Modelling the International Flight Attendant Tour of Duty Problem

Chris A. Wallace and David M. Ryan
Department of Engineering Science
University of Auckland
New Zealand
c.wallace@auckland.ac.nz
d.ryan@auckland.ac.nz

Abstract

A Tour of Duty (TOD) is an alternating sequence of duty periods and rest periods that makes up a work schedule for a number of people. An airline's flight schedule is fully crewed by the interaction of a large number of TODs, combined in such a way that each flight will have the required number of crew on board. The International Flight Attendant problem is probably the most difficult of all the TOD problems because the number of flight attendants required on a flight depends on the aircraft type, and flight attendants are qualified to operate on all aircraft types. This means that crews can split into smaller groups and recombine with other groups to form different complements. This paper describes the problem and presents a novel approach that will be used to solve the problem. This approach involves a combination of column generation and model refinement using row generation.

1 Problem Description

A Tour of Duty (TOD) is an alternating sequence of duty periods with rest periods between them, such that the combined workload does not violate any contract rules that must be satisfied (including compliance with Civil Aviation Authority flight regulations). Specifically, our work has been modelled on the contract rules applying to Air New Zealand’s International Flight Attendants.

The Flight Attendant problem is a very combinatorially complex problem. Following a flight (sector) there are a large number of combinations of work that can be completed next. The crew do not have to stay on the same type of aircraft, and can be required to reposition to other locations to enable them to perform future work. After completing a sector, crew can be rostered to immediately continue the duty and depart on another sector, or the duty period ends and a period of rest begins. If the duty period continues, crew can be required to operate the next sector or fly as a passenger on the next sector. Paxing is the term used to describe flying as a passenger. The required minimum rest time between duties depends on both the previous duty period and the next duty period. However, actual rest time achieved can be much longer to allow connections to many different subsequent sectors. Minimum rest time requirements also increase rapidly if the crew operate longer than a 12 hour duty. There is also a requirement that some of the rest periods must be extended rest periods. Often there is a choice of when the crew can be assigned these extended breaks. Finally, the best overall solution to the TOD problem depends critically on the actual flight schedule that is being operated (and to a lesser extent on the cost of the various alternatives). For example, the decision depends on the frequency of flights to ports at particular times of the week.

In addition to the problem complexity described above, there is also a complement combination complexity. The flight attendants that stay together for an entire TOD are referred to as a TOD crew complement. A number of different complements can together crew a particular sector. Each complement can then split and join up with other crew complements, on different Tours of Duty, to fully crew the following sectors of their respective TODs. For example, a 747 aircraft requires a complement of 14 flight attendants. This could be made up from a complement of 8 flight attendants and another of 6 flight attendants. There is essentially no limit on the number and combinations of complement splitting that can occur in practice. From our initial analysis of the manual solutions produced by Air New Zealand planners, we identified that for each of the four different aircraft types being operated there were generally only a small set of different crew complements being used on each aircraft type.

There are very complicated rules that have to be embedded in the TOD generation process. For example, if duty hours exceed 29 hours in 72 consecutive hours then a minimum of 24 hours rest must be given at the end of the duty period where this occurred. It should be recognised that this applies to *any* 72 hour period. Thankfully, this is the worst of the rolling time horizon rules that must be dealt with. There are other rules that apply when linking successive duty periods and rest periods. For example, at least one in every four rest periods must be a rest period containing two consecutive local nights rest[1] (except that this rules does not apply for a Trans Tasman Only TOD). There are also rules that apply to duty periods only, such as, two sector duties must be no longer than 16 hours; and new operating sectors must not start after 14 hours have elapsed since the start of the duty period. Another complication is that some rules refer to local time while others (the majority) refer to elapsed time from some reference point. Other rules refer to origin/destination of sectors rather than specific times. Every one of these rules, and all others not mentioned, must be satisfied if a TOD is to be valid[2].

Rules (such as those described above) are a feature of any TOD problem, and the underlying model formulation is not altered by the imposition of these types of rules. The Tour of Duty problem for International Pilots also has a large number of (different) rules that must be satisfied. The two problems differ in terms of fundamental restrictions/features applying to each problem. For example, the International Pilot problem has the feature that the maximum length of a duty period depends on how many pilots are rostered to fly the sector(s). Usually there are two pilots required for every sector, but if three are rostered to fly the sector(s) the overall effect of the third pilot is that the crew can operate a duty period which is longer than that which is allowed with only two pilots. Thankfully, this is a unique feature of the International Pilot problem, and we do not have to deal with this in the International Flight Attendant problem. We do however have to deal with a different type of fundamental feature, which will be described in the next paragraph.

Unlike pilots, who only fly (operate) on one aircraft type (and occasionally its other variants), flight attendants can be qualified to operate on any aircraft type. This means

[1] Defined as achieving 8 hours rest between 10pm and 8am local time.

[2] There has since been a contract round and a few of the rules described here have now changed.

in practice that there are usually many more different combinations of flights that the crew can be productively employed on (operating as opposed to paxing). Also, since the carrying capacity of each aircraft type differs, the number of flight attendants required on each aircraft varies with the aircraft type being operated. In addition, there are a number of different crew ranks (with different levels of experience/training), and while the overall number of flight attendants needed per sector is fixed, the number of each rank (cf. complement configuration) is permitted to vary. If we are to model this in an integrated way[3], then this feature adds significant additional modelling complexity to the problem.

2 Methods/Techniques used to Solve the Problem

The standard assumption, which we also applied – to restrict the types of problems to be considered – assumes that the flight schedule does not change from one week to the next. In practice this is not true. Often minor changes are made to the flight schedule to allow for one-off charter flights, or for aircraft maintenance purposes. However, the vast majority of flights[4] are not altered for a period of at least several months. The results of the optimisation program provide TODs that together fully crew the "standard week" flight schedule. Provided the flight schedule doesn't change from week to week, these TODs can be repeated *every* week with the result that every flight will always have the correct number of crew. This is true despite the fact that the individual TODs can be over two weeks in duration. We make this "standard week" assumption for simplification reasons, mainly because the lengths of some TODs are much longer than one week. We need to know the full sequence of flights for each TOD, starting from home base and finishing again at home base[5]. We can later extend this to handle fully dated solutions. The assumption we make is not very restrictive (provided the schedule doesn't change much from one week to another). There is usually enough flexibility in the solutions so that rules are not violated by making a number of simple swaps to cater for flight schedule differences. TOD planners are also very experienced in making these sorts of changes. Note the standard week formulation introduces its own complications associated with wrapping one week onto the next week (i.e. back onto itself). All the flight schedule time references for the rule checks need to be very carefully thought out to ensure that they will work correctly for every possibility.

The overall approach that we adopted to solve this problem is based on the concept of column generation. This optimisation decomposition technique (compared to heuristic techniques) is starting to be applied to, and is generally accepted as the best

---

[3] There are good reasons for doing so. Firstly, it allows for the possibility of using more highly qualified crew to replace lower qualified crew (which can sometimes make the total cost cheaper overall); i.e. the "level of service" on the flight will just be higher than normal. The second reason, is that it is desirable, for operational robustness, that different crew ranks match the duty sequences of other ranks as much as possible. The quality of solutions produced will be much better and will be more likely to be accepted in a practical implementation.

[4] For airlines that publish flight schedules and who do not rely on charter flights for their main business.

[5] To solve the full problem would involve having to solve for a long sequence of specific weeks' flights. The sequence would have to be long enough so that "end effects" did not have that much of an impact on the solutions produced. Also, the solutions produced would have a definite weekly pattern to them, which would be almost identical to the "standard week" solution proposed.

way to solve these sorts of large combinatorial problems *optimally*. It has already been applied successfully at Air New Zealand on a number of different (but related) problems. It involves generating new Tours of Duty (columns) only when they are shown to give an improvement to the current solution. This is instead of the traditional technique of generating [often only a subset of] all possible columns (commonly known as 'a priori' generation) before the optimisation is started. The columns are generated by solving a resource constrained shortest path problem. This is not an easy problem to solve, but existing solution techniques are available to solve this problem. It is also possible to combine these approaches; i.e. 'a priori' generate an initial number of columns and then use column generation to generate more columns if required.

We know that large combinatorial set-partitioning problems, such as the International Flight Attendant problem, have hundreds of thousands to millions of possible variables (i.e. Tours of Duty). Any of these TODs could end up in the optimal solution. We also know that the number of variables in an optimal solution to a set-partitioning problem is very limited (less than 200 for the Air New Zealand problem). Most variables have a solution value equal to zero at the optimal solution[6]. Even if it were possible to generate all possible variables, the time taken to solve such a large integer programming problem would be prohibitively long. A decomposition approach was required. The idea behind column generation is to start by solving the problem with a small number of variables in it. Then with post optimal analysis, price all remaining variables to see if they will improve the current solution (or price only a subset if a lot give an improvement) and add only those (best) variables back into the original problem. The method is based on iteratively refining "resource prices". This naturally generates "profitable" TODs and prevents "unprofitable" ones from being generated. When the procedure terminates, all variables that have not been added have been implicitly "priced out" of the optimal solution. In effect, the problem has been solved as if all those variables had been generated and placed in the problem at the start. Note the optimal solution must only have variables at integer values. A standard constraint branching strategy is used to obtain integer solutions. Guaranteed optimality is maintained throughout the optimisation procedure by allowing column generation to continue to generate new "profitable" TODs whilst branching to obtain integer solutions. Large problems solved with column generation can be solved in a significantly shorter time compared to 'a priori' generation.

As previously mentioned, new variables (or columns) are generated by solving a resource constrained shortest path problem. Each time this is solved it incorporates the most up to date "resource prices" so potential new columns can be compared equally with existing columns. Resource prices are provided by the set-partitioning (LP) optimisation problem (which is separate from the shortest path problem) and get updated to reflect price changes associated with new columns generated via the shortest path problem. Our shortest path problem is represented by a network where nodes represent duty periods and arcs represent rest periods[7]. A path through the network represents a partial or complete TOD. All the complicated rule checks are embedded in the shortest path solution process. When constructing the path, the rules are checked for validity at each step. Only a list of the best paths (one for each possible combination of

---

[6] Since a TOD must be either part of a solution, or not part of a solution.

[7] An alternative possibility would be to further break the nodes down so that they represent sectors rather than duty periods.

resource usage) to a given node are kept as possibilities for creating the next part of the path. In this way, only paths that guarantee an improvement to the overall solution are kept. Paths that are omitted will be generated when the problem is solved again in subsequent iterations – if they later can be shown to improve the solution.

Our initial plan was to solve the problem for Inflight Service Directors (ISDs) – Air New Zealand's highest qualified flight attendants. Each flight in the flight schedule has a requirement for exactly one ISD. Therefore this problem fits into the standard set-partitioning formulation described in existing literature. It was chosen initially because it is a much simpler problem to solve (relative to the entire flight attendant problem) and also because much of the code necessary to solve the full problem would have to be developed, including much of the column generator code and code for all of the rules that need to be satisfied. In fact, it turned out to be more efficient to set up most of the code structure with the full problem in mind rather than just for the simplified problem. The code to solve the simple problem was easily produced from a natural restriction of the full code structure.

We set up the set-partitioning LP formulation for the full problem in a special manner. This allows all the aspects of the problem to be modelled in an integrated way. The advantages of doing this are that we can enforce patterns that are desirable. An example of this is that we would like all of the crew ranks to operate the same Tours of Duty as much as possible. Another is that we will be able to allow substitution of crew by others of higher rank rather than forcing the complement (rank) configuration requirements to remain fixed per sector. The rationale behind the formulation is that we wanted to replicate sensible patterns found in the manual solutions. We identified that often the crew on a 747 aircraft sector (the largest aircraft type operated) had a full 767 crew complement, plus the additional supplementary crew required to make it up to a full 747 crew complement. To help explain the formulation in more detail, we introduce the novel idea of constraint (or row) generation in addition to the normal column generation method.

The general approach of the combined row (constraint) generation and column generation method is fairly similar to that of a standard column generation method. That is, new TODs are priced and those that improve the current solution are added. The complication arises because each TOD has an associated crew complement. We will have the situation where one complement does not fully satisfy some of the "constraints". Each "constraint" represents an aircraft sector that needs to be fully crewed. For a given "constraint" we have already decided how it will be covered in terms of the complements used; i.e. we have decided the number of complements and their configurations which will come together to crew the sector. There is a single constraint in the set-partitioning matrix for each elementary complement that will be used, and the matrix coefficients enforce exact coverage in the configuration selected. At any time the complements used to crew a sector can be changed. This is where we apply row (constraint) generation. It is possible to determine via a pricing step whether changing the complements will give an improvement to the solution. Once the decision is made to change, we can apply normal column generation on the revised set of "constraints". To show how this model differs from the standard set-partitioning model, essentially each "1" in a standard set-partitioning problem is expanded to a *matrix* representing more that one TOD and more than one constraint which together fully cover what the "1" in a standard problem covers. We have made the formulation sufficiently general so that at any time (or just during the transitionary phase of changing complement configurations), it is possible to set up two sets of constraints representing each of the two alternative complement configurations. The optimisation model will select the complement configuration that is best. Note considerable care is needed to keep track of the complement that each TOD in the problem represents, and also the "complement" each individual constraint represents (so that the correct matrix coefficients can be easily determined).

Acknowledgements

We would like to thank the staff at Air New Zealand for all their continued help associated with this project.