CWL DIR		q	q		q	q		q	q	q		q	q	q		q	q	q	q		q	q	
BKN DIR		q	q		q	q	q	q	q	q		q	q	q		q	q	q	q		q	q	
CWL DEPS		q	q		q	q	q	q	q	q		q	q	q		q	q	q	q		q	q	q
BKH DEPS		q	q		q	q	q	q	q	q		q	q	q		q	q	q	q		q	q	q
ADC	q	q	q	q	q	q	q	q	q	q	q	q	q	q		q	q	q	q		q	q	
GND	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
PAR	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
CWL SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
BKN SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q

(b)

Position/Controller	K	S	T	V	X	Y	O	H	Z	B	E	M	Q	W
APP	q	q	q	q	q	q								
CWL DIR	q	q	q	q	q	q	q							
BKN DIR	q	q	q	q	q	q	q	q						
CWL DEPS	q	q	q	q	q	q	q	q	q					
BKH DEPS	q	q	q	q	q	q	q	q	q					
ADC	q	q	q	q	q	q	q	q		q	q	q		
GND	q	q	q	q	q	q	q	q	q	q	q	q	q	q
PAR	q	q	q	q	q	q	q	q	q	q	q	q	q	q
CWL SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q
BKN SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q

(c)

Position/ Controller	C	D	F	G	J	K	L	N	P	R	S	T	V	X	Y	O	H	Z	B	E	M	Q	W
APP	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q								
CWL DIR	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q							
BKN DIR	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q						
CWL	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q					
BKH DEPS	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q					
ADC	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q		q	q	q		
GND	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
PAR	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
CWL SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q
BKN SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q

Fig. 8.4 **a** Qualifications of controllers by position (*q* indicates qualified). **b** Controllers used to construct roster. **c** Grouping staff by qualifications. **d** Grouping staff by qualifications and currency, indicating training needs. **e** Controllers used to construct roster maintaining currency. **f** Endorsements completed or renewed

(d)

Position/ Controller	C	F	J	P	S	V	Y	Q	D	G	K	L	N	R	T	X	O	H	Z	B	E	M	W
APP	3	6	3	4	4	3	6		5	4	3	4	2	6	3	2							
CWL DIR	1	6	5	3	3	2	7		5	3	6	2	2	5	6	7	5						
BKN DIR	7	1	3	1	1	6	5		3	3	7	6	4	5	6	6	2	7					
CWL	7	6	6	2	3	7	6		2	7	7	6	6	7	5	2	7	6	4				
BKH DEPS	7	5	1	4	4	6	6		1	3	6	6	5	5	2	5	7	2	3				
ADC	6	7	7	6	6	7	5		7	2	4	4	2	6	5	5	6	2		7	5	6	
GND	3	3	3	2	3	7	7	1	6	4	2	5	4	3	2	3	5	2	7	6	5	6	3
PAR	3	3	7	6	5	2	1	4	7	6	6	2	3	4	2	7	6	2	2	7	2	4	4
CWL SRA	6	7	6	6	3	7	7	4	7	4	4	7	4	5	2	5	5	3	3	2	6	5	3
BKN SRA	7	3	6	2	4	1	2	5	7	4	3	3	4	3	7	2	4	4	3	3	3	2	6

(e)

Training Allocated	C	S	F	V	J	Y	P						Q	
Position/Controller	C	S	F	V	J	Y	P	H	Z	B	E	M	Q	W
APP	q	q	q	q	q	q	q							
CWL DIR	q	q	q	q	q	q	q							
BKN DIR	q	q	q	q	q	q	q	q						
CWL DEPS	q	q	q	q	q	q	q	q	q					
BKH DEPS	q	q	q	q	q	q	q	q	q					
ADC	q	q	q	q	q	q	q	q		q	q	q		
GND	q	q	q	q	q	q	q	q	q	q	q	q	q	q
PAR	q	q	q	q	q	q	q	q	q	q	q	q	q	q
CWL SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q
BKN SRA	q	q	q	q	q	q	q	q	q	q	q	q	q	q

(f)

S	3	APP;	BKN DIR;	CWL DEPS		
X	5	APP;	CWL DIR;	BKN DIR;	CWL DEPS;	BKH DEPS
K	4	APP;	CWL DIR;	PAR;	BKN SRA	
Y	5	APP;	CWL DIR;	BKN DIR;	BKH DEPS;	GND
T	3	CWL DIR;	BKH DEPS;	GND		
O	4	CWL DIR;	BKN DIR;	BKN DIR;	CWL DEPS	
H	4	BKN DIR;	CWL DEPS;	BKH DEPS;	ADC	
Z	3	CWL DEPS;	BKH DEPS;	BKN SRA		
M	4	ADC;	GND;	PAR;	CWL SRA	
B	4	ADC;	PAR;	CWL SRA;	BKN SRA	
E	3	ADC;	GND;	CWL SRA		
W	4	GND;	PAR;	CWL SRA;	BKN SRA	
Q	4	GND;	PAR;	CWL SRA;	BKN SRA	
V	2	CWL SRA;	BKN SRA			
Total	52					

Fig. 8.4 (continued)

Table 8.1 Average time required to generate a solution given in milliseconds

Search/number of controllers	20	19	18	17	16	15	14	13
dfs	1.13	19,515	9441	5627	17,322	11,050	32,078	187,795
dfsC	2.43	19,495	9447	5632	17,276	11,047	32,074	187,969
dfsQ	1.30	80	35	71	1076	593	884	7527
dfsQC	1.30	80	34	71	1080	593	884	7495

Table 8.2 Average currency value for solution

Controllers	20	19	18	17	16	15	14	13
dfs	1968	1989	1922	1887	1917	1898	1851	1860
dfsC	2408	2375	2364	2303	2069	2147	2122	2019
dfsQ	1968	1490	1922	1887	1917	1898	1851	1860
dfsQC	2408	2375	2364	2303	2069	2147	2122	2019

Table 8.3 Positions within ATC planning

Position	
APP	Approach
CWL DIR	Cranwell director
BKN DIR	Barkston heath director
CWL DEPS	Cranwell departures
BKH DEPS	Barkston heath departures
ADC	Aerodrome control
GND	Ground control
PAR	Precision approach radar
CWL SRA	Cranwell search radar approach
BKN SRA	Barkston heath search radar approach

The trend then re-emerges and continues to reduce, which implies some form of phase change is occurring in the algorithm. One possible explanation for this is that the search space size is reduced as the number of controllers is lowered. Early on in this reduction, the change in size offers a performance boost. At some point, the effect of making the problem more constrained begins to overcome the gains from the reduction in search space size, to increase the solution times.

Clearly, the heuristic sorting tends to speed up the search. Table 8.2 shows the average solution quality for each set of comparisons. The solution quality is the sum of the currencies of controllers for each assignment. Larger values suggest more high currency controllers have been assigned and therefore will have the opportunity to reset their currency. This has the effect of preventing all controllers from going out of currency over time.

These results imply that dfs is the worst performing variant, with dfsQC producing better quality rosters. Intuitively, these results make sense. The dfsQ search

tries to assign the most difficult (in terms of number of qualified controllers) first, which means the search is more likely to backtrack earlier. Each time the search backtracks, it removes large parts of the search space and reduces the number of possible nodes to be evaluated. This should have the effect of decreasing the solution time.

The dfsC presents the least current controllers for assignment first. This should lead to an upward pressure on the values of currency for the solution. The result being that on average, higher total currency values are found as the first solution. These values are not optimal, but they are larger than both the dfs and dfsQ algorithms.

Due to the structure of the implementation of the algorithm, it is relatively simple to combine both the above sorting methods simultaneously which as demonstrated, leads to an improvement in both speed and quality of generated rosters.

8.4 Discussions and Future Research

This case study has presented a new class of rostering problem, which is focussed on a real world application and genuine need for a solution method. The ability to algorithmically generate daily rosters is of great use to operational controllers and watch supervisors, alike. The above results show that feasible and useful daily rosters can be created automatically to satisfy ATC needs, in reasonable time.

The next objective is being able to plan over a longer planning horizon. ATC units tend to fix staff onto rotating shift patterns to simplify this process. At Cranwell, staff tends to work a week of the same shift type at a time. These shifts are usually designated as early, day and late shifts and their durations tend to overlap. Using the current planning system, this eliminates the need for any consideration of shift patterns. However, there remains the problem of deciding how to allocate controllers to shift patterns to ensure suitable coverage so that normal operations can occur.

One possible approach would be when a valid roster is generated for a given planning horizon, it can be used as a template to create a more general version for future use. Some controllers share equal sets of qualifications and each such group can be classified as a controller type. Due to the structure of the training program at most units, there will only ever be a few of these groups and they will be relatively stable over time. When planning shift staffing, it then becomes possible to set limits on which type of controller can be assigned to which shift and still produce a valid roster. If instead of assigning specific controllers to positions, types of controllers were assigned then this would allow the system to create general rosters and experiment with different configurations of staff. As the algorithm is deterministic and can be restarted, there is no reason for the algorithm to ever truly terminate. As each new type categorised roster is generated it can be stored and used as the basis of a new roster, by replacing each type with a controller. These general rosters can also be rated for specific attributes, e.g. the average number of positions worked by

a controller in a day, as a more varied work day can assist in preventing boredom and dissatisfaction.

Other preferences can be easily added to the system. This is useful for temporary situations like annual competency checks. Controllers are regularly checked by a colleague to ensure they are capable to continue controlling in a particular position. Usually, the scheduling of these checks is complicated by the need to maintain ATC operations and currency. Using this algorithm, check preferences can be added to a roster before it is generated thereby removing the difficulty and because the algorithm is deterministic it is guaranteed that if a valid solution exists it will be found.

The proposed extension of the algorithm will prioritise training, by using the preference ordering rules explained above. Training starts with a ground school which lays down the basic rules for a position and includes local peculiarities in procedures and processes. On completion, the student controller is placed into the live training environment and for the first time begins to work with real aircraft. Clearly, this is a critical and potentially dangerous situation and as such an experienced instructor will be given the responsibility of guiding the student through their training. What this entails is having both the instructor and student work on the same position until such point as the student is prepared for examination. After a successful exam, the student becomes endorsed and can now control independently for that single position. At RAF air traffic control units, controllers are usually only posted to that unit for 3–5 years. This causes a constant turnover of staff and creates a training burden that must be satisfied to ensure effective operations. The current method employed by the RAF is to define suitable time periods in a day that would afford the best opportunity to train and to then attempt to roster student/instructor pairs to those positions. This could be achieved by adding exceptions to the algorithm that attempt to satisfy these requests, but if such rosters are infeasible the algorithm would default to producing feasible rosters.

Finally, ensuring a fair allocation of tasks to controllers would be an obvious next step for the research. Initially, this type of investigation was hampered by the lack of any method to generate feasible rosters. Using fairness as measure of roster quality is a recent addition to the literature, but one which requires further investigation. One possible approach would use a multi-phase approach, beginning with the current algorithm to generate feasible rosters and then use a secondary heuristic method to maintain a fair task allocation over time.