World Academy of Science, Engineering and Technology
International Journal of Aerospace and Mechanical Engineering
Vol:7, No:10, 2013

Scheduling Maintenance Actions for Gas Turbines Aircraft Engines

Anis Gharbi

***Abstract*—**This paper considers the problem of scheduling maintenance actions for identical aircraft gas turbine engines. Each one of the turbines consists of parts which frequently require replacement. A finite inventory of spare parts is available and all parts are ready for replacement at any time. The inventory consists of both new and refurbished parts. Hence, these parts have different field lives. The goal is to find a replacement part sequencing that maximizes the time that the aircraft will keep functioning before the inventory is replenished. The problem is formulated as an identical parallel machine scheduling problem where the minimum completion time has to be maximized. Two models have been developed. The first one is an optimization model which is based on a 0-1 linear programming formulation, while the second one is an approximate procedure which consists in decomposing the problem into several two-machine subproblems. Each subproblem is optimally solved using the first model. Both models have been implemented using Lingo and have been tested on two sets of randomly generated data with up to 150 parts and 10 turbines. Experimental results show that the optimization model is able to solve only instances with no more than 4 turbines, while the decomposition procedure often provides near-optimal solutions within a maximum CPU time of 3 seconds.

***Keywords*—**Aircraft turbines, Scheduling, Identical parallel machines, 0-1 linear programming, Heuristic.

I. Introduction

EACH year, a large amount of money is spent on the maintenance of high cost equipment such as aircraft. Such equipment consists of several life-limited parts which frequently require replacement and which never seem to fail (or need replacement) at the same time. Each failure of key parts causes the machine to fail to perform its intended function. Since supplying the inventory with spare parts is expensive, there must be a maintenance action that makes the aircraft functions without breakdowns until the inventory is replenished.

In this paper, we consider the application of scheduling maintenance actions on gas turbine engines of an aircraft. Each aircraft contains several gas turbine engines. These turbines are identical. A finite inventory of spare parts is available and all parts are ready for the replacement at any time. The inventory consists of both new and refurbished parts. Hence, these spare parts have different field lives. The objective of this paper is to find a replacement part sequence that maximizes the time that the aircraft will keep functioning without breakdowns before the inventory is eventually replenished.

The problem can be formally described as follows: A set J of n spare parts (jobs) has to be scheduled on m identical parallel turbines (machines) with. The processing time (the life cycle of the spare part) is denoted by (p_j). All the data are deterministic. Each machine processes at most one job at one time. All jobs are ready for processing at time zero. The problem consists in maximizing the minimum machine completion time denoted by C_{min}. This problem is denoted by $Pm \mid\mid C_{min}$.

The $Pm \mid\mid C_{min}$ received scant attention in the scheduling literature. The first polynomial-time approximation scheme has been derived by Woeginger [5]. Tan and He [4] proposed two asymptotically optimal algorithm classes which have worst-case ratios very close to the upper bound for any given m. Tan et al. [3] considered the on-line variant problem with two uniform machines. They presented a comprehensive lower bound on the competitive ratio, which is a piecewise function of machines speed ratio s, and derived an algorithm which is optimal for any $s \geq 1$.Haouari and Jemmali [1] proposed the first exact algorithm which includes several distinctive algorithmic features. It is based on tight lower and upper bounds as well as an effective symmetry-breaking branching strategy.

A very closely related problem is minimization of maximum completion time denoted by $Pm \mid\mid C_{max}$. Clearly, for the case of two machines, the $Pm \mid\mid C_{min}$and $Pm \mid\mid C_{max}$are equivalent. A lot of literature has been devoted to this problem. For a comprehensive survey in $Pm \mid\mid C_{max}$, the reader is referred to Mokotoff [2].

In this paper, two models have been developed. The first one is an optimization model which is based on a 0-1 linear programming formulation, while the second one is an approximate procedure which consists in decomposing the problem into several two-machine subproblems.

The paper is organized as follows. In Section II, a 0-1 linear programming model is proposed. Section III is devoted to the presentation of the decomposition-based approach. Finally, the experimental performance of our algorithms is analyzed in Section IV.

II. A 0-1 Linear Programming Model

Decision variables:

x_{ij}= 1 if job j is assigned to machine i
0 otherwise.
C_{min} = Denote the minimum completion time.
The 0-1 linear programming model is described in the

Anis Gharbi is with Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia. He is also a research fellow with BEM Bordeaux Management School Bordeaux, France (phone: 0096614676829; fax: 0096614678657; e-mail: a.gharbi@ ksu.edu.sa).

following:

Maximize C_{min}

Subject to:

$$C_{min} - \left(\sum_{j=1}^{n} P_j x_{ij} \right) \leq \forall i = 1,.....,m\, 0 \quad (1)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j = 1,.....,n \quad (2)$$

$$C_{min} \geq 0 \quad (3)$$

$$x_{ij} \in \{0,1\} \quad (4)$$

Constraints (1) ensure that the completion time of minimum machine does not exceed the completion time of any other machine. Constraints (2) guarantee that each job is assigned to exactly one machine. Constraints (3) ensure that the minimum completion time is nonnegative. Constraints (4) show the binary nature of the decision variables x_{ij}.

This model has been implemented using the optimization software LINGO 10. A CPU time limit of 2 minutes has been fixed. If this time limit is reached, then the program delivers the best found solution as well as a computed upper bound.

III. A Decomposition Approach

In this section, we propose an approximate procedure which consists in decomposing the problem into several two-machine subproblems. Each subproblem is optimally solved using the 0-1 linear programming model.The decomposition approach can be described by the following steps:

Step 1. Construct an initial solution using Constraints (1)-(4).

Step 2. Rank machines in non-increasing order of their completion time (the maximum machine will have rank one and the minimum machine will have rank *m*).

Step 3. Determine the optimal solution on the subproblem defined by the jobs assigned on machine 1 and machine *m* using the 0-1 linear program.

Step 4. If the value of C_{min} is improved, then update it and go to Step 3.

Step 5. If there is no improvement, then replace the maximum machine by the next ranked machine in the two-machine subproblem and reiterate.

Step 6. If all the *m*-1 subproblems yield no improvement then the procedure stops.

The model will keep looping until one of the following stopping criteria occurs (the processing times are assumed to be ranked according to the non-increasing order):

1. C_{min} reaches an upper bound U which is computed as follows:

$$U = \min\left\{ \left\lfloor \sum_{j=1}^{n} P_j / m \right\rfloor, \left\lfloor \left(\sum_{j=1}^{n} P_j - \max\left(p_1, p_m + p_{m+1}, \sum_{j=1}^{n} P_j / m \right) \right) / (m-1) \right\rfloor \right\}$$

2. There are no improvements between C_{min} and all other machines.
3. The CPU time reaches 2 minutes.
4. The number of iterations (number of two-machine subproblems) reaches 1000.

IV. Experimental Results

We assessed the two proposed approaches on two sets of randomly generated data. In the first data set (A), all processing times are generated using the discrete uniform distribution on [1, 25]. On the other hand, each instance of the second data set (B) includes 50% of jobs with processing times equal to 25, whereas the remaining processing times are generated using the discrete uniform distribution on [1,20]. This set is assumed to simulate real data where 50% of the parts are new (with maximum life time equal to 25 months) and the remaining parts are refurbished (having thus less life times).

TABLE I
Performance of the 0-1 Linear Programming Model on Data Set A

		***m*=2**	***m*=4**	***m*=6**	***m*=8**	***m*=10**
***n*=20**	$Time_{avg}$	0.30	1.30	26.20	94.20	114.40
	$Time_{max}$	1	4	120	120	120
	Solved	10	10	9	3	1
	Gap	0	0	0.17	2.86	4.37
***n*=50**	$Time_{avg}$	0.20	0.70	29.20	73.60	99.00
	$Time_{max}$	1	1	120	120	120
	Solved	10	10	9	4	2
	Gap	0	0	0.07	0.73	1.50
***n*=100**	$Time_{avg}$	0.20	0.50	73.00	108.90	120.00
	$Time_{max}$	1	1	120	120	120
	Solved	10	10	4	1	0
	Gap	0	0	0.21	0.64	1.04
***n*=150**	$Time_{avg}$	0.10	0.70	28.90	95.60	120.00
	$Time_{max}$	1	1	120	120	120
	Solved	10	10	8	3	0
	Gap	0	0	0.05	0.25	0.79

TABLE II
Performance of the 0-1 Linear Programming Model on Data Set B

		***m*=2**	***m*=4**	***m*=6**	***m*=8**	***m*=10**
***n*=20**	$Time_{avg}$	0.50	120.00	74.70	120.00	120.00
	$Time_{max}$	2	120	120	120	120
	Solved	10	0	4	0	0
	Gap	0.00	1.74	1.48	10.77	6.05
***n*=50**	$Time_{avg}$	0.00	26.90	96.10	120.00	120.00
	$Time_{max}$	0.1	106	120	120	120
	Solved	10	9	3	0	0
	Gap	0.00	0.04	0.97	1.39	4.11
***n*=100**	$Time_{avg}$	0.10	34.80	106.00	112.70	120.00
	$Time_{max}$	1	120	120	120	120
	Solved	10	10	2	1	0
	Gap	0.00	0.00	0.39	1.12	3.29
***n*=150**	$Time_{avg}$	0.10	42.30	107.00	120.00	120.00
	$Time_{max}$	1	120	120	120	120
	Solved	10	0	2	0	0
	Gap	0.00	0.06	0.35	1.10	1.84

The number of parts n has been taken equal to 20, 50, 100 and 150. The number of turbines m has been taken equal to 2, 4, 6, 8 and 10. For each pair (n,m), 10 instances have been generated, resulting in a total number of 400 instances.

A. Performance of the 0-1 Linear Programming Model

Tables I and II depict the performance of the proposed 0-1 linear programming model on data sets A and B, respectively. For each pair (n,m), the following performance indicators are reported:

- $Time_{avg}$: the average CPU time (in seconds) computed over the corresponding 10 instances
- $Time_{max}$: the maximum CPU time computed over the corresponding 10 instances.
- *Solved*: the number of solved instances out of 10.
- *Gap*: the average gap computed with respect to the best obtained upper bound.

From Tables I and II, we observe that the average CPU time required by the 0-1 linear programming model increases as the number of machines increases. In particular, almost all instances with 10 machines are not optimally solved within the time limit of 2 minutes. Also, it seems that Data Set B is more difficult to solve. Indeed, except for m=2, the model is not able to solve all the instances for a given number of turbines.

B. Performance of the Decomposition Approach

Tables III and IV depict the performance of the decomposition approach on data sets A and B, respectively. From these tables, it is clear that the decomposition approach performs much better than the 0-1 linear programming model both on instances A and B. Indeed, the maximum required CPU time never exceeds 3 seconds. Moreover, the solutions provided by the decomposition approach are very close to the upper bounds. In particular, all of the largest instances with 100 parts and 10 turbines are optimally solved for both types of data sets. Also, it should be noticed that the decomposition approach does not seem to be sensitive to the data generation, which indicates a promising robustness behavior.

TABLE III
PERFORMANCE OF THE DECOMPOSITION APPROACH ON DATA SET A

		m=4	m=6	m=8	m=10
n=20	$Time_{avg}$	0.90	1.40	1.20	2.00
	$Time_{max}$	3	2	2	3
	Solved	10	10	10	10
	Gap	0.23	0.54	3.31	7.02
n=50	$Time_{avg}$	0.60	1.20	0.70	1.80
	$Time_{max}$	1	2	1	2
	Solved	10	10	10	10
	Gap	0.00	0.41	0.18	0.00
n=100	$Time_{avg}$	0.50	1.70	0.70	2.10
	$Time_{max}$	1	2	1	3
	Solved	10	10	10	10
	Gap	0.00	0.00	0.00	0.00
n=150	$Time_{avg}$	0.50	2.00	1.00	2.70
	$Time_{max}$	1	2	1	3
	Solved	10	10	10	10
	Gap	0.00	0.00	0.00	0.00

TABLE IV
PERFORMANCE OF THE DECOMPOSITION APPROACH ON DATA SET B

		m=4	m=6	m=8	m=10
n=20	$Time_{avg}$	1.10	1.10	1.60	1.80
	$Time_{max}$	5	2	2	2
	Solved	10	10	10	10
	Gap	1.84	1.64	11.65	6.05
n=50	$Time_{avg}$	0.50	1.50	1.50	2.50
	$Time_{max}$	1	2	2	3
	Solved	10	10	10	10
	Gap	0.04	0.45	0.87	3.56
n=100	$Time_{avg}$	0.30	1.60	1.20	2.20
	$Time_{max}$	1	2	2	3
	Solved	10	10	10	10
	Gap	0.00	0.00	0.04	0.00
n=150	$Time_{avg}$	0.40	2.00	1.30	2.00
	$Time_{max}$	1	2	3	3
	Solved	10	10	10	10
	Gap	0.00	0.00	0.09	0.00

ACKNOWLEDGMENT

This paper is supported by the National Plan for Science & Technology (NPST) program at King Saud University (project number 11-MAT-1491-2).

REFERENCES

[1] Haouari, M. andJemmali, M. (2007) "Maximizing the minimum completion time on parallel machines" 4 OR, 6, pp 375-392.
[2] Mokotoff, E. (2001) "Parallel machine scheduling problems: a survey", *Asian Pacific Journal of Operational Research*, 18 (2), pp 193-24.
[3] Tan, Z., He Y., Epstein L. (2004) "Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data", Information and Computation, 196, pp 57-70.
[4] Tan, Z. and He, Y. (2004) "Ordinal scheduling problem and its asymptotically optimal algorithms on parallel machine system", *Science in Chaina Ser. F Information Sciences*, 47 (2), pp 161-169.
[5] Woeginger, G. (1995) "A polynomial-time approximation scheme for maximizing the minimum machine completion time", *Operations Research Letters*, 20, pp149-154.