An Aircraft Service Staff Rostering using a Hybrid GRASP Algorithm

Vincent Cho[1], Gene Pak Kit Wu[2] and W.H. Ip[3]
[1]Department of Management and Marketing, The Hong Kong Polytechnic University, Hong Kong, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[3]Department of Industrial and Systems Enginneering, The Hong Kong Polytechnic University, Hong Kong, China
Corresponding author e-mail: msvcho@polyu.edu.hk

Abstract: *The aircraft ground service company is responsible for carrying out the regular tasks to aircraft maintenace between their arrival at and departure from the airport. This paper presents the application of a hybrid approach based upon greedy randomized adaptive search procedure (GRASP) for rostering technical staff such that they are assigned predefined shift patterns. The rostering of staff is posed as an optimization problem with an aim of minimizing the violations of hard and soft constraints. The proposed algorithm iteratively constructs a set of solutions by GRASP. Furthermore, with multi-agent techniques, we efficiently identify an optimal roster with minimal constraint violations and fair to employees. Experimental results are included to demonstrate the effectiveness of the proposed algorithm.*
Keywords: *Hybrid approach, GRASP, Rostering, Aircraft*

1. Introduction

Staff rostering is an essential issue in many companies across different industries. To fully utilize staff members and to enhance profitability of a company, the ability to generate an effective roster is important (Tsang et al., 2007). Rostering, defined by Belew et al., (1992), is a process to arrange the employees to working hours which fulfill some constraints for a preset period of time, typically weekly or monthly. In the aviation industry, regulations must be satisfied in formulating the roster and these regulations are represented in the form of constraints. Second, rosters should satisfy the demand of the company to delivery its goods and services. In general, the rostering problem requires satisfying regulatory and other types of hard and soft constraints (Chu, 2007).
This paper is organized as follows: Section 2 presents the objectives and the constraints that make up the rostering problem and how the solution being represented. Section 3 is a theoretical framework to state interrelated concepts that construct the algorithm. In section 4, hybrid metaheuristics for solving the rostering problem are described. Experimental results of different heuristics are presented in section 5. The paper ends with the conclusions to summarize the findings and suggest some directions for future work.

2. The Rostering Problem

Staff rostering is how staff members are assigned to different planned shifts (working hours) subject to a set of resources (supply of labors) and constraints. The outcome of this problem is a list of staff assignment which covers the demand of a company. In this paper, a novel technique using GRASP is considered. The input data to this algorithm includes 1) workload demand, which is the number of labors required in different predefined patterns in a day, 2) labor supply, which is the number of labor available in a day and 3) additional requirements and constraints, which limit how labors can be assigned. Each input data is described as follows. Workload demand consists of labor demand and predefined shift patterns. Labor demand is the requirement of manpower to work in different periods in a day. Given a set of days $D = \{D_1, \ldots, D_{|D|}\}$ in a roster, the labor demand is defined as a 2D matrix s_{dj}, $d = 1, \ldots, |D|$, $j = 1, \ldots, |J|$ where s_{dj} is a non negative integer representing the number of staff needed on day d in predefined shift j. Suppose $J = \{J_1, \ldots, J_{|J|}\}$ represents a set of predefined shift patterns whose values j are the working period. The total number of possible shift patterns $|J|$ is a default setting. Suppose $I = \{I_1, \ldots, I_{|I|}\}$ represents a set of available staff whose values i are the identifications of the staff member. For any i, there is a set of feasible shift $F(i)$ which varies with his/her availability. For instance, staff member i who works in night shift today, the feasible shift, $F(i)$, of him/her will be either night shift or holiday on tomorrow according to the requirement of adequate rest time.
Labor supply is the number of staff available $|I|$. Associated with each staff member i, there is a set of assignment which is defined as x_{ijd} where $x_{ijd} = 1$ if staff member i works in shift j on the day d and 0 otherwise.
For practicality of the roster, a set of constraints is considered. The types of constraints are either hard which must be satisfied or soft which is preferred to be satisfied.

In this paper, seven types of constraints are defined in which the first five are hard and last two are soft constraints. They are as follows:

i. *Workforce supply.* The supply of labors should exceed or be equal to the workload demand. This constraint indicates that always sufficient manpower is able to be assigned.
ii. *Service demand*. Assignment of shifts to staff should exceed or be equal to the required number of staff for all shifts. This constraint indicates that always sufficient staff assign to different predefined work shifts so as to meet the workload demand.
iii. *Single shift per day.* All workers are assigned single shift every day. In other words, a worker cannot work multiple shifts in a working day.
iv. *Shift*. The gap time between 2 working days should exceed 12 hours. This constraint indicates that workers have been provided adequate rest time between work days. For example, a worker working night shift in the previous day should not be assigned morning shift and afternoon shift in the current day.
v. *Holiday*. Each worker should have at least a day off in a week. The Hong Kong government regulates any employee should not work for seven consecutive days without taking a day off.
vi. *Consecutively holiday*. For holiday of 2 days, it should be consecutive. This constraint is given by the aircraft ground service company which surveyed the staff preference. This is considered a soft constraint in our algorithm.
vii. *Shift change*. In a roster (weekly or monthly), the change of shift patterns for a worker should be minimized. This constraint indicates that a worker is assigned the same shift in a period if possible. It is because too much alternation of shift patterns for a worker very often affects his/her biological cycle. It is also a preference in the survey of the employee.

The presentation of a sampled roster is shown in Table 1. Mathematically, the solution of the problem is a vector of all x_{ijd} whose value is *1* for $\forall i, j, d$. To evaluate a roster, there is a criterion *P(m)* to measure penalty costs generated by constraints. Penalty values are assigned to each occurrence of a constraint violation.

In sum, the rostering problem as described above can be formulated as follows:

Day / Staff	Mon	Tue	Wed	Thur	Fri	Sat	Sun
1	AM	OFF	PM	PM	OFF	N	N
2	PM	PM	OFF	OFF	PM	AM	AM
3	OFF	AM	AM	AM	OFF	PM	AM
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
N	N	N	OFF	PM	PM	OFF	AM

Table 1. A sampled roster.

Let

- x_{ijd} be the decision variables which take value *1* if worker $i \in I$ works shift-pattern $j \in J$ on day $d \in D$ and *0* otherwise;
- $|M|$ be the number of constraints;
- $|I|$ be the number of workers;
- *P(m)* be the penalty cost if constraint $m \in M$ is violated;
- s_{dj} be the demand of workers in shift-pattern $j \in J$ on day $d \in D$;
- *F(i)* be the set of feasible shift-patterns for worker $i \in I$.

The aim is to minimize the following penalty function:

$$\sum_{m=1}^{|M|} P(m) \tag{1}$$

subject to the following constraints:

$$\sum_{j \in F(i)} x_{ijd} = 1, \quad \forall i, d \tag{2}$$

$$\sum_{i=1}^{|I|} x_{ijd} \geq s_{dj}, \quad \forall d, j \tag{3}$$

The objective (1) is to minimize all penalty costs generated by all soft constraints. The penalty function is defined as:

$$P(m) = c_m w_m \tag{4}$$

where c_m is the occurrence of the violation of the m^{th} constraints and w_m is the penalty cost associated to the violation of the m^{th} constraint. Constraint (2) is that all the workers are assigned exactly for one shift-pattern in a day. The function *F(i)* controls the feasible shift to be assigned to workers. The constraint indicates that not all shifts are feasible for assignment. For instance, a worker *i* working shift-pattern *N* in the previous day cannot be assigned shift-pattern AM and PM in the current day. Therefore, in this case the set of feasible shift *F(i)* to be assigned to the worker are either "N" or "OFF". Another example is the holiday constraint for workers. If a worker works 6 days consecutively, the feasible shift-pattern *F(i)* to assign should be a day-off as the worker has to take a leave in order to satisfy the government regulation. Constraint (3) requires a minimum number of workers to be rostered for each shift every day so as to meet the service demand.

3. Theoretical Framework

The algorithm pursued in solving the rostering problem is based on greedy and random metaheuristics. It requires setting up the encoding scheme and constraints to search for the solution. This paper is supposed to tune the input parameters according to some advice suggested by the roster planners in a local aircraft ground service company as well as the similar problems presented in the literatures (Chu, 2007); (Lucic & Teodorovic, 2007). To solve the short-comings in applying basic heuristics to the problem, such as large initial search space and long computational time, some local searches are applied in addition to the greedy randomized adaptive search procedure (GRASP) framework. Areas to be improved are as follows.

i. *Hybridization:* A combination of metaheuristics to solve a problem is believed to complement the weaknesses of metaheuristics but take advantages of them as suggested by the literature (Abboud, et al., 1998; Bailey et al., 1997; Caprara et al., 1998). In the rostering problem, it is difficult to seek feasible solutions and even harder to seek near-optimal solutions. With the hybridizations of greediness, randomness and a local search strategy, our proposed algorithm aims to solve the rostering problem effectively and efficiently.

ii. *Multi-objective:* In reality, a trade-off exists between rosters with different satisfactions of soft constraints. In order to deal with the trade-off, a multi-objective model will be developed so that the roster planners can balance the soft constraints. (Keung et al., 2001a, 2001b) have proposed a GAs approach to handle multiple objective functions in a multiple machine tool selection problem which minimizes both the number of tool switches and tool switching instances. This proposed algorithm takes into account multiple objectives : 1) minimize the change of shift pattern for a staff over a week and 2) maximize the chance of having a two consecutive holidays for a staff.

4. A Hybrid GRASP Algorithm

This section will first describe the basic concepts of our proposed algorithm. Then, the detail of each component will be explained.

The algorithm is under the GRASP framework to generate a set of feasible rosters. This set is with minimal violation of hard and soft constraints and the one with minimal penalty costs is served as the solution found.

The direct encoding scheme is adopted to formulate the roster. A 2D matrix is used to represent an assignment of a duty shift to a staff member on a specified day. This matrix representation is depicted in Table 2, where each element x_{id} denotes the shift-pattern j assigned to the staff member i on day d. For convenient, we assume a set of actual values {AM, PM, N, OFF} in a set of shift patterns J. This encoding scheme would also represent the 2D roster matrix by a gene string as shown in Fig. 1.

Staff \ Day	1	2	…	d	…	…	|D|
1	x_{11}	x_{12}	…	x_{1d}	…	…	$x_{1\|D\|}$
2	x_{21}	x_{22}	…	x_{2d}	…	…	$x_{2\|D\|}$
…	…	…	…	…	…	…	…
i	x_{i1}	x_{i2}	…	x_{id}	…	…	$x_{i\|D\|}$
…	…	…	…	…	…	…	…
$\|I\|$	$x_{\|I\|1}$	$x_{\|I\|2}$	…	$x_{\|I\|d}$	…	…	$x_{\|I\|\|D\|}$

Table 2. A 2D roster matrix

	Gene 1	Gene 2	Gene 3	……	Gene n
Value Assigned	x_{11}	x_{12}	x_{13}	……	$x_{\|I\|\|D\|}$
	Staff 1 Day 1	Staff 1 Day 2	Staff 1 Day 3	……	Staff |I| Day |D|

Fig. 1. A gene string to represent the 2D roster matrix

This encoding scheme guarantees that a single shift is assigned to a worker in a day. Also, it does not require much effort and high computational cost to decode the solution found by the algorithm. GRASP in this algorithm is used to generate a set of initial rosters as shown in Fig. 2.

The algorithm aims to assign a shift for a staff member greedily and randomly. With the use of a tabu list, it functions the short-term memory to store the assigned shifts to avoid duplication of assignments and ensures the correctness of shift assignments which satisfy the service demand constraint. A demand list P_d is a vector of required shift patterns to be assigned on a specific day. A tabu list T_d is a vector of binary values to indicate whether these required shift patterns are assigned *1*, or unassigned *0*. The demand list and the tabu list are in pair and thus the positions of the demand list correspond to these of the tabu list. They are defined as:

$$P_d = [\underbrace{J_1,\cdots,J_1}_{S_{d1}},\underbrace{J_2,\cdots,J_2}_{S_{d2}},\underbrace{J_j,\cdots,J_j}_{S_{dj}},\underbrace{J_{|J|},\cdots,J_{|J|}}_{S_{d|J|}}] \qquad (5)$$

```
Initialize all demandList by the service demand
Initialize all tabuList
**For each** staff 'Randomly pick a staff member
**For each** day 'Randomly pick a day
  pDayShift = retrieve yesterday's shift
  **if** demandList contains pDayShift **then**
    **if** randomStart() **then**
      tShift = pick a shift equal to pDayShift
      tabuList.update()
    **else**
      tShift = pick a shift obeying assignment rule
               except pDayShift from demandList
      tabuList.update()
    **end if**
  **else**
    tShift = pick a shift obeying assignment rule
             from shiftList
    tabuArray.update()
  **end if**
  **return** tShift
**next** day
**next** staff
```

Fig. 2. A pseudo code for GRASP

$$T_d = [T_1, T_2, \cdots, T_{\sum_{j \in J} s_{dj}}] \qquad (6)$$

The algorithm initially constructs $|D|$ pairs of demand lists and tabu lists. Each demand list has s_{dj} number of required shift patterns. The algorithm is illustrated as follows. For instance, on day 3, it requires 1 worker to work on "AM", 3 workers to work on "PM", 2 workers to work on "N" and 2 workers off duty. Given this input data, we initialize the demand list and tabu list for day 3 as P_3 = [AM, PM, PM, PM, N, N, OFF, OFF] and T_3 = [0, 0, 0, 0, 0, 0, 0, 0] respectively. In this example, all shift patterns in the demand list are unassigned so all values in the tabu list are 0. After all pairs of demand lists and tabu lists are initialized, the algorithm constructs the initial rosters iteratively. The iteration begins with randomly picking a worker *i* and a day *d*. Since the algorithm greedily assigns a shift to workers, it is necessary to retrieve the previous day's shift x_{id-1} (*pDayShift* in pseudo code*)* for worker *i*. With the information of *pDayShift*, the algorithm will check whether *pDayShift* is available on day *d*'s demand list P_d and tabu list T_d. If P_d contains *pDayShift*, worker *i* will be assigned to shift pattern as same as *pDayShift* and the assigned position in P_d will be recorded. After the assignment, the value in the corresponding position of T_d will be updated from *0* to *1*. A completely roster is generated after all staff members are assigned a shift in all days. The iteration will continue until all completely rosters are constructed. For instance, staff 2 and day 3 are randomly chosen. Let say in day 2 staff 2 is assigned shift "AM", i.e. *pDayShift* = "AM". Recalling the previous example, demand list P_3 contains "AM" in *1st* position and the value in the corresponding position of tabu list T_3 is 0, so *pDayShift* is available to assign to staff 2. Therefore, staff 2 will be assigned to work on "AM" in day 3 which is the same shift pattern assigned in previous day. After this assignment, the tabu list T_3 will be updated to [1, 0, 0, 0, 0, 0, 0, 0]. This restricts the next assignment that other staff are unable to be assigned to "AM" because the *1st* position of the tabu list is 1. If another worker, staff 4, works on "AM" in day 2, he/she cannot be assigned to work on "AM" in day 3 as the tabu list indicates that "AM" is occupied by staff 2 in this case. Staff 4 will be assigned to another type of shift patterns according to the set of feasible shifts *F(i)* and it should be still available, i.e. the correspondent position in the demand list has value 0 in the tabu list. The assignment rule is that given the previous day's shift pattern, some choices of shift patterns for the current day are not allowed. In our problem, there are 4 types of shift patterns, namely "AM", "PM", "N" and "OFF".

Table 3 shows the choices of shift patterns governed by assignment rule. The aim of the assignment rule is to provide always sufficient rest time for staff in 2 consecutive work days. This will generate a number of completely rosters, which is a parameter entered by users in the initial stage.

Previous day's shift pattern	Choices of shift patterns for current day
AM	AM, PM, N, OFF
PM	PM, N, OFF
N	N, OFF
OFF	AM, PM, N, OFF

Table 3. Choices of shift patterns by assignment rule.

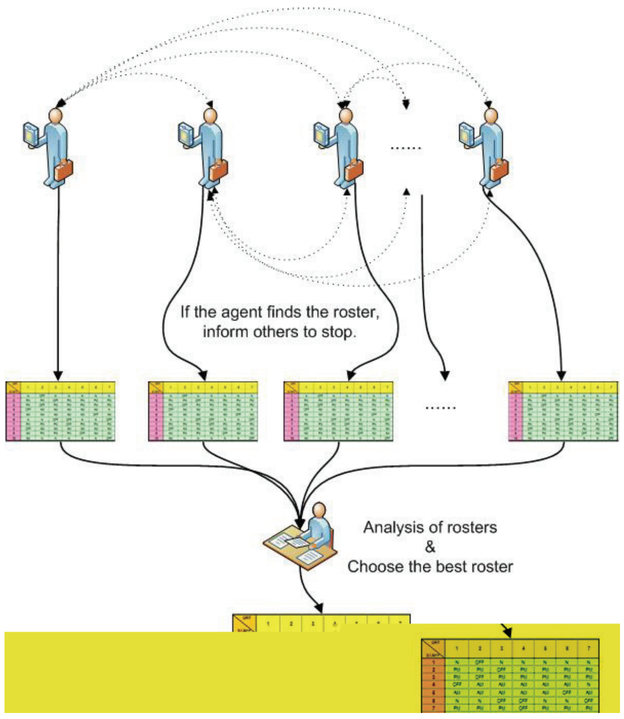

Fig. 3. Multi-agent to find the best roster.

The algorithm GRASP enables us to obtain a good feasible solution. However, the algorithm presented above introduced the local search which increases the chance the solution to be trapped in the local optima. With multi-agent as shown in Figure 3, the algorithm runs interactively to obtain the best solution.

Each agent is equipped with the knowledge of the proposed algorithm and able to communicate to one another. When an agent finds a solution, it will inform other agents to stop running and return the best solutions. The agent, who gathers all solutions, compares the quality of the rosters received and chooses the one with minimal penalty costs to be the final solution.

5. Discussion

This section describes the experiments carried out in order to assess the performance of the proposed algorithm The experiment requires the input of the workload demand, the supply of labors and the shift-patterns. The aim of the experiment is to study the effectiveness and the efficiency of the proposed algorithm and the feasibility of the implementation. A simulated program for the rostering problem solved by the proposed algorithm is implemented by Microsoft Excel with Visual Basic for application. The users can adjust the parameters in order to fine-tune the algorithm and suit their needs.

Approach	Initialization	Metaheuristics
1) Standard GAs with random initialization	Random assignment of shift-patterns to workers	Genetic Algorithms
2) GeneHunter	Genetic algorithms and improvement by a commercial black box algorithm	
3) Our algorithm	Hybrid GRASP and multi-agent	

Table 4. Initialization strategies and metaheuristics.

In order to evaluate the performance of the proposed algorithm, two more approaches are implemented. The first approach is metaheuristics by genetic algorithms with randomized initial solutions. The second approach is a genetic algorithm by GeneHunter which is a commercial Microsoft excel add-in tool implemented with some black box improvements. The third one is the proposed algorithm which hybridizes GRASP and multi-agent. Table 4 summarizes the initialization strategies and the metaheuristics used of these three approaches.

This set of experiments compared the quality of solutions generated by different approaches. For all approaches, their parameters are set to the same values in order to make them consistent. The crossover rate and the mutation rate for genetic algorithms used by different approaches is 0.98 and 0.2 respectively. The experiments consisted of 50 generated solutions of the described rostering problem while the population size of the GAs is 50. The termination condition in all runs was a maximum of 30000 iterations. These values assigned are tuned by experiments and adjustments by the rosterers.

The penalty costs for each hard constraint and soft constraint are allowed for the users to adjust in the program so to facilitate the interactivity. In our experiment, these values are defined and listed in the Table 5. Hard constraints are given higher penalty costs for the pressure to force the search to filter infeasible solutions. Soft constraints, which should be fulfilled if possible, are given smaller and different penalty costs for distinguishing the quality of the solutions. Some input data are required, besides the parameters of the program. These input data are shown in Table 6.

Table 7 reports on the results given by the standard GAs, GeneHunter and the proposed algorithm. On average, our algorithm obtained best solution among 3 approaches

m	Constraint	Type of constraints	Penalty cost (w_m)
1	Service demand	Hard	5000
2	Shift	Hard	5000
3	Holiday	Hard	5000
4	Shift change	Soft	50
5	Consecutively holiday	Soft	10

Table 5. Penalty costs for violations of constraints

Number of staff	100		
Number of days	30		
Shift patterns	{AM, PM, N, OFF}		
Staff demand (s_{dj})			
Day d / Shift j	1	……	30
AM	30	……	30
PM	30	……	30
N	20	……	20
OFF	20	……	20
Total	100	……	100

Table 6. Input data in the experiment.

Metaheuristics	Avg. CPU Time (second)	Total Penalty $P(m)$		
		Min.	Avg.	Max.
Standard GAs	1443	35580	35580	35580
GeneHunter	12647	50570		
Proposed	1410	110	160	220

Table 7. Performance of different approaches

For the standard GAs, the generation stopped to evolve after 1469 iterations although the GAs continued to run and terminated after 30000 iterations. The best result produced by GeneHunter was 50570, which is the worst among all. It also consumed the longest computational time, 12647 seconds. The average and maximum penalty cannot be obtained because GeneHunter does not enable us to do so. From the experiment, it shows that different agents produced different quality of solutions. The discrepancy of total penalty costs between the best agent and the worst agent was 390. These solutions from agents had the average and the standard deviation penalty costs of 293 and 117 respectively. With the use of multi-agent, the local optimum can be tackled and better quality of solutions can be obtained. However, it requires higher computational power to operate the agents in parallel runs. The average CPU time to run the agents was 1409.889 seconds.

6. Conclusion and Future Work

Our algorithm has been successfully implemented as a simulated program. The simulation was served as the experiment tools to analyze the result. Compared our algorithm with standard GAs, our algorithm was a robust and effective metaheuristics approach. This evaluation of the result is consistent with the literature that the hybridization of metaheuristics performed positively towards finding near optimal solutions. The GAs with randomized population initialization cannot provide good solutions while the proposed approach does it satisfactorily and is still low computational cost. The application of multi-agent reduces the chance the solution falls into the local optima but increases the chance to yield the better solution.

7. Acknowledgement

This research is supported from the Hong Kong Polytechnic University under the grant G-YG81.

8. References

Abboud, N., Inuiguchi, M., Sakawa, M. and Uemura, Y. (1998). Manpower allocation using genetic annealing. *European Journal of Operational Research*, Vol. 111, No. 2, pp. 405–420.

Bailey, R., Garner, K. and Hobbs, M. (1997). Using simulated annealing and genetic algorithms to solve staff scheduling problems. *Asia–Pacific Journal of Operational Research,* Vol. 14, No. 2, pp. 27–43.

Belew, R. K., McInerney, J. and Schraudolf, N. N. (1992). Evolving networks: using the genetic algorithm with connectionist learning, *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Ed., pp. 511-547, Santa Fe institute studies in the sciences of complexity, Addison-Wesley, Reading, MA.

Caprara, A., Focacci, F., Lamma, E., Mello, P., Milano, M. , Toth, P. and Vigo, D. (1998). Integrating constraint logic programming and operations research techniques for the crew rostering problem. *Software Practice and Experience*, Vol. 28, No. 1, pp. 49–76.

Chu, SCK. (2007). Generating, scheduling and rostering of shift crew-duties: applications at the Hong Kong international airport. *European Journal of Operational Research*, Vol. 177, No. 3, pp. 1764 – 1778.

Keung, KW., Ip, WH. and Lee, TC. (2001a). A genetic algorithm approach to the multiple tool selection problem. *Journal of Intelligent Manufacturing,* Vol. 12, No. 4, Auguest 2001, pp. 331-342.

Keung, KW., Ip, WH. and Lee, TC. (2001b). The solution of a multi-objective tool selection model using the GAs approach. *The International Journal of Advanced Manufacturing Technology*, Vol. 18, No. 11, pp. 771-777.

Lucic, P. and Teodorovic, D. (2007). Metaheuristics approach to the aircrew rostering problem. *Annals of Operations Research*, Vol. 155, No. 1, pp. 311-338.

Tsang, E., Ford, J., Mills, P., Bradwell, R., Williams, R. and Scott, P. (2007). Towards a practical engineering tool for rostering. *Annals of Operations Research*, Vol. 155, No. 1, pp. 257–277.