Column generation with a rule modelling language for airline crew pairing

Curt Hjorring, Jesper Hansen
Carmen Systems AB
Gothenburg, Sweden
curt.hjorring@carmen.se, jesper.hansen@carmen.se

Abstract

An important problem in the airline planning process is the construction of pairings, or legal lines of work, for anonymous crew members. Experimentation has shown that (delayed) column generation can produce high quality solutions, but there are difficulties in implementing a column generation method in a production environment. The rules that define legal lines of work are often complex, and in the past have been hard coded into the column generator, albeit with some parameters. European carriers often wish to modify rules, and these modifications sometimes require more than just a parameter change.

One solution to the problem of changing rules is to provide a rule system that allows the airline carriers to easily write and maintain their own rules. These rule systems often present a black box interface to client applications, i.e. we can only ask if a partial/complete pairing is legal, and the rule system returns a simple yes/no. This limited interface causes problems with a column generation approach. In this paper we show how these difficulties can be overcome with a pricing subproblem based on a duty network and a k-shortest path algorithm. Results based on production problems are given.

1 Introduction

A *pairing*, or *tour of duty*, consists of a sequence of flights to be flown by a crew member that starts and ends at the crew member's home base. For shorthaul problems the lengths of the pairings typically range from one to five days, with up to ≈ 25 flights. Each pairing has an associated cost, and must satisfy a number of safety, union, and operational rules. The airline crew pairing problem consists of finding a set of pairings that covers all the airline's flights, at minimal cost.

1.1 Earlier approaches

The airline crew pairing problem has been studied intensively for many years by OR practitioners. This is because crew costs are very large (annual figures range from

hundreds to thousands of millions of dollars), and it is computationally expensive to find high quality solutions.

The problem can be formulated as an integer problem, in which columns correspond to pairings. Unfortunately the number of columns is immense, Anbil et al. [1] estimate that a daily problem for a large American carrier has roughly 10^{14} columns. Therefore current solution practice consists of solving a sequence of restricted integer programs. Solution methodologies can be classified by how they form these restricted IPs:

generate then optimise Iteratively select a subset of flights, generate all pairings for this subset, and solve the resulting IP

delayed column generation Start with some initial set of columns. Solve an LP relaxation of the IP. Use the dual variables to generate new columns that will improve the current solution. Repeat until all columns have non-negative reduced cost.

In a generate then optimise approach, the subset of flights depends on the current solution (set of pairings). An early example of this approach is the TRIP system [2]. Here a typical subproblem selection consists of all the flights covered by a small number of randomly chosen pairings. More refined subproblem selections are discussed in Andersson et al. [3].

An early application of delayed column generation to the pairing problem is due to Minoux [7], in which it is shown that columns can be priced by solving a shortest path problem on a network with arcs representing flights and overnight connections. In the airline world the sequence of flights flown in a typical working day is known as a *duty period*, and for many rules the legality of a single duty period is independent of preceding or succeeding duties. Lavoie et al. [5] take advantage of this additional structure and form a duty period network where nodes are duty periods with state information, and arcs represent legal overnights.

Vance et al. [8] present results for flight and duty based implementations. In both versions, resources (labels) are added to each node to track legality conditions, and a *resource constrained shortest path problem* is solved. Comparisons between the two versions are difficult to make since they implement different variants of the rules, but in general the flight based version spends a larger portion of time in the pricing routine, than does the duty based version. However, the duty version cannot solve as large problems as the flight version due to memory limitations. Storage of the duty periods and the duty period connections is prohibitive for large problems (there is one arc for each legal duty period connection). In addition to this comparison, the paper presents many implementation details, including a discussion on branching strategies.

1.2 Modelling real life problems

The rules governing the legality of a pairing are complex and nonlinear, and vary from airline to airline, especially in the European arena. As an example, the Lufthansa shorthaul ruleset contains around 140 rules. These rules can be fairly

easily incorporated into the generate then optimise framework, and in fact the Carmen System [3] provides a rule modelling language that permits the airline to easily write their own rules and cost function.

The previously mentioned delayed column generation approaches require a careful manual analysis of the rules to determine how to model the rules as resources, such that each resource constrained path represents a legal pairing. This is difficult work, requiring an optimisation expert familiar with the column generation implementation. While an implementation can be configured so that some parameters can be adjusted by planners, a more powerful approach would be to combine column generation and a black box rule system. In this paper we show how that can be done using a k-shortest path algorithm within the pricing routine.

In Section 2 we discuss some of the limitations of our modelling approach. A column generation scheme based on a complete duty network is presented in Section 3. Computational results presented in Section 5 show that this was useful for small problems, but required too much memory, therefore a dynamic duty network was developed. This dynamic network starts with a relaxed network based on flight-flight connections, and iteratively refines the network where necessary. This work is presented in Section 4. Computational results are presented in Section 5, and conclusions are given in Section 6.

2 Limitations of the model

In this section we describe the rule and cost assumptions we make about the pairing problem. Firstly the pricing algorithm assumes that a legal chain cannot contain any illegal subchains, i.e. if the chain $a-b-c-d$ is legal, then the subchains $a-b$, $b-c$, $c-d$, $a-b-c$, and $b-c-d$ must also be legal. We assume that the length of a pairing is limited to a given number of days, and build this into the network.

A more stringent restriction is the form of the function defining the cost of a pairing. For the column generator to produce valid lower bounds (and therefore provably optimal solutions) the pricing problem must never overestimate the cost of a pairing, $f(p)$, and for good performance should rarely underestimate $f(p)$.

We assume that $f(p)$ is linear over the duties, plus the duty period connections. More precisely, if $p = D_1 \circ D_2 \circ D_3$ represents a pairing, p, composed of three duties, then

$$f(p) = f(D_1) + d(D_1, D_2) + f(D_2) + d(D_2, D_3) + f(D_3)$$

where $d(D_i, D_j) = f(D_i \circ D_j) - f(D_i) - f(D_j)$ is the cost of connecting two duties. For nearly all of Carmen's clients (13 major airlines), this accurately represents their desired pairing cost. For a few pairing problems credit is an important part of the cost function. The current approach can be extended to this type of cost functions, but we do not address that in this paper.

In fact a simpler cost function is often sufficient. The connection cost is simplified to only consider the last flight of the first duty, and the first flight of the second duty. This captures working day and (most) hotel cost information, but misses penalties related to consecutive "heavy" duty periods. This additional structure is taken advantage of in the relaxed duty network, presented in Section 4.

If in fact the assumed restricted function overestimates the true objective function, then the column generator will not provide valid lower bounds. However, it may still find good quality solutions, although this is more by chance than design.

3 Complete duty network

In this section we describe a column generation approach based on a duty period network in which all duty period connections are calculated in a preprocessing phase, and are then represented as unique arcs in the pricing network.

3.1 The master problem

The master problem is the set covering relaxation of the standard crew pairing formulation, i.e. we allow legs to be covered more than once (an overcover). We refer to Andersson et al. [3] for details of the formulation and a discussion of the issues involved in this conversion. The master problem can be solved using a simplex based solver such as CPLEX, but we have also implemented the hybrid subgradient/dual ascent approach described in Gustafsson [4]. In general this outperforms the CPLEX implementation.

Additional constraints are often enforced. For example, base constraints model the fact that the number of crew stationed at each base is limited. These constraints must also be considered in the pricing problem, but we do not consider them in this paper.

3.2 The pricing problem

To find attractive columns the column generator solves a pricing problem. In earlier pairing work the pricing problem is a resource constrained shortest path problem. As explained in Section 1.2 this requires a manual analysis of the rules in order to define the resources, but the result is a network in which each resource feasible path corresponds to a legal pairing. Our approach is to define a simpler network in which some paths correspond to illegal pairing, and then solve a k-shortest path problem on this network.

Nodes in the complete network correspond to duty periods, and arcs to connections between duty periods. The cost of a node is the cost according to the rule system of the corresponding duty period less the sum of the duals for the legs covered by the duty. The cost of an arc is given by the rule system ($d()$ from Section 2), and usually represents hotel and overnight allowances. The sum of the node and arc costs gives the reduced cost of the path. The network is complete in the sense that each duty-duty connection is represented explicitly in the network by a unique arc. The results of Section 5 show that this requires a prohibitive amount of memory for large problems.

The pricing routine works by first calculating the shortest path tree for the network. This returns the cheapest path in the network. If the cost is ≥ 0 there are no attractive pairing, and the pricing routine terminates. If the path has negative reduced cost, the rule system is called to see if the corresponding pairing is legal,

and also has negative reduced cost. If so the pairing is added to the LP, and the pricing routine can either terminate, or continue producing attractive pairings. If the pairing is illegal, or doesn't have negative reduced cost, then the pricing routine finds the next shortest path, and repeats the tests. This continues until one of the above termination criteria is satisfied, or all paths are enumerated.

We use the algorithm of Martins and Santos [6] to solve the k-shortest path problem (the determination of the next shortest path). Since the network is acyclic, the initial shortest path tree can be calculated efficiently by simply examining nodes in chronological order.

If there are n nodes in the network, v arcs, and at most t nodes in a path, then calculating the shortest path tree takes $O(n + v)$ time, and each k-shortest path calculation requires $O(tv/n)$ time, if the node degrees are distributed uniformly.

4 Dynamic duty network

The complete duty network is impractical for large daily problems because of the number of duty-duty connections. However, analysis of the connections showed that there is a considerable amount of structure in the way duties connect. For instance, a common pattern is that a set of say 100 duty periods, X, that terminate at some airport, can legally connect to another group of say 80 duty periods, Y, that originate from the same airport. In the complete duty network this would be represented by $100 * 80 = 8000$ arcs, but the same information can be represented by $100 + 80 = 180$ arcs (if the cost of connecting the duty periods $x \in X$ and $y \in Y$ can be represented as $f(x) + g(y)$).

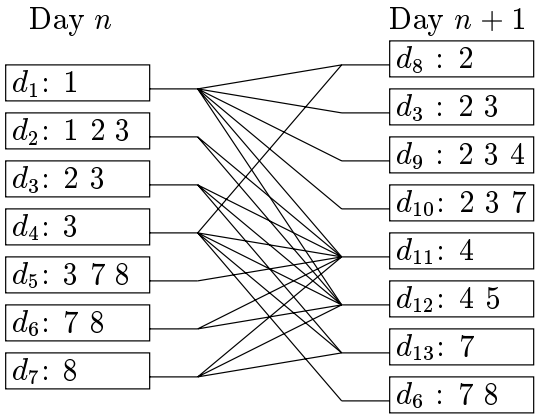
One possibility is to calculate all duty-duty connections at an airport, reduce the number of connections where possible, and then continue for the other airports. This will reduce the memory requirements compared to the full network, but will require as much time to compute as before. Also it is not obvious exactly how to reduce the connections.

Instead we take another approach and create a network based initially on connections between the boundary chunks of the duty periods, i.e. the first and last chunks of the duty periods. So duty period d_x will be connected to d_y if it is legal to connect the last chunk of d_x with the first chunk of d_y. Note that this is a relaxation, it may in fact be illegal to connect d_x to d_y.

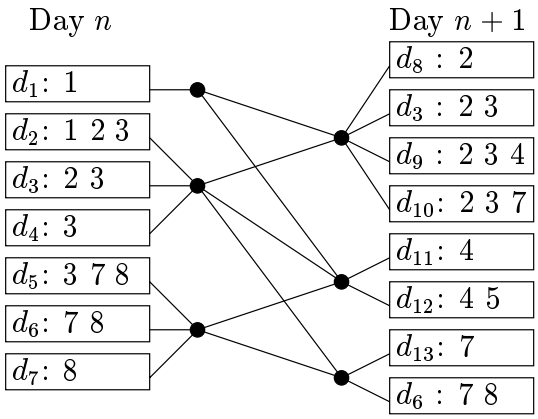
In Figure 1 we show the connections at an airport for three duty networks, the complete duty network, the relaxed version, and a refined version (the refined version is discussed in more detail in Section 4.1). First of all one can verify that Figure 1b is in fact a relaxation of Figure 1a, i.e. each duty-duty connection in 1a exists in 1b. Figure 1b is a strict relaxation, there exists paths in 1b that do not exist in 1a, e.g. d_2 on day n to d_3 on day $n + 1$.

The network is structurally different to the complete duty network of Section 3. In the dynamic duty network there are two sets of arcs, one representing duty periods (duty arcs), the other duty period connections (legality arcs). Nodes are used to group duty periods into similar sets (similar legality wise).

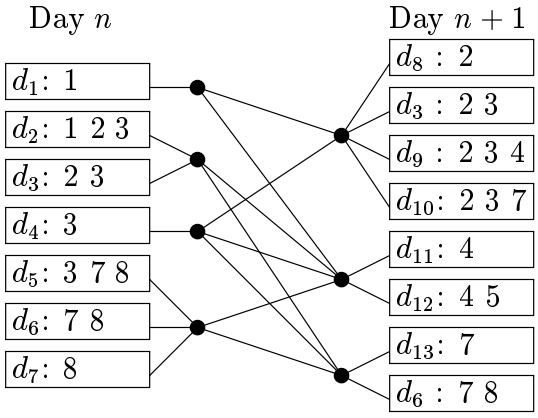
The costs associated with the legality arcs are based on the cost connecting a single chunk to another single chunk. As discussed in Section 2 this is only an



a) Complete duty network



b) Relaxed duty network



c) Refined duty network

Figure 1: Comparison of the three duty networks (only shows connections at a single airport)

approximation for several European pairing problems.

Once this relaxed duty network is created, the k-shortest path algorithm can be applied to it in the normal manner. In comparison to applying k-shortest path on the complete network, more paths will be illegal since the relaxed duty network does not prevent some of the illegal duty-duty connections. However, the initial shortest path tree calculation (the first step of the MS k-shortest path algorithm) will be much faster on the relaxed network, since there are much less arcs.

4.1 Refining the dynamic duty network

The number of illegal paths in the relaxed network can be reduced by refining the network between pairs of "problem duty periods sets." The refinement prevents occurrences of illegal connections between the duty periods in the problem sets. A pair of problem duty period sets x, y is defined as follows:

1. $x \subseteq X$, $y \subseteq Y$
2. X is a set of incoming duty periods, that are connected in the relaxed network to a set of outgoing duty periods, Y, by an arc representing a legal flight-flight connection, c, between the last flight l of $i \in X$, and the first flight f of $j \in Y$
3. It is illegal to connect $i \in x$ with $j \in y$, forall (i, j)
4. For the duty period connections to be a "problem," the duty period connections $x - y$ should occur relatively often in the set of illegal paths produced by the k-shortest path algorithm.

Let $X' = X - x$, and $Y' = Y - y$. Denote the tail of c as c_t and the head as c_h. The network is refined by "splitting" nodes c_t and c_h, to form two new nodes, c_t^* and c_h^*. The arcs in X' continue to point to c_t, but the arcs in x are redirected to c_t^*. Similarly, arcs in y are redirected to originate from c_h^*. The outgoing arcs of c_t are copied to c_t^*, and a new arc is created from c_t to c_h^*. A similar operation is performed for nodes c_h^* and c_h. Note that there is no connection between c_t^* and c_h^*, since this corresponds to an illegal connection between the duty periods.

An example of the network refinement is shown in Figure 1c. The duty sets, $\{d_2, d_3\}$, $\{d_8, d_3, d_9, d_{10}\}$, have been identified as a pair of problem duty sets, and Figure 1c shows the refinement necessary to represent the currently known illegal connections.

We allow the three sets of connections, $(i \in x, j \in Y')$, $(i \in X', j \in y)$, and $(i \in X', j \in Y')$, since with the information we have gathered so far from the rule system, we have no evidence to show that any of these connections are illegal. In fact some of these connections might be illegal. If they occur often in the k-shortest paths then in a subsequent iteration of the pricing routine they will be prevented by another network refinement. As well as improving legality information, the network can also be refined so as to improve the cost estimate.

5 Results for the two duty networks

The two duty networks were tested on five daily shorthaul cabin crew problems, as shown in Tables 1 and 2. BA Cabin is a British Airways problem, with at most three day long pairings. The other problems are from Lufthansa, and for those a maximum of four day long pairings are allowed. LH Paros 1 and LH Paros 2 are the same problem, except that LH Paros 2 contains a number of fixed connections to make the problem easier to solve.

Problem	Flights c	Flight connections k	DPs d	DP connections l
BA Cabin	290	2,888	5,235	275,808
LH FRA	268	2,409	5,438	254,382
LH Paros A320	195	2,026	3,790	352,229
LH Paros 1	927	81,722	144,624	????????
LH Paros 2 (hl)	618	27,685	24,269	$\approx 20,000,000$

Table 1: Connection comparison for some test problems

Let c be the number of flights, k the number of flight-flight connections, d the number of duty periods, l the number of duty period connections, and r be the maximum number of days in a pairing. The memory requirements of the networks are dependent on the number of nodes and arcs in the network. The complete duty period network has $O(dr)$ nodes and $O(l)$ arcs, and the relaxed network has $O(2cr)$ nodes, and $O(d+k)$ arcs. Given the data from Table 1, this represents a substantial savings in memory usage, and also results in much faster shortest path calculations. Note that due to memory limitations, we stopped complete connection generation at roughly the half way stage for LH Paros 2, and did not even attempt this for LH Paros 1.

For the dynamic network the refinement of the network will of course also consume memory. In practice we have observed the number of arcs to increase by a factor of at most ten.

Problem	flights	DPs	%'ge from optimality	CPU (mins) complete	refined
BA Cabin	290	5,235	0	3	2
LH FRA	268	5,438	0	7	4
LH Paros A320	195	3,790	0	8	5
LH Paros 1	927	144,624	0.5	out of mem	620
LH Paros 2 (hl)	927	24,269	0.03	out of mem	28

Table 2: Computational results for some daily problems

Table 2 shows that the complete duty network column generator works well for small problems such as BA Cabin, LH FRA, and LH Paros A320, but is impractical for large daily problems such as LH Paros 1 and LH Paros 2. For all problems the refined network works well, and even outperforms the complete network on the

small problems, primarily because the calculation of the duty-duty connections takes significant time.

The results of Table 2 are from an early version of the column generator, with relatively simple strategy and heuristics. Since then major improvements have been implemented, with total run times sometimes reduced by over a factor of ten. The improved implementation is now part of the standard Carmen production system, and Table 3 shows numerical results on some realistic production problems. The rules column gives the number of rules present in the rule set, and CPU times are on a 360 MHz UltraSPARC-II.

Problem	Period	Flights	Rules	Quality	CPU time
European 1	weekly	2,000	40	0.2%	25 min
European 2	weekly	2,900	140	1.0%	5 hours
European 2	month	15,000	140	1.0%	10 hours
Non-European	month	10,500	10	0.5%	2 hours

Table 3: Recent computational results for some production like problems

6 Conclusions

Delayed column generation is a viable approach for solving crew pairing problems, as is evidenced by the literature and the results of Section 5. We have shown that column generation can be integrated with a black box rule system, but that an implementation based on a complete duty network is impractical for large pairing problems. However, if the network is initially relaxed, and then dynamically refined where required, the approach can generate high quality solutions in a reasonable amount of time.

We note that if you are willing to sacrifice the flexibility of a pure black box system, the best implementation performance wise is probably a hybridisation of the k-shortest path approach with the resource constrained technique. This is because some rules are very difficult to model using resources, but might be violated only rarely. At the same time, commonly violated rules (with greater than duty period range), might be relatively easy to model as resources, but in a pure black box system would result in many k-shortest path evaluations.

Acknowledgements

This work was partially funded by the Esprit project 20.115 (Paros).

References

[1] R. Anbil, J. J. Forrest, and W. R. Pulleyblank. Column generation and the airline crew pairing problem. *Documenta Mathematica*, Extra Volume ICM III:677–686, 1998.

[2] R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent advances in crew-pairing optimization at American Airlines. *Interfaces*, 21:62–74, 1991.

[3] E. Andersson, E. Housos, N. Kohl, and D. Wedelin. Crew pairing optimization. In G. Yu, editor, *OR in Airline Industry*. Kluwer Academic Publishers, 1997.

[4] T. Gustafsson. *A Heuristic Approach to Column Generation for Airline Crew Scheduling*. Lic. thesis, Chalmers University of Technology, Gothenburg, Sweden, 1999.

[5] S. Lavoie, M. Minoux, and E. Odier. A new approach of crew pairing problems by column generation and application to air transport. *European Journal of Operations Research*, pages 45–58, 1988.

[6] E. Martins and J. Santos. A new shortest paths ranking algorithm. Technical report, Universidade de Coimbra, Portugal, Oct. 1996.

[7] M. Minoux. Column generation techniques in combinatorial optimization: A new application to crew pairing problems. In *XXIVth AGIFORS Symposium*, 1984.

[8] P. H. Vance, A. Atamtürk, C. Barnhart, E. Gelman, E. L. Johnson, A. Krishan, D. Mahidhara, G. L. Newhauser, and R. Rebello. A heuristic branch-and-price approach for the airline crew pairing problem. Technical report, School of Industrial and Systems Engineering, Georgia Tech., June 1997.