Solving the Aircraft Engine Maintenance Scheduling Problem using a Multi-objective Evolutionary Algorithm

Mark P. Kleeman
Department of Electrical and Computer Engineering
Graduate School of Engineering & Management
Air Force Institute of Technology
Wright-Patterson Air Force Base
Dayton, OH 45433
mark.kleeman@afit.edu

Gary B. Lamont
Department of Electrical and Computer Engineering
Graduate School of Engineering & Management
Air Force Institute of Technology
Wright-Patterson Air Force Base
Dayton, OH 45433
gary.lamont@afit.edu

Categories and Subject Descriptors

I.2.8-Problem Solving, Control Methods, and Search; G.1.6-Optimization; J.7-Computers in Other Systems; H.4.2-Types of Systems

General terms

Design, algorithms

Keywords

Multi-objective Evolutionary Algorithms, scheduling, aircraft engine scheduling, military application, flow shop, job shop

1. INTRODUCTION

Scheduling problems are a very common research topic. This is because, for efficiency reasons, our world relies heavily on schedules and deadlines. Aircraft engine maintenance is no exception. The United States Air Force has many planes that it must keep up and running. But with the downsizing that has occurred in recent years, the number of planes that are operational has become more critical. This means that every effort needs to be made to ensure that not only are the engines repaired in an efficient manner, but that their component's scheduled maintenance cycles are in sync so that the engine has fewer trips to the logistics maintenance center [1].

2. THE PROBLEM

The U.S. Air Force has many aircraft in its arsenal. Its fleet is aging and it appears that plans are for the aircraft to be around for a while longer. But many of these engines are beyond their design life. In fact, 97% of all F100 engines and 84% of all F110 engines will be past their design life by the year 2010 [1]. One essential thing that the Air Force relies upon is a dependable fleet. If the Air Force does not have reliable aircraft, then they must procure redundant systems in order to ensure the success of any mission with a high degree of certainty. To achieve this type of dependability, the old "fix it when it breaks" mentality or the on condition maintenance (OCM) practice had to be revamped in order to overcome much of the uncertainty that is inherent in this type of repair mentality. A reliability-centered maintenance (RCM) philosophy is necessary to keep the planes flying for longer periods of time.

Aircraft engines are brought into the shop for two reasons: unscheduled maintenance and routine maintenance. Routine maintenance occurs on predefined intervals, when the engine component is still operational, but history shows that the mean time to repair (MTTR) for that component is almost up.

With routine maintenance as our focal point, our problem is set-up in the following manner. When an engine comes into a logistics workcenter for repair, it is logged into the system. Aircraft engines are commonly divided into smaller subcomponents which can be worked on individually and then recombined. For this problem, we divided the engine into five logical subcomponents. We assumed that the maintenance shop had one specific work area for each of the components. This is an example of the job shop problem, but with a twist. After all maintenance is completed on an engine, each engine component's mean time to repair (MTTR) is compared with other components on the engine. If there is a large disparity among the MTTRs then a component swap may be initiated with another engine in an effort to ensure the MTTRs of the components of a particular engine are similar. This is done so that the engine can have more time on wing (TOW) and less time in the shop.

Once the swaps are done, the engine is reassembled and tested as a whole to ensure functionality. This represents a small flow shop problem in that each engine has to have maintenance done first, followed by swapping and then testing. So the problem at hand is actually a hybrid of two

[1]The views expressed in this article are those of the authors and do not reflect the official policy of the United States Air Force, Department of Defense, or the U.S. Government.

scheduling problems. It could be viewed as a sort of meta scheduling problem, with a flow shop dictating the path of the engines and a job shop dictating the multiple paths of the sub-components.

The problem was written as a static scheduling problem, where the program receives an initial set of inputs and it outputs results based on them. A dynamic scheduling problem allows the user to add additional items to the schedule while the program is running. For our problem, it is reasonable to assume that the logistics center is always alerted ahead of time as to what engines are coming and when. Since the shop is alerted a priori of the items to be scheduled, we felt that static scheduling was more appropriate for our problem domain.

Our input file is made up of all the information that may be useful for our scheduling problem: arrival times, MTTR values, due dates, and priority levels . Some of the information is not used currently, but it was included for completeness and for future research.

The chromosome representation used can be effectively divided into two parts: the precedence order for the engines requiring repair and the component swaps. The first portion of the chromosome uses a job-based GA representation. Each allele in the chromosome represents an engine. The first allele listed has precedence over all other engines for component repairs. So if it has three components that need repair, they will be scheduled first in those respective repair locations.

The second portion of the chromosome determines the precedence of the component swaps, the number of swaps, the components to be swapped, and the engines that the components are to come from. The most difficult part of this problem is determining how many swaps to allow. By having a lot of swaps, you increase the problem complexity which could decrease the algorithm efficiency. We decided that as the number of engines is increased, the number of swaps should increase proportionally. So our chromosome was developed in order to accommodate the varying number of swaps based on the number of engines that are to be repaired. But we did not want to force the program to make swaps when none were needed. So the number of swaps can be zero or go as high as the total number of engines being repaired.

Figure 1 shows a representation of a chromosome for four engines. Note that the first four alleles are for the engine scheduling precedence, while the last 12 alleles are for the swaps. There are four possible swaps. Each swap is represented by three alleles, the two engines that will be swapping components and the actual component to be swapped. Note that for this example, there will be only three swaps because one of the swaps has a zero for the component, signifying no swap is to occur.

We determined that the best fitness functions to use for this problem were the makespan and the aggregate swap count.

The makespan, is equivalent to the completion time of the last job to leave the system. This calculation takes into account the time it takes to comply with the scheduled maintenance and complete the proposed swaps based on their precedence, and test the reassembled engine.

The second fitness function is the aggregate swap count. The aggregate swap count is calculated after the makespan. The program compares the MTTR for all the components of an engine. If the MTTR is outside the provided tolerance levels for MTTR, then the aggregate swap count is increased. This is done for each engine and the summation of the results is the aggregate swap count.

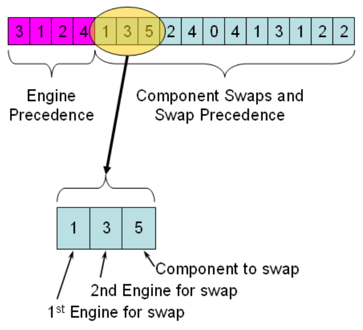

Figure 1: Example chromosome representation

3. GENMOP SOFTWARE DESIGN

The problem was integrated into the General Multi-objective Parallel Genetic Algorithm (GENMOP). GENMOP was originally designed as a real-valued, parallel MOEA. This section briefly describes MOEAs and then discusses in more detail GENMOP operations.

GENMOP is an implicit building block MOEA that attempts to find good solutions with a balance of exploration and exploitation. It is a Pareto-based algorithm that utilizes real values for crossover and mutation operators. The MOEA is an extension of the single objective GENOCOP algorithm [7, 6]. GENMOP has been used for bioremediation research [4, 5]and used to optimize quantum cascade laser parameters [2, 3].

The algorithm has four crossover methods and three mutation methods that are used. Each operator is chosen based upon an adaptive probability distribution, where the operators that produce the most fit individuals have their selection probability increased.

4. RESULTS AND ANALYSIS

For these initial data results we looked at two instances: the five engine and ten engine scheduling problem.

The analysis of the average number of swaps shows that the number of swaps for the five engine problem is about right. Of the five swaps allocated, 7 out of 10 instances utilized more than 50% of them. But with respect to the ten engine problem, only one of nine surpassed the threshold. This indicates that our baseline number of swaps is adequate for the 5 engine problem, but for 10 engines it may be limiting the efficiency of our system. This means our linear swap scaling factor of 1:1 was too much. A more appropriate method would be to implement a variable scaling factor that increases the number of swaps in a chromosome when a certain threshold is reached, or decreases the number when a declining threshold is reached. This would make our chromosome lengths vary while the program is running.

Looking at the average Overall Nondominated Vector Generation (ONVG), there really is no trend that one can decipher between the various instances, with the exception of the 10 engine, 1000 generation instance. It has a large mean and standard deviation. This is because of several runs that approached 70 members on the Pareto front. Most of these members had the same fitness values, but had slightly different schedules.

As for the makespan, the trend is as one would expect, the average makespan of the members tends decrease as the number of generations and/or number of population members is increased. It is also interesting to note that the standard deviation tends to creep higher as the number of generations is increased. This can be expected to happen in a multi-objective problem. As population members spread out along the Pareto front, the more diverse the population becomes and therefore the standard deviation increases.

Figure 2 shows a comparison of Pareto front members at the two ends of the spectrum. One instance was generated using 1000 population members and 1000 generations, while the other used just 10 population members and 10 generations. Note that there are many duplicate members on the Pareto front for both instances. Each instance has more than 30 nondominated points, but no instance has more than five glyphs on the graph. This illustrates that many different schedules have the same fitness values.

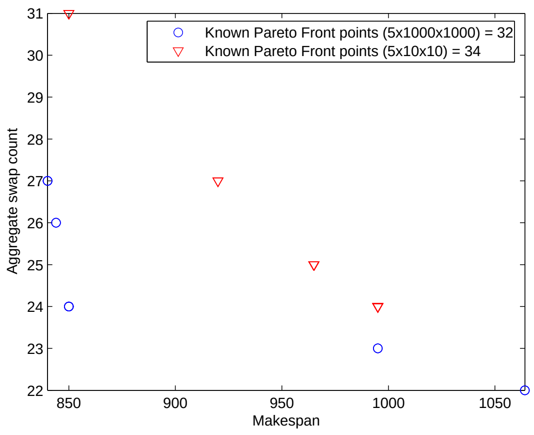

Figure 2: Comparison of the Pareto front members for the 5 engine instance

5. CONCLUSION

The goal of our research is to decrease engine downtime. We want our application to find a schedule that repairs engines rapidly and increases the time on wing, by ensuring similar MTTR values for the components on each engine.

Our initial research confirms that an aircraft engine maintenance scheduling problem can be effectively solved with an MOEA and in particular our GENMOP algorithm. Our chromosome representations permits the handling of swaps effectively and thus GENMOP tends to converge toward an acceptable Pareto Front. We also found that the standard deviation of the makespan tends to increase as the number of generations increases.

With a validated model and MOEA there are other avenues that can be investigated. The problem domain can be integrated with other MOEA algorithms for performance comparisons. Higher engine numbers should be addressed. The current instances at higher population rates and generations could be studied in an effort to determine where the efficiency of the algorithm decreases, with respect to obtaining new members of the Pareto front. Moreover, changing the chromosome representation may improve search effectiveness. Currently the last 3/4 of our chromosome is for the engine swaps. We would like to reserve five alleles for the swaps and let each one represent a component. Then the values could be implemented for the alleles in a linked list. We are currently trying to determine if the whole chromosome should be a linked list, or if multiple data types in the chromosome are better. Additionally, we need to analyze the utility of the four crossover and three mutation operators. Determining which ones are effective and which ones are not in order to improve the algorithm either by limiting the operator pool to a fewer number of operators, or modifying less effective operators in order to generate better results. Currently, better heuristic crossover algorithms are being developed for this type of problem.

The future work on this problem can give us further insight into the algorithm domain and make this an effective tool that the Air Force can use in its logistic centers.

6. REFERENCES

[1] T. Dues. Quality Engine Development and Sustainment. In *Proceedings of the 2001 Defense Manufacturing Conference*, Oklahoma City Air Logistics Center, November 2001. Defense Manufacturing Conferences.

[2] T. A. Keller. Optimization of a Quantum Cascade Laser Operating in the Terahertz Frequency Range Using a Multiobjective Evolutionary Algorithm. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 2004.

[3] T. A. Keller and G. B. Lamont. Optimization of a Quantum Cascade Laser Operating in the Terahertz Frequency Range Using a Multiobjective Evolutionary Algorithm. In *17th International Conference on Multiple Criteria Decision Making (MCDM 2004)*, volume 1, December 2004.

[4] M. R. Knarr. Optimizing an In Situ Bioremediation Technology to Manage Perchlorate-Contaminated Groundwater. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 2003.

[5] M. R. Knarr, M. N. Goltz, G. B. Lamont, and J. Huang. *In Situ* Bioremediation of Perchlorate-Contaminated Groundwater using a Multi-Objective Parallel Evolutionary Algorithm. In *Congress on Evolutionary Computation (CEC'2003)*, volume 1, pages 1604–1611, Piscataway, New Jersey, December 2003. IEEE Service Center.

[6] Z. Michalewicz. Evolutionary computation techniques for nonlinear programming problems. *International Transactions in Operational Research*, 1(2):175, 1994.

[7] Z. Michalewicz and C. Z. Janikow. Genocop: a genetic algorithm for numerical optimization problems with linear constraints. *Commun. ACM*, 39(12es):223–240, 1996.