A heuristic method for solving airline crew pairing problems

Khusnul Novianingsih^{1,*} and *Rieske Hadiani*²

¹Department of Mathematics Education, Universitas Pendidikan Indonesia, 40154 Bandung, Indonesia

²Department of Mathematics, Institut Teknologi Bandung, 40132 Bandung, Indonesia

Abstract. The airline crew pairing problem is one of the optimization problems which classified as a NP-hard problem. Since the number of feasible pairings in flight schedules can be numerous, the exact methods will not efficient to solve the problem. We propose a heuristic method for solving crew pairing problems. Initially, we generate a feasible solution by maximizing the covered flights. Then, we improve the solution by constructing a procedure to avoid the local optimal solution. We test our method to an airline schedules. The computational results show that our method can give the optimal solution in short period of time.

1 Introduction

The airline crew pairing problem can be modelled as a set covering problem (SCP):

$$\min\{c\mathbf{x} : \mathbf{A}\mathbf{x} \geq \mathbf{1}, \mathbf{x} \in \{0,1\}\}, \quad (1)$$

where its columns matrix $\mathbf{A} = (a_{fp})$ represent all possible pairings in flight schedules, in which $a_{fp} = 1$ if flight f in pairing r , and $a_{fp} = 0$ otherwise. The element x_p of $\mathbf{x}$ is a decision variable that has value 1 if pairing p is selected as a solution, and $x_p = 0$ otherwise. The element c_p of $\mathbf{c}$ is cost associated to pairing p . One pairing consists a sequence of flights starting and ending at a specific airport which called as a crew base. A number of regulations and airline working rules must be complied when a pairing is constructed (see [1]).

Generally, the SCP is an NP-hard problem. Many algorithms have been resulted for solving it. The exact algorithms are mostly based on branch-and-bound or branch-and-cut [2, 3]. Other researchers have applied column generation to find global optimal [4, 5]. The integer solution are found by applying branch-and-bound where column generation is allowed in the search tree. This method is referred to as branch-and-price. In column generation technique, it is not easy to find an initial feasible solution. Solving the linear relaxation of the model in column generation technique is straightforward, but to get the integer solution another algorithm is needed.

* Corresponding author: k_novianingsih@upi.edu

Since exact methods require substantial computational effort to solve large-scale SCP, heuristic algorithms are often used to find a good or a near-optimal solution in a short time. The greedy algorithms are the most natural heuristic approach for quickly solving SCP. The simplest greedy approach is the algorithm proposed by Chvatal [6]. The algorithm is fast and easy to code, but it could rarely generate a good solution. Some researchers have tried to improve greedy algorithms by introducing some randomness. Modern heuristics, often classified as Meta-heuristics, introduce randomness in a systematic manner to improve the solution quality. They include genetic algorithms [7, 8], Meta RaPS [9], Simulated annealing [10], Neural Network algorithm [11], and ant colony algorithm [12]. Other heuristics are developed based on Lagrangian relaxation [13]. Beasley [3, 7] points out that their algorithms based on the Lagrangian relaxation and genetic algorithm were not recommended for SCP with the same column costs, since the cost play an important role in these algorithms [9]. From these literatures, most heuristics that could generate good solutions are difficult to implementation. On the other hand, some simple heuristics often could not find good results.

In this paper, we propose a new heuristic method for solving airline crew pairing problem. We allow the deadhead flights, and we want to minimize the total selected pairing cost and deadhead cost. The method works iteratively based on random search, and it consists two procedures. The first procedure is used to generate a feasible solution and to maximize the covered flights in selected pairings. Then, the solution will be improved by the second procedure. To avoid the local optimal solution, we define same parameters and criteria functions. We test our method to an airline schedules. We found that our method can work effectively and it gives good solutions.

This paper is organized as follows. We discuss the airline crew pairing model in Section 2. Section 3 explains how our method works to find the optimal solution. We present our computational results in Section 4. We give a conclusion in the last section.

2 Crew pairing problem with deadhead

We consider a crew pairing problem with deadhead. A deadhead is a set of crew who flies as a passenger in a flight. The deadhead is used to the airline for repositioning the crew to work to another flight or to return the crew to the crew base.

Let F be a set of flights and P be a set of possible pairings. For all $i \in F$ and $p \in P$, let c_p be the cost of the pairing p and d_f be the deadhead cost of the flight f . let a_{fp} has value one if flight f in pairing p , and zero otherwise. We define decision variables x_p which have value one if pairing p is selected as the optimal solution and zero otherwise. We model the crew pairing problem with deadhead as follow:

Minimize:

$$z = \sum_{p \in P} c_p x_p + \sum_{f \in F} d_f \left(\sum_{p \in P} a_{fp} x_p - 1 \right), \quad (2)$$

subject to:

$$\sum_{p \in P} a_{fp} x_p \geq 1, \forall f \in F, \quad (3)$$

$$x_p \in \{0,1\}, \forall p \in P. \quad (4)$$

The objective function (2) is to minimize total selected pairing cost. The cost consists of the operating pairing cost and the deadheading cost. Constrains (3) ensure that all flights are covered by the selected pairings.

In fact, the number of feasible pairings can be numerous, even though we have a few number of flights. This condition makes the crew pairing problem hard to be solve by exact methods. In this paper, we propose a heuristic method for solving the crew pairing problem. This method is constructed to solve the problem effectively and it can give an optimal (near optimal) solution.

3 The heuristic method for solving crew pairing problem

One of the most representative construction algorithms for SCP is the greedy algorithm proposed by Chvatal [6], which iteratively selects the most effective column until all rows are covered. In our case, a row represents a flight and a column represent a possible pairing. However fast, simple, and easy to implemented, greedy algorithm is often far from optimal solution. We also note that the solution resulted by greedy algorithm may contain redundant columns because the columns selected in a certain iteration to cover uncovered rows, can lead previously-selected column to become redundant, [14]. In our computational experience, when we work with large number of columns, implementation of removing redundant columns procedure is relatively time consuming. Based on greedy algorithm by omitting the removing redundant columns procedure, we construct a heuristic algorithm for solving crew pairing optimization problem. We also modify the greedy score in [6] for each pairing p by

$$f(p) = \frac{c_p}{|U \cap F_p|} + \sum_{f \in F} d_f t_f, \quad (5)$$

where t_f is the number of deadhead flight f which resulted by pairing p if it will be selected in the next iteration, and F_p is a set of flights covered by pairing p .

Our method consists two main procedures. The first procedure construct to obtain feasible solution. Then the feasible solution will be improved by the second procedure. Before two main procedure are invoked, we order rows F increasingly according the number of pairings which cover each flight. Let P_i be a set of pairings which cover flight i . In first procedure, for each uncover flight $i \in F$, we list candidate pairing set (CP). Then randomly, we choose one pairing in CP. To decreasing possibility of the solution is local optimal, we use two parameters in Meta-RaPS algorithm, i.e *%priority* and *%restriction* [9]. Meta-RaPS is a meta-heuristic developed by DePuy et. al. [15]. The *%priority* parameter defines the percentage of time the pairings with lowest greedy score will be added in CP. The remaining time (i.e $100 - \%priority$) defines the percentage of time pairings with larger greedy score according *%restriction* will be selected in the next iteration. So, we have two types of elements CP, that is

$$\{j \in P : f(j) = \min f(p), \forall p \in P_i\}, \quad (6)$$

atau

$$\{j \in P : f(j) \leq (1 + \%restriction \cdot \min f(p)), \forall p \in P_i\}. \quad (7)$$

For example, if we set *%priority* = 30 and *%restriction* = 40, then the probability of choosing one pairing in first type is 0.3. The remaining 0.7 define the probability one

pairing will be selected in second CP. The following algorithm gives complete explanation about procedure for constructing feasible solution.

Constructing feasible solution procedure

1. Set $i \leftarrow 1$, $U \leftarrow F$, $w_i \leftarrow 0$, for all $i \in F$, $S \leftarrow \{\}$, $x_p \leftarrow 0$, for all $p \in P$.
2. If $w_i = 0$
 - 2.1. For all $p \in P_i$, update greedy score $f(p)$.
 - 2.2. Find column $j \in P_i$ with minimum $f(p)$.
 - 2.3. Generate a random number $Q = \text{rand}(1, 100)$.
 - 2.4. If $Q < \%priority$, construct CP in (5). Otherwise, construct CP in (6).
 - 2.5. Choose one element k in CP randomly.
 - 2.6. Set $S \leftarrow S \cup \{k\}$, $x_k \leftarrow 1$, $U \leftarrow U - F_k$, and $w_i \leftarrow w_i + 1$, for all $i \in F_k$.
 - 2.7. Return to step 2.
- Otherwise, go to step 3.
3. Set $i \leftarrow i + 1$. If $i > |F|$, stop. Otherwise, go to step 2.

The objective function z resulted by the first procedure can be calculate by the following formula:

$$z = \sum_{p \in P} c_p x_p + \sum_{i \in F} d_i (w_i - 1). \quad (8)$$

The second procedure is called as the improvement procedure. The purpose of constructing second procedure is to improve solution quality of the selected pairings resulting from first procedure. First, a number of selected pairings in S , as determined by user, randomly removed. The parameter *%removed* is used to define how percentage of pairings in S will be deleted. The exactly number of removed pairings in S given by the following formula:

$$\lfloor |S| \cdot \%removed / 100 \rfloor. \quad (9)$$

After we identify uncover flights, we execute first procedure. If we get the better solution, we update the previous solution by the current solution. Otherwise, keep the old solution. We repeat both procedures until the maximum number of iteration (we use parameter *max_iter*).

Improvement procedure

1. Set $i \leftarrow 1$.
2. Randomly remove a number of pairings in S , according (8). We call them as D .
3. Set $S \leftarrow S - D$. For all $d \in D$, set $x_d \leftarrow 0$ and $w_j \leftarrow w_j - 1$, for all $F_j \in D$.
4. Find $U = \{f \in F : w_f = 0\}$.
5. If $U = \{\}$, go to step 6. Otherwise, call constructing feasible solution procedure.
6. Calculate the new objective solution z^* . If $z < z^*$, keep the old solution. Otherwise, set $z \leftarrow z^*$, and update S , w , x .
7. Set $i \leftarrow i + 1$. If $i > max_iter$, stop. Otherwise, go to step 2.

The complete steps to solve the crew pairing problem is given below.

1. List all possible pairings from a given flight schedules.
2. Set the value of *%priority*, *%restriction*, *%removed*, and *max_iter*.
3. Run constructing feasible solution procedure to obtain selected pairings S .
4. Remove a number of selected pairings in S according *%removed*.

5. Identify uncover flights. Then, run constructing feasible solution procedure for uncover flights to obtain a new solution S^* .

6. If $z(S^*) < z(S)$, update the solution S by S^* .

7. Go to Step (4).
- We repeat Step (4) – (7) until the maximum iteration is fulfilled.

4 Computational results

We use flight schedules of an airline in Indonesia as the computational study. We list all possible pairings that can be constructed from the flight schedules. The number of flights and the number of possible pairings are shown in Table 1. We obtain that each possible pairing in Table 1 contain 2-6 flights.

We implement our algorithm to obtain the optimal solution of the problem in Table 1. We set the parameter of our algorithm by `%priority = 40`, `%removed = 50`, `%restriction = 50`, and `max_iter = 12000`. For each data, we record the number of selected pairings and the value of objective function in the optimal solution (see Table 2). As the comparison, we also solve the problems by implementing Greedy Algorithm [6], and `intlinprog` function of Matlab R2015b.

Table 1. The flight schedule characteristics.

No.	#Flights	#Possible pairings
1	70	1436
2	108	2147
3	166	2303
4	185	2562
5	204	3177
6	254	4836
7	298	6012
8	312	7113
9	431	12349
10	501	18301

According our experiments, we found that, the `intlinprog` function can't solve the problem 4-10, because of the memory limitation. The data in Table 2 shows that the algorithm with the smallest value of the objective functions does not always give the less number of selected pairings in the optimal solutions, since our crew pairing optimization model is solved to determine the selected pairings with minimum cost. According the smallest value of objective function we conclude that our algorithm can give the good solution and it works efficiently, since the computation time of our algorithm for solving the above problems is not up to 4 minutes.

Table 2. The number of selected pairings and objective value in the optimal solution.

No.	# Selected pairings			Objective values		
	Our Method	Greedy	Matlab	Our Method	Greedy	Matlab
1	13	16	13	17	37	17
2	19	19	19	22	40	22
3	63	71	63	149	381	149
4	66	73	n.a	183	202	n.a
5	79	74	n.a	200	246	n.a
6	87	85	n.a	257	304	n.a
7	94	109	n.a	297	356	n.a
8	112	121	n.a	326	390	n.a
9	104	98	n.a	388	436	n.a
10	178	201	n.a	434	512	n.a

Since the value of the parameters of our algorithms will affect to the quality of the optimal solutions and their computational times, we analyze the parameters by testing the algorithm for solving the number of crew pairing problems. We obtain that the best parameters setting for our algorithm are $\%priority = 30-40$, $\%restriction = 40-60$, $\%removed = 40-60$ and $max_iter \geq 12000$.

5 Conclusion

We propose a heuristic method for solving airline crew pairing problems. The method consists of two main procedures. The first procedure is used to generate a feasible solution. Then we improve the solution by applying the second procedure. According our computational results, we conclude that our method works effectively and it gives the good solution. So, the method can be used as an alternative method for solving traditional crew pairing problems. In the future research, we will try to apply the method for solving robust crew pairing problems.

References

1. S. AhmadBeygi, A. Cohn, M. Weir, *Comput. Oper. Res.* **36**, 1284-1298 (2009)
2. E. Balas, M. Carrera, *Oper. Res.* **44**, 875-890 (1996)
3. J. E. Beasley, *Nav. Res. Log.* **37**, 151-164 (1990)
4. G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, Crew pairing at air France, *Eur. J. Oper. Res.* **97**, 245-259 (1997)
5. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance, *Oper. Res.* **3**, 491-499 (1998)
6. V. Chvatal, *Math. Oper. Res.* **4**, 233-235 (1979)
7. J. E. Beasley, P. C. Chu, *Eur. J. Oper. Res.* **31**, 85-93 (1996)

8. B. Zeren, I. Ozkol, *Journal of Intelligent Learning System and Applications* **4**, 70-80 (2011)
9. G. Lan, G. W. DePuy, G. E. Whitehouse, *Eur. J. Oper. Res.* **176**, 1387-1403 (2007)
10. L. Jacobs, M. Brusco, *Nav. Res. Log.* **42**, 1129-1140 (1995)
11. M. Ohlsson, C. Petersen, B. Soderberg, *Eur. J. Oper. Res.* **133**, 583-595 (2001)
12. Z.-Ren, Z.-Feng, L.-Ke, Z.-Zhang, *Comput. Ind. Eng.* **58**, 774-784 (2010)
13. J. E. Beasley, K. Jornsten, *Eur. J. Oper. Res.* **58**, 293-300 (1992)
14. A. Caprara, M. Fischetti, P. Toth, *Ann. Oper. Res.* **47**, 730-743 (1999)
15. G. W. DePuy, G. E. Whitehouse and R. J. Moraga, *Proceedings of the 2002 Industrial Engineering Research Conference* (2002)